

Development and Validation of a Mobile Autonomous Omnidirectional Soccer Robot

Dem Fachbereich Maschinenbau
an der Technischen Universität Darmstadt
zur
Erlangung des Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

D i s s e r t a t i o n

vorgelegt von

Dipl.-Ing. Rainer Immel

aus Mainz

Berichterstatter:	Prof. Dr.-Ing. R. Nordmann
Mitberichterstatter:	Prof. Dr.-Ing. H. Birkhofer
Tag der Einreichung:	5. Dezember 2005
Tag der mündlichen Prüfung:	15. Februar 2006

Erklärung

Ich erkläre, dass ich die vorliegende Arbeit – abgesehen von den ausdrücklich genannten Hilfen – selbstständig verfasst habe.

Dexheim, 18. November 2005

Rainer Immel

Danksagung

Ich danke Dieter Grimm, Jan Sporleder und Eduardo Antunes de Souza, die durch ihre Mithilfe in Form von Praktika und Diplomarbeiten meine Arbeit wesentlich vorangebracht haben. Ich danke ebenfalls Josefin Meusinger, die mir die Finanzierung des Projekts vermittelte und die Durchführung neben meiner regulären Arbeit ermöglichte. Außerdem danke ich Hans-Dieter Weiland, der mir immer wieder hilfsbereit und unkompliziert mechanische Teile fertigte. Mein besonderer Dank gilt Lars Peter Thiesen für das Korrekturlesen der Dissertation und für die moralische Unterstützung.

Dexheim, im November 2005

Rainer Immel

Abstract

An autonomous mobile robot is developed and experimentally validated. The robot is designed according to the regulations of the RoboCup Middle Size League 1999. Two basic skills are identified as underdeveloped in the robots participating in the 1999 RoboCup competition: Vision and agility. The goal of this research project is to develop a robot equipped with these two skills.

All aspects of the robot being a mechatronic system are covered, including an omnidirectional propulsion system, an omnidirectional vision system, and a simple ball handling mechanism for dribbling a soccer ball. Video cameras are used as sensors, and a video image recognition algorithm is developed that computes both robot and ball position. A robust trajectory planning algorithm based on a global coordinate system is developed. It computes a robot trajectory surrounding a set of control points in given orientations, accounting for the interaction between robot and ball. A controller closes the feedback loop from sensors to actuators keeping the robot on its predetermined course.

An experimental validation examines sensor accuracy, performance of actuators and controllers, and the overall skill to dribble a ball along a given trajectory. The vision system computes both robot and ball position 25 times per second. The average error of the robot position is below 10 mm, the average error of the ball position is below 5% of the ball's distance from the robot. All driving tests are carried out using a benchmark slalom course. The actuators behave as expected according to their specifications, they allow dynamic driving with longitudinal and lateral accelerations of up to 4 m/s^2 . In that extreme case, the controllers manage the robot to stay on its course with maximum total and lateral deviations below 0.25 m and 0.1 m, respectively. A stable ball dribbling is possible at a maximum velocity and lateral acceleration of 1.5 m/s and 2.5 m/s^2 , respectively.

Zusammenfassung

Ein autonomer mobiler Roboter wird entwickelt und experimentell validiert. Der Roboter ist gemäß den Regeln für die RoboCup Middle Size League 1999 ausgelegt. Es wird festgestellt, dass bei den teilnehmenden Robotern in diesem Jahr zwei grundlegende Fähigkeiten unterentwickelt waren: Sensorik und Wendigkeit. Ziel der vorliegenden Forschungsarbeit ist die Entwicklung eines Roboters, der mit diesen beiden Fähigkeiten ausgestattet ist.

Alle Aspekte des Roboters als mechatronisches System werden berücksichtigt, insbesondere ein omnidirektionales Antriebsystem, eine omnidirektionale Sensorik, sowie ein einfacher Mechanismus zum sicheren Führen eines Fußballs. Videokameras werden als Sensoren eingesetzt und es wird ein Algorithmus zur Bilderkennung entwickelt, der die Positionen von Roboter und Ball berechnet. Ein robuster Algorithmus zum Planen von Bahnkurven wird ebenfalls entwickelt; er basiert auf einem globalen Koordinatensystem. Er berechnet die Bahnkurve für den Roboter so, dass sie in vorgegebenen Orientierungen um definierte Kontrollpunkte führt; dabei wird die Interaktion zwischen Roboter und Ball berücksichtigt. Ein Regler schließt den Kreis zwischen den Sensoren und den Aktoren und hält somit den Roboter auf dem vorbestimmten Kurs.

Zur experimentellen Validierung werden die Genauigkeit der Sensoren, das Verhalten der Aktoren und der Regler, sowie die Fähigkeit untersucht, einen Ball entlang einer vorgeschriebenen Bahnkurve zu dribbeln. Die Sensorik berechnet die Positionen von Roboter und Ball 25 Mal pro Sekunde. Dabei ist der mittlere Fehler der Roboterposition kleiner als 10 mm, der Fehler der Ballposition ist kleiner als 5% der Entfernung des Balls vom Roboter. Ein Slalomkurs wird als Bahnkurve für alle Tests verwendet, bei denen der Roboter sich bewegt. Die Aktoren erlauben ein dynamisches Fahren mit Beschleunigungen von 4 m/s^2 in longitudinaler und lateraler Richtung. Bei diesen extremen Beschleunigungen beträgt die maximale Kursabweichung des Roboters weniger als 0.25 m insgesamt und weniger als 0.1 m in lateraler Richtung. Ein sicheres Balldribbeln ist möglich bei einer maximalen Geschwindigkeit von 1.5 m/s, dabei treten laterale Beschleunigungen von 2.5 m/s^2 auf.

Contents

Motivation	1
1 Robot System	9
1.1 Design Considerations	9
1.2 Mathematical Description	14
1.2.1 Kinematics	14
1.2.2 Kinetics	18
1.2.3 Equation of Motion	20
1.3 Robot Hardware	23
2 Actuators	27
2.1 Technical Specifications	27
2.2 Motor Dimensioning	31
2.3 Dribbling Mechanism	36
3 Trajectory Planning	39
3.1 Two Body Problem	39
3.2 Interaction between Passive Object and Robot	40
3.2.1 Equation of Motion for the Passive Object	40
3.2.2 Kinematics and Kinetics of Interaction	41
3.3 Trajectory Parameterizations	43
3.4 Piecewise Polynomial Approach	44
3.5 Minimal Curve Length Approach	46
3.5.1 Trajectory Shape	47
3.5.2 Trajectory Speed	52
3.5.3 Robot Trajectory	56
3.6 Circle and Tangent Approach	58
3.6.1 Trajectory Shape	59
3.6.2 Trajectory Speed	60
3.6.3 Robot Trajectory	62
3.7 Weighted Polynomial Approximation	66

4	Controllers	71
4.1	Motor Controller	71
4.2	Robot Motion Controller	74
4.3	Robot Position Estimation	78
5	Sensors	81
5.1	Image Processing	81
5.1.1	Image Capturing Hardware	82
5.1.2	Color Recognition	84
5.1.3	Outline Detection	85
5.1.4	Object Localization	87
5.2	Robot Self Localization	88
5.2.1	Definitions	88
5.2.2	Least Squares Approach	91
5.3	Ball Localization	93
5.3.1	Ball Position in Video Image	94
5.3.2	Ball Position in Robot Coordinates	95
6	Experimental Validation	97
6.1	Trajectory Planning	97
6.1.1	Minimal Curve Length Method	99
6.1.2	Circle and Tangent Method	108
6.2	Sensors	118
6.2.1	Calibration	118
6.2.2	Robot Self Localization	120
6.2.3	Ball Localization	123
6.3	Robot System	127
6.4	Actuators	133
6.5	Controllers	139
6.5.1	Motor Controller Tuning	139
6.5.2	Robot Position Estimation	145
6.5.3	Robot Motion Controller Tuning	148
6.6	Ball Handling	155
	Conclusions	161
A	Robot Implementation	169
A.1	Technical Data Components	169
A.2	Camera Calibration Program	172
A.3	Robot Control Program	177

B Numerical Solutions	183
B.1 Minimal Curve Length Method: Trajectory Shape	183
B.2 Minimal Curve Length Method: Trajectory Speed	188
B.3 Minimal Curve Length Method: Robot Trajectory	191
B.4 Circle and Tangent Method: Trajectory Shape	193
B.5 Circle and Tangent Method: Trajectory Speed	197
B.6 Circle and Tangent Method: Robot Trajectory	202
B.7 Image Processing: Robot Self Localization	206
B.8 Image Processing: Ball Localization	209
Bibliography	211

List of Figures

1	Robot as mechatronic system	6
2	Robot developed in this thesis	7
1.1	Omnidirectional wheel, single roller type	10
1.2	Omnidirectional wheel, double roller type	10
1.3	Omnidirectional wheel, cutting edge type	11
1.4	Four different wheel and roller configurations	13
1.5	Coordinate systems definitions	15
1.6	Robot mass distributions	20
1.7	Robot assembly without cover	24
1.8	Robot assembly with cover	25
1.9	Robot photograph without cover	26
2.1	Motor dimensioning: Electric power scheme	32
2.2	Motor dimensioning: Speed–torque characteristic	35
2.3	Ball dribbling mechanism photographs	38
3.1	Interaction between robot and ball	42
3.2	Procedure for solving the Two Body Problem	44
3.3	MCL approach: Trajectory through control points	48
3.4	MCL approach: Velocity profile along trajectory	53
3.5	CAT approach: Trajectory shape	59
3.6	CAT approach: Velocity ² –distance diagram	61
3.7	CAT approach: Trajectory speed	62
3.8	CAT approach: Angle geometry for arbitrary trajectory	63
3.9	CAT approach: Secant geometry for circular sections	64
3.10	CAT approach: Weight factor ψ and damping ratio δ	66
3.11	Weighted polynomial approximation	67
4.1	Motor controller scheme	72
4.2	Robot motion controller scheme	75
4.3	Robot position estimation: Video image time delay	78

4.4	Robot Position Estimation: Position extrapolation	79
5.1	Image processing algorithm	82
5.2	Camera hardware setup	83
5.3	RGB color space	84
5.4	HSI color space	85
5.5	YUV color space	85
5.6	Pixel neighborhood patterns	86
5.7	Soccer field geometry	89
5.8	Ball geometry in video image	94
5.9	Ball projection in robot coordinates	95
6.1	MCL trajectory: <i>Eight</i> course	103
6.2	MCL trajectory: Position and orientation	103
6.3	MCL trajectory: Velocity and acceleration	104
6.4	MCL trajectory: Wheel speeds	105
6.5	MCL trajectory: Wheel torques	105
6.6	MCL trajectory: Wheel loads	106
6.7	MCL trajectory: Friction coefficients	107
6.8	MCL trajectory: Wheel powers	108
6.9	CAT trajectory: <i>Eight</i> course	112
6.10	CAT trajectory: Position and orientation	112
6.11	CAT trajectory: Velocity ² -distance diagram	113
6.12	CAT trajectory: Velocity and acceleration	114
6.13	CAT trajectory: Wheel speeds	114
6.14	CAT trajectory: Wheel torques	115
6.15	CAT trajectory: Wheel loads	116
6.16	CAT trajectory: Friction coefficients	117
6.17	CAT trajectory: Wheel powers	117
6.18	Sensor testing: Video camera calibration pattern	118
6.19	Field of play for driving tests	120
6.20	Sensor testing: Robot position	121
6.21	Sensor testing: Ball position	124
6.22	Sensor testing: Ball position error	126
6.23	Robot system: Single roller wheel	129
6.24	Robot system: Double roller wheel	130
6.25	Robot system: Cutting edge wheel	132
6.26	Actuator testing: Wheel speeds	134
6.27	Actuator testing: Wheel torques	136
6.28	Actuator testing: Wheel speed-torque diagram	138
6.29	Motor current controller: Initial step response	140

6.30	Motor current controller: Final step response	140
6.31	Motor speed controller: Initial wheel velocity and torque . . .	142
6.32	Motor speed controller: Final wheel velocity and torque	142
6.33	Motor speed controller: Initial trajectory	144
6.34	Motor speed controller: Final trajectory	144
6.35	Robot position deviation estimation	146
6.36	Robot motion controller: MCL trajectory, slow, open loop . .	149
6.37	Robot motion controller: CAT trajectory, slow, open loop . . .	149
6.38	Robot motion controller: MCL trajectory, slow, closed loop . .	151
6.39	Robot motion controller: CAT trajectory, slow, closed loop . .	151
6.40	Robot motion controller: MCL trajectory, fast, closed loop . .	152
6.41	Robot motion controller: CAT trajectory, fast, closed loop . .	152
6.42	Robot motion controller: MCL controller velocity, fast	154
6.43	Robot motion controller: CAT controller velocity, fast	154
6.44	Ball dribbling: MCL trajectory, slow	156
6.45	Ball dribbling: CAT trajectory, slow	156
6.46	Ball dribbling: MCL trajectory, fast	159
6.47	Ball dribbling: CAT trajectory, fast	159
A.1	Camera calibration program: Screen 1, video image	173
A.2	Camera calibration program: Screen 2, ξ - η coordinates	174
A.3	Camera calibration program: Screen 3, camera conspectus . .	176
A.4	Robot control program: Screen 1, video images	179
A.5	Robot control program: Screen 2, ξ - η coordinates	181
B.1	Circle and tangent shape – case 1: Generic tangent point . . .	194
B.2	Circle and tangent shape – case 2: Circle to circle, odd	195
B.3	Circle and tangent shape – case 3a: Circle to equal circle . . .	196
B.4	Circle and tangent shape – case 3b: Circle to circle, even . . .	197
B.5	Circle and tangent speed – step 1: Velocity constraints	198
B.6	Circle and tangent speed – step 2: Acceleration constraints . .	199
B.7	Circle and tangent speed – step 3: Deceleration constraints . .	199
B.8	Circle and tangent speed – step 4: Final design	200

List of Tables

2.1	Estimated masses of robot components	28
2.2	Wheel loads during acceleration	30
2.3	Robot technical specifications for design	31
2.4	Motor dimensioning: Robot load cases	32
2.5	Motor dimensioning: Results for propelling motors	34
3.1	MCL approach: Trajectory design in three steps	47
3.2	MCL approach: Trajectory shape algorithm	51
3.3	MCL approach: Trajectory speed algorithm	57
5.1	Outline detection sample pseudo code	86
5.2	Robot self localization algorithm	92
6.1	Common points of generic MCL and CAT trajectories	98
6.2	MCL trajectory control points	100
6.3	CAT trajectory control points	109
6.4	Sensor testing: Robot position	122
6.5	Sensor testing: Ball position	125
6.6	Robot system: Physical parameters (double roller wheels) . . .	127
6.7	Robot system: Power consumption	132
6.8	Motor controller tuning: Initial parameter set	139
6.9	Motor controller tuning: Final parameter set	145
A.1	Robot components and suppliers	169
A.2	Robot components technical data	171
B.1	CAT approach: Trajectory speed algorithm	201
B.2	Robot self localization coefficients	208
B.3	Ball localization coefficients	210

Nomenclature

Variables, Parameters

a_{acc}	maximum longitudinal acceleration (accelerating), $\frac{\text{m}}{\text{s}^2}$
a_{dec}	maximum longitudinal acceleration (decelerating), $\frac{\text{m}}{\text{s}^2}$
a_{lat}	maximum lateral acceleration, $\frac{\text{m}}{\text{s}^2}$
a_{max}	maximum specified translational acceleration, $\frac{\text{m}}{\text{s}^2}$
a_n	acceleration in normal direction, $\frac{\text{m}}{\text{s}^2}$
a_t	acceleration in tangential direction, $\frac{\text{m}}{\text{s}^2}$
c_k	coefficients used for trajectory parameterization (MCL method)
d_0	deceleration free rolling zone (CAT method), m
d_{cal}	calibration horizon, m
d_i	modified distance from pixel to field marking object, m or m^2
\mathbf{d}_i	modified distance from pixel to ball outline, pixel^2
\tilde{d}_i	geometrical distance from pixel to field marking object, m
$\tilde{\mathbf{d}}_i$	geometrical distance from pixel to ball outline, pixel
D_B	passive object viscous damping, $\frac{\text{kg}}{\text{s}}$
\mathcal{D}	generic diagonal matrix
\mathcal{D}_y	diagonal robot system matrix
$\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$	unit vectors in wheel axis directions
$\mathbf{e}_{1'}, \mathbf{e}_{2'}, \mathbf{e}_{3'}$	unit vectors in wheel circumferential directions
$\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$	cartesian unit vectors in global coordinate system
$\mathbf{e}_\xi, \mathbf{e}_\eta, \mathbf{e}_\alpha$	cartesian unit vectors in robot coordinate system
E_{kin}	kinetic energy, J
f_z	auxiliary force in vertical direction, N
f_α	torque about α -axis in robot coordinates, Nm
f_ξ, f_η	forces in robot coordinates, N
\mathbf{f}	generalized forces vector in robot coordinates
$\tilde{\mathbf{f}}$	auxiliary forces vector in robot coordinates
F_x, F_y	forces in global coordinates, N

F_α	torque about α -axis in global coordinates, Nm
\mathbf{F}	generalized forces vector in global coordinates
\mathbf{F}_R	generalized robot forces vector in global coordinates
\vec{F}	two dimensional force vector in global coordinates
\vec{F}_B	two dimensional ball force vector in global coordinates
g	gravitational constant, $9.81 \frac{\text{m}}{\text{s}^2}$
G_1, G_2, G_3	wheel loads, N
\mathbf{G}	wheel load vector
\mathbf{G}_{dyn}	dynamic wheel load vector
$\mathbf{G}_{\text{shift}}$	shifting of wheel load vector due to dynamics
\mathbf{G}_{stat}	static wheel load vector
h_M	height of robot mass center, m
i_{gear}	gear ratio of propulsion system
I	electric current, A
k	spring stiffness, $\frac{\text{N}}{\text{m}}$
k_I	motor current controller gain
k_M	motor torque constant, $\frac{\text{Nm}}{\text{A}}$
k_n	motor speed controller gain
k_N	motor speed constant, $\frac{\text{rpm}}{\text{V}}$
k_{xy}	robot motion controller translational gain, $\frac{\text{m/s}}{\text{m}} = \text{s}^{-1}$
k_α	robot motion controller rotational gain, $\frac{\text{rad/s}}{\text{rad}} = \text{s}^{-1}$
m_B	mass of rolling ball as passive object, kg
m_C	robot chassis mass, kg
m_{disk}	mass of sliding disk as passive object, kg
m_R	robot total mass, kg
m_W	robot wheel mass, kg
M_B	generalized mass of passive object, kg
\mathcal{M}	robot mass matrix
$\mathcal{M}_{\text{glide}}$	robot mass matrix gliding contribution
$\mathcal{M}_{\text{roll}}$	robot mass matrix rolling contribution
n	angular speed, rpm
n_1, n_2, n_3	angular speeds at wheels, rpm
n_c	number of coefficients for trajectory shape (MCL method)
n_p	number of control points for trajectory
n_{WPA}	order of approximation polynomial (WPA method)
$n(O_k)$	number of instances of object class k
$n_{\text{pix}}(B)$	number of ball outline pixels
$n_{\text{pix}}(F)$	number of valid field marking pixels
\mathbf{n}	angular speed vector
\vec{n}	two dimensional vector in robot coordinates normal to line
O_x, O_y, O_c	object classes for field localization

O_k^l	instance l of object class k for field localization
\mathcal{O}	robot orientation matrix (global to robot velocities)
P	power, W
P_1, P_2, P_3	power at wheels, W
\mathbf{P}	wheel power vector
\mathcal{P}	robot system matrix (robot to global position)
r	trajectory independent variable (MCL method)
\mathbf{r}	vector containing the values of r_p (MCL method)
r_B	ball radius, m
\mathbf{r}_B	ball radius in digital image
r_p	value of r at control point p (MCL method)
r_R	distance robot center to wheel contact point on floor, m
r_W	wheel radius, m
R	ohmic resistance, Ω
R_{bat}	internal ohmic resistance of the battery, Ω
R_{mot}	effective ohmic resistance of the motor windings, Ω
R_{pk}	matrix with powers of r_p for trajectory shape (MCL method)
$R_\xi, R_\eta, R_{\bar{\alpha}}$	entries of Newton residual vector for field localization
\mathbf{R}	residual vector for Newton method for field localization
s	distance along trajectory, m
\mathbf{s}	distance along trajectory as function of r (MCL method)
\widehat{s}	arc length in CAT trajectory generation secant method, m
s^*	trajectory curve length (MCL method), m
S_B	sum of distances of all ball pixels to ball outline
S_F	sum of distances of all field marking pixels to associated object
t	time, s
t_1, t_2, t_3	wheel circumferential forces, N
t^*	trajectory arrival time (MCL method), s
\mathbf{t}	wheel circumferential force vector
\vec{t}	two dimensional vector in robot coordinates parallel to line
Tol	tolerance for numerical approximation algorithm
T	torque, Nm
T_1, T_2, T_3	wheel torques, Nm
\mathbf{T}	wheel torque vector
$\mathbf{T}_{\text{glide}}$	wheel torque vector for zero wheel moments of inertia
$\dot{u}_1, \dot{u}_2, \dot{u}_3$	wheel circumferential velocities, $\frac{\text{m}}{\text{s}}$
$\dot{\mathbf{u}}$	wheel circumferential velocities vector
U	voltage, V
\mathbf{U}_{PWM}	vector with voltages to three motor windings
\mathcal{U}	robot system matrix (wheel angular to circumferential velocities)
v_∞	trajectory speed final velocity (MCL method), $\frac{\text{m}}{\text{s}}$
v_{max}	maximum specified translational velocity, $\frac{\text{m}}{\text{s}}$

v_{\max}^2	squared velocity constraint by maximum velocity, $\frac{\text{m}^2}{\text{s}^2}$
v_{lat}^2	squared velocity constraint by lateral acceleration, $\frac{\text{m}^2}{\text{s}^2}$
w^*	objective function weight factor (MCL method)
\mathcal{W}	robot system matrix (horizontal force to wheel load shift)
$\widetilde{\mathcal{W}}$	auxiliary robot system matrix (force to wheel load shift)
x	first coordinate in global coordinate system, m
x_{B}	first ball coordinate in global coordinate system, m
x	x -coordinate of trajectory as function of r (MCL method)
x_{B}	horizontal ball coordinate in digital image
x_p	first global coordinate of control point p , m
x_{R}	first robot coordinate in global coordinate system, m
\mathbf{x}	generalized global coordinate vector
\mathbf{x}_{R}	generalized global robot coordinate vector
\mathbf{x}_{B}	solution vector for ball localization in digital image
\vec{x}	two dimensional position vector in global coordinates
\vec{x}_0	two dimensional ball neutral position in global coordinates
\vec{x}_{B}	two dimensional ball position vector in global coordinates
\vec{x}_{R}	two dimensional robot position vector in global coordinates
\vec{x}_{T}	two dimensional generic trajectory vector in global coordinates
$\Delta\vec{x}$	two dimensional ball dislocation in global coordinates
y	second coordinate in global coordinate system, m
y_{B}	second ball coordinate in global coordinate system, m
y_p	second global coordinate of control point p , m
y_{R}	second robot coordinate in global coordinate system, m
y	y -coordinate of trajectory as function of r (MCL method)
y_{B}	vertical ball coordinate in digital image
\mathcal{Y}	robot system matrix (robot to wheel angular coordinates)
z^*	objective function for trajectory shape (MCL method)

Greek Symbols

α	global robot orientation, rad
$\dot{\alpha}$	robot angular speed about vertical axis, $\frac{\text{rad}}{\text{s}}$
$\bar{\alpha}$	field orientation relative to robot, rad
β	angle of velocity vector, rad
γ	angle between wheel axis and roller axis, $^\circ$
Γ_{CAT}	parameter array containing ψ, δ, d_0 (CAT method)
Γ_{MCL}	parameter array containing ψ, δ (MCL method)

δ	ratio viscous damping:mass of passive object, s^{-1}
Δ_c^l	radius of object class c instance l , m
Δ_k^l	distance of object class $k=x,y$ instance l from origin, m
$\Delta_{c,\min}^l, \Delta_{c,\max}^l$	radial bounds of object class c instance l , m
$\Delta_{k,\min}^l, \Delta_{k,\max}^l$	normal bounds of object class $k=x,y$ instance l , m
$\epsilon_{k,\min}^l, \epsilon_{k,\max}^l$	tangential bounds of object class $k=x,y$ instance l , m
ϵ_v	non-uniformity of maximum velocities in different directions
ζ	angle between ball force and velocity (CAT method), rad
η	second coordinate in robot coordinate system, m
η_{CMP}	energetic efficiency of component CMP, %
η_{F}	second field coordinate in robot coordinate system, m
θ	motor rotor angle, rad
Θ	moment of inertia, kg m^2
$\Theta_{\text{ax,W}}$	wheel principal moment of inertia of its rotational axis, kg m^2
Θ_{B}	ball principal moment of inertia of rolling axis, kg m^2
$\Theta_{\alpha,\text{C}}$	robot chassis principal moment of inertia of α -axis, kg m^2
$\Theta_{\alpha,\text{W}}$	wheel principal moment of inertia of α -axis, kg m^2
κ	trajectory curvature (MCL method), m^{-1}
κ^*	trajectory curvature penalty (MCL method), m^{-2}
Λ	parameter array containing velocity and acceleration limits
μ	friction coefficient
μ_1, μ_2, μ_3	friction coefficients at wheels
μ_{max}	maximum specified friction coefficient
$\boldsymbol{\mu}$	friction coefficient vector for dynamic wheel load
ξ	first coordinate in robot coordinate system, m
ξ_0	ball neutral position in robot coordinate system, m
ξ_{F}	first field coordinate in robot coordinate system, m
$\boldsymbol{\xi}$	generalized robot coordinate vector
$\boldsymbol{\xi}_{\text{F}}$	generalized field coordinates in robot coordinate system
$\vec{\xi}$	two dimensional vector in robot coordinates
$\vec{\xi}_i$	two dimensional vector in robot coordinates for pixel i
$\vec{\xi}_{\text{F}}$	two dimensional vector in robot coordinates for field
$\rho(r)$	trajectory curve radius function (MCL method), m
$\hat{\rho}$	trajectory smallest curve radius (MCL method), m
τ	trajectory speed parameter (MCL method), s
$\dot{\varphi}_1, \dot{\varphi}_2, \dot{\varphi}_3$	wheel angular speeds, $\frac{\text{rad}}{\text{s}}$
$\dot{\boldsymbol{\varphi}}$	wheel angular speeds vector
χ_{B}	perspective parameter for ball localization
ψ	weight factor for robot position relative to generic trajectory
ω	angular speed of rolling ball, $\frac{\text{rad}}{\text{s}}$

Subscripts

123	wheel axis coordinate vector base
1'2'3'	wheel circumferential coordinate vector base
acc	acceleration, accelerating
avg	average
aux	auxiliary electric consumers
ax	axial
B	ball
bat	battery
<i>c</i>	object class: circle (field localization)
<i>c</i>	coefficients (trajectory shape)
C	chassis
CAT	related to the circle and tangent trajectory generation method
ctr	controller
cyc	drivecycle trajectory interpolated state
dec	decelerating
DC	directed current into DES
DES	measured value from DES
dyn	dynamic
EXT	extrapolated value from video image recognition
fric	friction
F	field
gear	gear box of propulsion system
glide	robot motion gliding contribution
<i>i</i>	counter index
ind	induced
is	actual physical value
I	integrator controller
<i>I</i>	motor current controller
<i>j</i>	counter for control points (index notation)
<i>k</i>	counter for trajectory shape coefficients (index notation)
kin	kinetic
<i>l</i>	counter for trajectory shape coefficients (index notation)
lat	lateral
max	maximum, upper limit
MCL	related to the minimal curve length trajectory generation method
min	minimum, lower limit
mot	motor

n	in normal direction
n	motor speed controller
n_{WPA}	polynomial order used in weighted polynomial approximation
p	counter for control points (index notation)
pix	pixel
P	proportional controller
PWM	pulse width modulated
q	counter for control points (index notation)
R	robot
ramp	input set speed for DES in speed regulation mode
res	resolution of digital variable
roll	robot motion wheel rolling contribution
s	sample (trajectory parameterization)
S	secant point in circle and tangent trajectory generation
set	controller set value
shift	shift due to dynamics
stat	static
t	in tangential direction
T	generic trajectory
tab	table of drivecycle trajectory data entries
v	velocity
VIR	video image recognition
W	wheel
WPA	related to the weighted polynomial approximation method
x	object class: line parallel to x -axis
$xy\alpha$	global coordinate vector base
y	object class: line parallel to y -axis
zzz	vertical coordinate vector base
$\xi\eta z$	auxiliary coordinate vector base (dynamic wheel loads)
$\xi\eta\alpha$	robot coordinate vector base

Superscripts

-1	matrix inverse, function inverse
(m)	Newton iteration counter
(o)	outer iteration counter
T	matrix transpose
$-T$	matrix inverse transpose

Operators

$\frac{d}{d\circ}(\square)$	total derivative of \square with respect to \circ
$\frac{\partial}{\partial\circ}(\square)$	partial derivative of \square with respect to \circ
$\dot{\square} = \frac{d\square}{dt}$	first derivative of \square with respect to time
$\ddot{\square} = \frac{d^2\square}{dt^2}$	second derivative of \square with respect to time
$\square' = \frac{d\square}{dr}$	first derivative of \square with respect to r
$\square'' = \frac{d^2\square}{dr^2}$	second derivative of \square with respect to r
$\circ \cdot \square$	vector dot product of \circ and \square
$\hat{\circ}$	maximum or minimum value of \circ
$\Delta\circ$	difference or deviation of \circ
$ \square $	absolute value of \square
$\delta\square$	variation of \square

Abbreviations

CAN	C ontroller A rea N etwork
CAT	C ircle A nd T angent trajectory planning method
CCD	C harge C oupled D evice
CPU	C entral P rocessing U nit
DC	D irect C urrent
DES	D igital E lectronically commutated S ervoamplifier
DMA	D irect M emory A ccess
FBAS	F arb- B ild- A ustast- S ynchron-Signal (Composite Video)
FIFA	F édération I nternationale de F ootball A ssociation

fps	f rames p er s econd
HSI	H ue S aturation I ntensity color model
MCL	M inimal C urve L ength trajectory planning method
MS-DOS	M icrosoft D isk O perating S ystem
PAL	P hase A lternation L ine
PC	P ersonal C omputer
PCI	P eripheral C omponent I nterconnect
PCMCIA	P ersonal C omputer M emory C ard I nternational A ssociation
PI	P roportional, I ntegrator controller architecture
PWM	P ulse W idth M odulated
qc	q uarter c ount
RAM	R andom A ccess M emory
RGB	R ed G reen B lue color model
RoboCup	The R obot W orld C up Soccer Games and Conferences
rpm	r otations p er m inute
UG	U nigraphics computer aided design application
VIR	V ideo I mage R ecognition
WPA	W eighted P olynomial A pproximation
YUV	Luminance (Y), Chromance (UV) color model

Motivation

This work was inspired by the RoboCup initiative. RoboCup is an international championship carried out between teams of autonomous mobile robots. Different tournaments are held every year in varying places around the world. RoboCup tournaments are divided into leagues differing in design restrictions and competition type. RoboCup is defined in [1]:

”RoboCup (Originally called as Robot World Cup Initiative) is an international research and education initiative. It is an attempt to foster artificial intelligence and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education.

For this purpose, RoboCup chose to use soccer game as a primary domain, and organizes RoboCup (The Robot World Cup Soccer Games and Conferences). In order for a robot team to actually perform a soccer game, various technologies must be incorporated including: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup is a task for a team of multiple fast-moving robots under a dynamic environment.”

The RoboCup rule set is based on the FIFA laws [2], [3]; RoboCup applies changes with respect to both abilities and restrictions of non-human players. The ultimate goal of the RoboCup events is for robots to win against the human soccer world champion team by the year 2050. Until then, RoboCup serves for comparison of the scientific progress between different teams in the areas of robotics and artificial intelligence. Refer to the official internet homepage [4] for more detailed information on RoboCup in general, different leagues, and rule sets.

The Middle Size League created the greatest challenge in 1999, it was also referred to as royal league, because it requires distributed intelligence incorporated in fully autonomous robots. This is the major difference to the Small

Size League where miniature robots are controlled by a central intelligence that uses a global camera above the field to survey the complete miniature soccer field. All subsequent considerations as well as the robot developed in the course of this work address the Middle Size or F-2000 League. The multitude of teams participating in the RoboCup competition are formed within or across computer science departments of universities and research institutes. Their focus is primarily the development and implementation of artificial intelligence and intelligent multi-agent systems. In 1999, very few teams cared about developing their individual robot hardware, typically some commercially available robot platform was purchased and equipped with a computer that runs the control software. Some of the commercial platforms are already equipped with ultrasonic distance sensors that can be used for obstacle avoidance at low velocities. Two very popular platforms are the Pioneer [5] and the Super Scout [6], both have a differential drive system. Video image recognition is a basic skill because colors play a vital role on the RoboCup soccer field. A multitude of participating teams also employed commercially available video processing hardware and software. An analysis of about ten hours of video material recorded at the RoboCup competitions in 1998 and 1999 allowed the following disappointing conclusions:

- The soccer matches were sluggish and extremely boring to watch
- Many robots were standing on the field and did not move at all
- Many robots did not react to the ball even if it was lying in their direct neighborhood
- It often happened that there was a cluster of more than four robots around the ball and nobody could move any more
- The robots were slow and not capable of dribbling the ball for an extended period of time; extremely few robots managed to dribble a curve
- The goalkeeper often left its goal sideways in the same moment an opposing robot was approaching to shoot a goal
- Robots drove into the walls surrounding the field and persisted driving in that direction which was not possible
- A team won a match without any robot from that team moving at all, because the opposing team managed to shoot an own goal

Some of the described malbehavior can be explained with insufficient communication between the field players, but most of the observations can only be explained with one reason:

The robots were lacking substantial basic skills!

Two basic skills were identified: First, the capability of acquiring a detailed view of the surrounding world. This includes the own position on the soccer field, the position of the ball, and the position of all other soccer players, predominantly opponents. Some teams reported that their video image recognition searched for a large blue blob in the camera's video images and assumed that this must be the blue goal. This is contrary to the strategy pursued in the development of the robot platform presented in this thesis, which introduces a global coordinate system and expresses the position of all mobile objects with these coordinates. Second, it is crucial to supply the robot with the agility and the skills to handle a rolling soccer ball and maneuver it to a desired position. The absence of this basic skill renders the most intelligent control algorithm useless. The most sophisticated multi-agent collaboration approach is not capable of shooting a goal if the basic skills vision and motion do not allow it. It was therefore decided to fill this gap and develop a robot platform that can both get its bearings and provides the required agility to maneuver. The scope of the development of the robot platform presented in this work was outlined as follows:

Develop a mobile robot that is capable of solving the **Two Body Problem** for a rolling ball on a given trajectory.

Definition: **Two Body Problem**

Based on the physical equations of motion, control the two dimensional planar trajectory of a rigid body with a mobile robot by solely pushing the rigid body along the line connecting the centers of gravity.

If the ball must be pushed around a curve, the robot needs to exert centripetal forces on the ball. The RoboCup competition rules forbid the robot to reach around the ball and hold it. The rules demand that the robot must not cover more than 30% of the ball's diameter. In order to dribble a curve without violating the RoboCup regulations, it is therefore necessary that the robot moves on a trajectory with a larger curve radius, facing inward to address the centripetal forces on the ball. If the robot has a differential drive system and simply drives around a curve, the ball will maintain its direction and get lost towards the outside of the curve. It therefore seems compelling that the robot propulsion system addresses all degrees of freedom individually, that is two translational and one rotational degree of freedom in a two dimensional plane. This directly calls for an omnidirectional drive system in order to decouple the translational and rotational degrees of freedom. In addition, it requires three independent actuators to address the three degrees of freedom. Regular

wheels can not accomplish this requirement, the wheels of an omnidirectional drive system must have an active and a passive direction of motion. This can be solved with little rollers distributed around wheel circumference, they can transmit full circumferential forces with very little friction in axial direction of the omnidirectional wheel.

As far as the sensor system is concerned, it was decided to allow no compromises and to use color camera vision as the only sensor. In 1999, it was already announced that the boundary walls would be removed in a few years, therefore any development of a sensor relying on reflection had no future. It was furthermore decided in the course of this work to introduce a global coordinate system to express the positions of the various mobile objects quantitatively rather than the qualitative 'follow the blob' approach.

The drawbacks caused by the robot hardware during the RoboCup competitions were very extreme in the year 1999 when the research project presented in this thesis was initiated. In the meantime, by the year 2004, the situation has shifted a bit. More teams are realizing that the robot platform plays a significant role, and they are beginning to develop platforms that are suited to their individual needs. This can be concluded from the following statistics, comparing the RoboCup competition held 1999 in Stockholm to the competition 2004 in Lisbon. For the year 2004, a very comprehensive overview of the technical data of the hardware components is available [28]. The data was obtained with a detailed questionnaire that was distributed to the participating teams prior to the competition. Additional information can be obtained from the team description papers [29] to [42]. The data for 1999 could only be obtained from the team description papers [8] to [27], and the content of these papers was not subjected to any template. Therefore, every team had its individual focus, some teams did not mention their robot hardware at all.

In Stockholm 1999, 20 teams participated in the Middle Size League, in Lisbon in 2004 the number increased to 24. Out of these, only 10 teams participated in both competitions. In the following, the data provided by the participating teams for these two years is compared with respect to the development of the propulsion systems, the maximum velocities, the sensors, and the accuracy of the obtained positions.

Propulsion Systems: In 1999, 5 teams reported that they used the Pioneer platform, at least 2 used Nomads, and in 4 cases the platform was developed by the team. There was one reported omnidirectional approach compared to at least 14 non-omnidirectional propulsion systems, most of them being differential drives. This relationship changed significantly in 2004 when robots in 15 teams were equipped with omnidirectional drive compared to 11 teams with pure differential drive, note that some teams participated with mixed

robot teams. This development underlines the analysis that the robot agility plays an important role, and the conclusion that an omnidirectional platform is the best option to achieve that result. The winning team in 1999 'CS-Sharif' started with a custom developed robot platform that incorporated aspects of both omnidirectional and differential drive behavior. It had two active front wheels and a castor wheel in the rear, each front wheel could be rotated about a vertical axis individually. This allowed the robot to perform its two primary maneuvers, driving forward and turning around the ball located in front of the robot, keeping it in the vision field of the camera permanently. These two maneuvers were executed one after the other, creating a very static robot behavior. Note that CS-Sharif was one of the few teams with mechanical engineers.

Velocities: For the year 1999, very little information is available from the team description papers. One team reported a maximum velocity below 1 m/s, 3 teams reported velocities between 1 and 2 m/s, and 2 teams ranged between 2 and 4 m/s. The maximum velocity of the robots participating in 2004 was distributed as follows: 13 robots had a maximum velocity between 1 and 2 m/s, 9 robots were able to drive at a maximum velocity between 2 and 4 m/s, while 1 team reported top velocities above 4 m/s. The motor power was smaller than 100 W in 16 teams, larger than 400 W in 1 team, and in between in 4 teams.

Sensors: Sensors for orientation on the field are difficult to compare because in 1999, the soccer field was surrounded by walls. This produced perfect conditions for reflection based sensors like laser scanners and ultrasonic sensors for self localization. These walls did not exist in the 2004 competition any more, only four corner posts and the goals remained. Therefore, video image recognition was used as primary sensor in 2004, laser scanners were only used by 5 teams as support system. Omnidirectional vision was reported by 21 teams, most of them use a single camera at the top of the robot. This camera faces upwards into a convex mirror, allowing a distorted wide angle view of the complete field and beyond. None of the teams used stereo vision in 2004.

Position Accuracy: Camera sampling rates ranged from 15 fps (frames per second) to 60 fps, but the majority consisting of 13 teams reported 30 fps. This rate is often not achieved for processing the video images by the computer. 11 teams reported processing rates of 3 to 20 fps, while 10 teams managed 20 to 30 fps. Two teams asserted to achieve higher processing rates. The resulting precision was also interrogated in the 2004 questionnaire. 15 teams claimed a precision of self localization of 1 to 30 cm, 3 teams ranged from 30 to 50 cm, and another 3 were worse than 50 cm. Note that 4 teams could not specify their precision, probably they did not compute their position on

the field but followed the blue blob. Five teams reported that they have a dynamic physical model of their robot system used for simulation.

The research project presented in this thesis focussed on three aspects required for successful participation in a RoboCup competition: First, a mobile robot chassis was developed that provides the agility and velocity to maneuver in the RoboCup environment. Second, a trajectory planning method associated with a global coordinate system was developed that is adapted to both dribbling and obstacle avoidance requirements during a competition. Third, fast and efficient video image recognition algorithms for self localization and ball detection were developed. The following targets were defined for the above aspects: First, it was the goal to equip the robot with an omnidirectional propulsion system guaranteeing optimal agility. The target values for the robot's maximum velocity and acceleration were defined as 5 m/s and 5 m/s², respectively. It was the goal to operate the robot at the limits defined by the laws of physics, not by weak actuators. These limits are friction between wheels and floor as well as the height of the robot's mass center causing it to fall over at too high accelerations. Second, the targets for the trajectory planning method were simplicity, robustness, little computational effort, and the intrinsic ability to avoid obstacles. Third, video image recognition was aiming at real time processing at 25 fps. This is the frame rate of standard PAL cameras. Self localization and ball detection were defined as required outputs of the sensor system. Detection of opposing robots was not considered because it was not necessary at this stage. It was the goal to obtain a resulting precision of the robot self localization in the range of 1 cm. The target for the error in the ball position was set to 5% of the ball's distance from the robot.

This thesis is structured according to the elements of a mechatronic system illustrated in figure 1. First, the open loop forward branch containing the elements *trajectory planning*, *actuators*, and *robot system* is discussed in reverse order. Thereafter, the design of the feedback loop consisting of the elements *sensors* and *controllers* is explained. The *experimental validation* finally proves the successful behavior of the robot software and hardware.

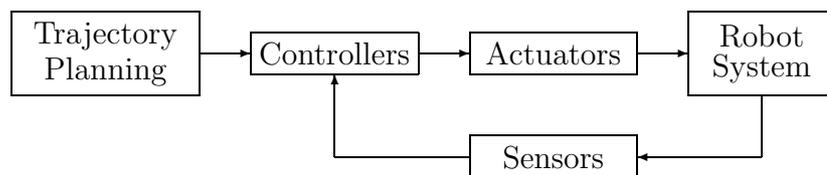


Figure 1: Robot as mechatronic system

Figure 2 shows a photograph of an intermediate version of the soccer robot with ball on a slalom course.



Figure 2: Robot developed in this thesis

Chapter 1

Robot System

This chapter provides a mathematical description of the robot system. It elaborates basic design considerations by evaluating different setups and variants of the omnidirectional propulsion system. Thereafter the kinematics, kinetics, and equation of motion for the design of choice are derived in detail.

1.1 Design Considerations

The considerations made in the introduction suggest an omnidirectional propulsion system for the mobile robot. This means that the three degrees of freedom, two translational and one rotational, are decoupled. All following investigations will therefore focus on omnidirectional motion.

Omnidirectional propulsion of a wheeled vehicle requires some specific considerations. Most importantly, each wheel must be designed for omnidirectional motion. In addition to the circumferential direction in which active forces can be applied, a second direction of motion must be defined for an omnidirectional wheel. This second direction must be linearly independent of the circumferential direction and it may be passive because it does not need to be capable of transmitting forces.

A possible way to incorporate a passive direction in a wheel is distributing a set of small rollers around the wheel's circumference. It is favorable to design the rollers in such a way that their envelope is located on a cylindrical surface defining the wheel's circumference. The angle γ between the circumferential wheel direction and a roller axis that is parallel to the ground primarily defines the wheel design. Figures 1.1 to 1.3 show three different omnidirectional wheels that were built and tested to compare their performance.



Figure 1.1: Omnidirectional wheel, single roller type

Figure 1.1 illustrates a wheel with one row of rollers, it is referred to as single roller wheel type. Eight rollers are distributed around the wheel's circumference. Their shape was chosen such that their envelope forms a circle. Each gap between two rollers creates a secant to that circle, the maximum deviation between that secant and the circle is smaller than 0.5 mm. It was expected that these gaps would induce vibrations into the complete robot system while the robot is driving. There was hope that these vibrations would not influence the robot's driving performance significantly. For this wheel design, the angle γ is zero.

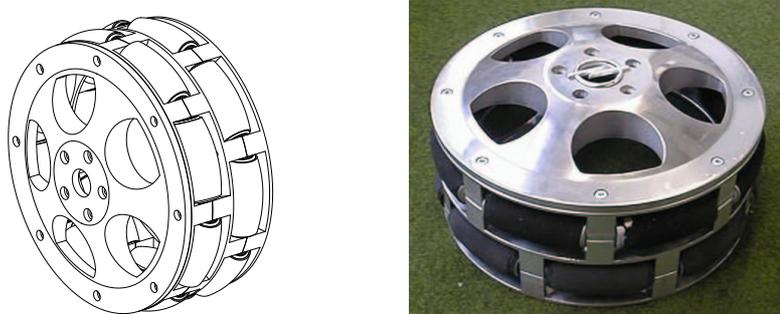


Figure 1.2: Omnidirectional wheel, double roller type

Figure 1.2 shows the so called double roller wheel type. A second set of eight rollers was added. Each roller of the second row covers a gap between two rollers of the first row. This was done to reduce the vibrations induced by the gaps between the rollers. The expected disadvantage of this design is that the distance of the wheel's contact point with the floor to the robot center is no longer uniquely defined by the geometry. It changes by the distance of the two rows of rollers depending on which roller is in contact with the floor. If the robot rotates about its vertical axis, then the wheel must spin with

different speeds, depending on which roller is in contact with the floor. The double roller wheel type is also described by $\gamma=0$.



Figure 1.3: Omnidirectional wheel, cutting edge type

Figure 1.3 shows a third wheel type, the cutting edge design. It was developed to reduce complexity, weight, and cost of the omnidirectional wheels. It was designed without rollers, instead the wheel has a large number of sharp edges around its circumference. These edges are supposed to cut into the necessary soft floor in order to create a preferred direction of passive slip along the edge. In addition to that, the edges are slanted at an angle $\gamma=30^\circ$ to reduce wheel induced vibrations. It was the goal to allow the next cutting edge to contact the floor before the previous one has lost its contact to it. This is similar to the helical tooth system for gear wheels. Two disadvantages were expected: A significantly increased friction between wheel and floor, and the undefined geometry of the contact point between wheel and floor.

It is also conceivable to design an omnidirectional wheel with $\gamma>0$ with rollers. This would combine the advantages of the single roller with the cutting edge type wheel. This concept was dropped at an early stage because it can be directly compared to the double roller concept as follows: First, it has the same disadvantage of an undefined contact point between wheel and floor. Second, it is much more complicated to manufacture. And third, the width of the wheel would not be smaller than the width of the double roller wheel if all gaps between the rollers are to be closed. Therefore, it was concluded that the double roller wheel design has more advantages and no disadvantages compared to a $\gamma>0$ wheel design with rollers.

In the following, the number and arrangement of the wheels is discussed. A minimum of three wheels is required because motion in two dimensions offers three degrees of freedom, two translational and one rotational. A four wheeled omnidirectional propulsion is also possible but it clearly implies the disadvantage of a kinetically over determined system. Each design will have

specific directions of motion that are different in their kinetic and kinematic behavior. The following investigation assumes that the maximum angular speed of each wheel is limited by its driving motor. Consequently, the maximum translational robot velocity depends on the driving direction, there are fast and slow directions. The ratio ϵ_v of greatest to smallest maximum robot velocity is defined in equation (1.1). It serves as a measure for the non-uniformity of motion in different directions:

$$\epsilon_v \stackrel{def}{=} \frac{v_{\max}}{v_{\min}} \quad (1.1)$$

It is defined under the following conditions:

- The robot undergoes a pure translational motion, there is no rotation about its vertical axis
- All wheels have the same radius
- Each wheel's angular speed is restricted by an upper limit

Values of $\epsilon_v \rightarrow 1$ indicate better designs because the robot can drive equally fast in any direction. Values of $\epsilon_v \rightarrow \infty$ indicate worse designs because the robot has certain directions in which the maximum translational velocity is much higher than in other directions. Figure 1.4 shows four different robot propulsion designs. The solid rectangles indicate the wheel positions. The dash-dotted lines represent the wheels' axles and the axles of the respective rollers that are currently in contact with and parallel to the floor.

The four wheel configuration in figure 1.4(a) is characterized by the value $\epsilon_v = \cos(\gamma)^{-1}$ for $\gamma \leq 45^\circ$ and $\epsilon_v = \sin(\gamma)^{-1}$ for $\gamma \geq 45^\circ$. The optimal angle γ would be 45° for which a minimum of $\epsilon_v = \sqrt{2}$ is obtained. This configuration has the advantage of a preferred translational direction (up/down in the figure) for which the rollers on every wheel stand still and thus do not suffer any wear.

The configuration in figure 1.4(b) is a modification of the four wheeled one, ridding it of its kinematic disadvantage. It does not have a preferred translational direction like the four wheel configuration, but for one direction (up/down in the figure) only one roller of the lower wheel is in motion. For a symmetric setup, three different wheel types are required, they are $\gamma_1 < 0$, $\gamma_2 > 0$ and $\gamma_3 = 0$. Again the minimum of ϵ_v is $\sqrt{2}$.

The three wheeled configurations in figures 1.4(c) and 1.4(d) have their wheel axles mutually oriented at angles of 120° . The angle γ does not affect the non-uniformity $\epsilon_v = 2/\sqrt{3}$, it only determines the fixed transmission ratio from wheel angular speed to robot velocity. There is no preferred direction as in the four wheel configuration. However, there are three directions

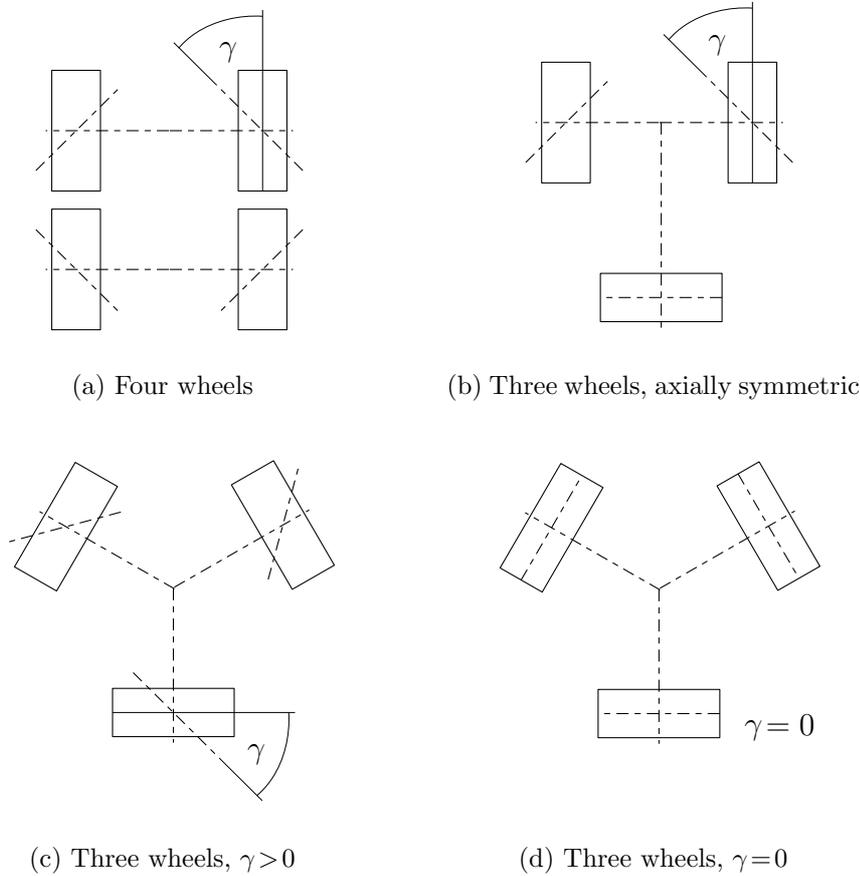


Figure 1.4: Four different wheel and roller configurations

(perpendicular to the wheel axes) for which the rollers of one wheel are not in motion. Both configurations in figures 1.4(c) and 1.4(d) were previously patented for an omnidirectional vehicle in an amusement park [45]. In the scope of this work, they were built and tested with respective wheel designs. Refer to figures 1.1 and 1.2 for two wheel designs with $\gamma = 0$, and to figure 1.3 for a wheel design with $\gamma > 0$.

The preferred wheel and roller configuration is the three wheeled one with $\gamma = 0$ according to figure 1.4(d) for the following reasons:

- There is no preferred direction and the robot is not designed for long or heavy duty operation; there is no need for a redundant fourth wheel; therefore, the four wheeled configuration is not reasonable
- The configuration in figure 1.4(b) requires three different wheel designs, this is not desirable from a manufacturing point of view

- Both configurations in figures 1.4(c) and 1.4(d) have a minimal value of non-uniformity ϵ_v
- The roller gaps in the $\gamma=0$ design produce a deviation in the wheel radius of less than 0.5 %; the bumps produced by the gaps during driving are considered minor
- Implementing double roller wheels with $\gamma=0$ is considered a backup solution because of the undefined distance from robot center to wheel and because of the increased mass and size
- The $\gamma>0$ wheel design also lacks a defined distance from robot center to wheel; it is more difficult to manufacture than the $\gamma=0$ design; it will also result in a higher wheel mass

1.2 Mathematical Description

1.2.1 Kinematics

This section describes the kinematics of the omnidirectional propulsion system. The main focus lies in the connection between wheels' angular speeds and robot velocity.

Figure 1.5 illustrates a top view of the robot for the $\gamma=0$ design, the three solid boxes represent the wheels. The common radius of all wheels is r_W , and the distance from robot center to the contact points between wheels and floor is r_R . Four sets of coordinates are introduced: The mutually perpendicular unit vectors $\{\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z\}$ are fixed in a global non-moving reference frame with \mathbf{e}_x and \mathbf{e}_y being parallel to the floor and \mathbf{e}_z facing upwards. The unit vectors $\{\mathbf{e}_\xi, \mathbf{e}_\eta, \mathbf{e}_\alpha\}$ are also mutually perpendicular. They are fixed on the robot, \mathbf{e}_ξ denotes the robot *forward* direction and \mathbf{e}_α is parallel to \mathbf{e}_z . Note that \mathbf{e}_z represents a translational dimension whereas \mathbf{e}_α is used to express rotational quantities. The third set of coordinates consists of the unit vectors $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, each of them points in the respective wheel's axial direction. The unit vectors $\{\mathbf{e}_{1'}, \mathbf{e}_{2'}, \mathbf{e}_{3'}\}$ are pointing in the wheel circumferential directions. They are linearly dependent on \mathbf{e}_ξ and \mathbf{e}_η and compute as vector cross products:

$$\mathbf{e}_{i'} = \mathbf{e}_\alpha \times \mathbf{e}_i \quad , \quad i = 1, 2, 3 \quad (1.2)$$

The origin of the ξ - η coordinate system is located in the robot's geometrical center where the wheel axes intersect. The robot orientation α is defined as the angle between \mathbf{e}_x and \mathbf{e}_ξ .

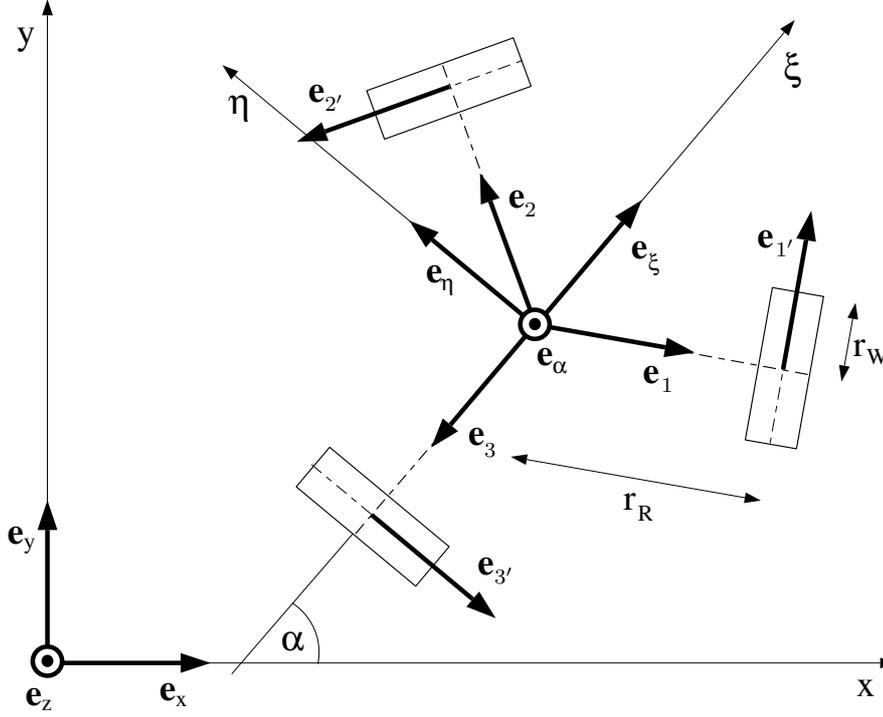


Figure 1.5: Coordinate systems definitions

Utilizing these coordinate systems, the robot's two translational and one rotational velocities can be expressed in form of generalized velocity vectors:

$$\dot{\mathbf{x}} \stackrel{def}{=} \dot{x} \mathbf{e}_x + \dot{y} \mathbf{e}_y + \dot{\alpha} \mathbf{e}_\alpha = [\dot{x} \ \dot{y} \ \dot{\alpha}]_{xy\alpha}^T \quad (1.3)$$

$$\dot{\boldsymbol{\xi}} \stackrel{def}{=} \dot{\xi} \mathbf{e}_\xi + \dot{\eta} \mathbf{e}_\eta + \dot{\alpha} \mathbf{e}_\alpha = [\dot{\xi} \ \dot{\eta} \ \dot{\alpha}]_{\xi\eta\alpha}^T \quad (1.4)$$

$$\dot{\boldsymbol{\varphi}} \stackrel{def}{=} \dot{\varphi}_1 \mathbf{e}_1 + \dot{\varphi}_2 \mathbf{e}_2 + \dot{\varphi}_3 \mathbf{e}_3 = [\dot{\varphi}_1 \ \dot{\varphi}_2 \ \dot{\varphi}_3]_{123}^T \quad (1.5)$$

$$\dot{\mathbf{u}} \stackrel{def}{=} \dot{u}_1 \mathbf{e}_{1'} + \dot{u}_2 \mathbf{e}_{2'} + \dot{u}_3 \mathbf{e}_{3'} = [\dot{u}_1 \ \dot{u}_2 \ \dot{u}_3]_{1'2'3'}^T \quad (1.6)$$

The robot velocity is expressed by $\dot{\mathbf{x}}_R$ in global coordinates and by $\dot{\boldsymbol{\xi}}_R$ in robot coordinates. Note that $\dot{\xi}$ and $\dot{\eta}$ do not represent a motion of the robot relative to the robot coordinate system. They are natural coordinates, expressing the motion of the robot coordinate system in terms of its own coordinates. The angular and circumferential velocities of the wheels are expressed by $\dot{\boldsymbol{\varphi}}$ and $\dot{\mathbf{u}}$, respectively. The indices on the right hand side of equations (1.3) to (1.6) denote the set of coordinate vectors which was used for the vector formulation. Angular speeds can also be expressed in units of

rotations per minute:

$$\mathbf{n} \stackrel{def}{=} \frac{60 \text{ rpm}}{2\pi \frac{\text{rad}}{\text{s}}} \dot{\boldsymbol{\varphi}} = [n_1 \ n_2 \ n_3]_{123}^T \quad (1.7)$$

The following transformations connect the kinematics expressed in the different coordinate systems:

$$\dot{\boldsymbol{\xi}}_R = \mathcal{O} \dot{\mathbf{x}}_R \quad (1.8)$$

$$\dot{\boldsymbol{\varphi}} = \mathcal{Y} \dot{\boldsymbol{\xi}}_R \quad (1.9)$$

$$\dot{\mathbf{u}} = \mathcal{U} \dot{\boldsymbol{\varphi}} \quad (1.10)$$

Herein the matrix \mathcal{O} transforms global coordinates into robot coordinates, the matrix \mathcal{Y} transforms robot coordinates into wheel angular speeds, and the matrix \mathcal{U} transforms wheel angular speeds into wheel circumferential velocities. According to the aforementioned convention, the coordinate bases in the following equations are subscripted to their left and right hand side. The orientation matrix \mathcal{O} is a function of the actual robot orientation α , the system matrices \mathcal{Y} and \mathcal{U} only depend on the kinematic parameters r_R , r_W and γ .

$$\mathcal{O}(\alpha) \stackrel{def}{=} \left[\begin{array}{ccc} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{array} \right]_{\xi\eta\alpha}^{xy\alpha} \quad (1.11)$$

$$\mathcal{U} \stackrel{def}{=} r_W \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array} \right]_{1'2'3'}^{123} \quad (1.12)$$

The system matrix \mathcal{Y} can be assembled from two contributions, the matrix \mathcal{Y}_0 accounting for the arrangement of the wheel axles, and the matrix \mathcal{Y}_γ accounting for the orientation of the roller axles. Both matrices can be derived from simple kinematic considerations. Comparing a $\gamma > 0$ wheel design according to figure 1.4(c) with a $\gamma = 0$ wheel design according to figure 1.4(d) for given wheel angular speeds, the matrix \mathcal{Y}_γ tunes the propulsion system in two ways: It reduces the translational velocity by a factor $\cos(\gamma)$ and it twists the translational direction by the angle γ . It does not affect rotation about the α -axis because in that case the rollers are not active. Note that \mathcal{Y}_γ becomes neutral for $\gamma = 0$.

$$\mathcal{Y} \stackrel{def}{=} \mathcal{Y}_0 \mathcal{Y}_\gamma \quad (1.13)$$

$$\mathcal{Y}_0 \stackrel{def}{=} \frac{-1}{r_W} \underset{123}{\begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & r_R \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & r_R \\ 0 & -1 & r_R \end{bmatrix}}_{\xi\eta\alpha} \quad (1.14)$$

$$\mathcal{Y}_\gamma \stackrel{def}{=} \underset{\xi\eta\alpha}{\begin{bmatrix} 1 & -\tan(\gamma) & 0 \\ \tan(\gamma) & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\xi\eta\alpha} \quad (1.15)$$

$$\mathcal{Y}_\gamma^{-1} = \underset{\xi\eta\alpha}{\cos(\gamma) \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\xi\eta\alpha} \quad (1.16)$$

There are some useful identities and simplifications in the system matrices. For the following considerations, let \mathcal{D} be a generic 3 by 3 diagonal matrix:

$$\mathcal{D} \stackrel{def}{=} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} \quad (1.17)$$

The orientation matrix \mathcal{O} is a simple rotation matrix for which the following equations apply:

$$\mathcal{O}^{-1} = \mathcal{O}^T \quad (1.18)$$

$$\mathcal{O}^T \mathcal{D} \mathcal{O} = \mathcal{D} \quad \forall \quad d_1 = d_2 \quad (1.19)$$

$$\mathcal{D} \mathcal{O} = \mathcal{O} \mathcal{D} \quad \forall \quad d_1 = d_2 \quad (1.20)$$

The matrix $\mathcal{Y}_0^T \mathcal{Y}_0$ is diagonal and allows an easy calculation of \mathcal{Y}_0^{-1} :

$$\mathcal{D}_y \stackrel{def}{=} \mathcal{Y}_0^T \mathcal{Y}_0 = \underset{\xi\eta\alpha}{\begin{bmatrix} \frac{3}{2} & 0 & 0 \\ 0 & \frac{3}{2} & 0 \\ 0 & 0 & 3r_R^2 \end{bmatrix}}_{\xi\eta\alpha} \quad (1.21)$$

$$\mathcal{Y}_0^{-1} = \mathcal{D}_y^{-1} \mathcal{Y}_0^T = \underset{\xi\eta\alpha}{\frac{-r_W}{3} \begin{bmatrix} \sqrt{3} & -\sqrt{3} & 0 \\ 1 & 1 & -2 \\ r_R^{-1} & r_R^{-1} & r_R^{-1} \end{bmatrix}}_{123} \quad (1.22)$$

1.2.2 Kinetics

The definitions and transformations of the system's kinetic behavior are equivalent to the kinematic description. The generalized forces in the respective coordinate directions are defined as follows:

$$\mathbf{F} \stackrel{def}{=} F_x \mathbf{e}_x + F_y \mathbf{e}_y + F_\alpha \mathbf{e}_\alpha = [F_x \ F_y \ F_\alpha]_{xy\alpha}^T \quad (1.23)$$

$$\mathbf{f} \stackrel{def}{=} f_\xi \mathbf{e}_\xi + f_\eta \mathbf{e}_\eta + f_\alpha \mathbf{e}_\alpha = [f_\xi \ f_\eta \ f_\alpha]_{\xi\eta\alpha}^T \quad (1.24)$$

$$\mathbf{T} \stackrel{def}{=} T_1 \mathbf{e}_1 + T_2 \mathbf{e}_2 + T_3 \mathbf{e}_3 = [T_1 \ T_2 \ T_3]_{123}^T \quad (1.25)$$

$$\mathbf{t} \stackrel{def}{=} t_1 \mathbf{e}_{1'} + t_2 \mathbf{e}_{2'} + t_3 \mathbf{e}_{3'} = [t_1 \ t_2 \ t_3]_{1'2'3'}^T \quad (1.26)$$

The variables \mathbf{F}_R and \mathbf{f}_R represent the force applied to the robot in global and robot coordinates, respectively. The vectors \mathbf{T} and \mathbf{t} contain the wheel torques and circumferential forces, respectively. In a mechanical sense, the variables F_x , F_y , f_ξ , f_η , t_1 , t_2 , and t_3 represent forces, whereas F_α , f_α , T_1 , T_2 , and T_3 represent torques. The generalized forces are transformed from one coordinate base to another as follows:

$$\mathbf{F}_R = \mathcal{O}^T \mathbf{f}_R \quad (1.27)$$

$$\mathbf{f}_R = \mathcal{Y}^T \mathbf{T} \quad (1.28)$$

$$\mathbf{T} = \mathcal{U}^T \mathbf{t} \quad (1.29)$$

Equations (1.27) to (1.29) follow directly out of equations (1.8) to (1.10) by expressing the total power introduced into the robot system for an arbitrary robot state in terms of the four coordinate systems:

$$P = \mathbf{F}_R^T \dot{\mathbf{x}}_R = \mathbf{f}_R^T \dot{\boldsymbol{\xi}}_R = \mathbf{T}^T \dot{\boldsymbol{\varphi}} = \mathbf{t}^T \dot{\mathbf{u}} \quad (1.30)$$

Additionally, a generalized power vector is introduced that contains the powers at each wheel. These are computed with the respective wheel torque and angular speed:

$$\mathbf{P} \stackrel{def}{=} [P_1 \ P_2 \ P_3]^T \quad (1.31)$$

$$P_i = T_i \dot{\varphi}_i \quad , \quad i = 1, 2, 3 \quad (1.32)$$

The wheel loads are the vertical forces acting between floor and wheels:

$$\mathbf{G} \stackrel{def}{=} G_1 \mathbf{e}_z + G_2 \mathbf{e}_z + G_3 \mathbf{e}_z = [G_1 \ G_2 \ G_3]_{zzz}^T \quad (1.33)$$

The dynamic wheel loads can be divided into static loads and shifted loads during acceleration phases:

$$\mathbf{G}_{\text{dyn}} = \mathbf{G}_{\text{stat}} + \mathbf{G}_{\text{shift}} \quad (1.34)$$

$$\mathbf{G}_{\text{stat}} = \frac{1}{3} m_R g [1 \ 1 \ 1]_{zzz}^T \quad (1.35)$$

$$\mathbf{G}_{\text{shift}} = -\mathcal{W}^{-1} \mathbf{f}_R \quad (1.36)$$

The wheel loads shift by the amount $\mathbf{G}_{\text{shift}}$ if a horizontal force \mathbf{f} is applied to the center of gravity located on the α -axis at the height h_M above the ground. In equation (1.36), this force is the reaction force created by the robot's inertia during acceleration, $\mathbf{f} = -\mathbf{f}_R$. The force that causes the acceleration acts on the wheel contact points with the floor. The matrix \mathcal{W} represents the robot geometry, it translates the horizontal force \mathbf{f} to the wheel loads shift $\mathbf{G}_{\text{shift}}$. Both \mathcal{W} and its pseudo-inverse \mathcal{W}^{-1} are singular because a torque about the α -axis does not shift wheel loads and vice versa. For calculation of the matrices \mathcal{W} and \mathcal{W}^{-1} , the mechanical problem needs to be formulated including a vertical component in the force acting on the center of gravity. Therefore, the auxiliary force vector $\tilde{\mathbf{f}}$ and the auxiliary system matrix $\tilde{\mathcal{W}}$ are introduced:

$$\tilde{\mathbf{f}} \stackrel{\text{def}}{=} [f_\xi \ f_\eta \ f_z]_{\xi\eta z}^T \quad (1.37)$$

The mechanical equilibrium conditions of the equation $\mathbf{G}_{\text{shift}} = \tilde{\mathcal{W}}^{-1} \tilde{\mathbf{f}}$ yield the following nonsingular matrix:

$$\tilde{\mathcal{W}} \stackrel{\text{def}}{=} \frac{r_R}{2h_M} \begin{bmatrix} 1 & 1 & -2 \\ -\sqrt{3} & \sqrt{3} & 0 \\ \frac{2h_M}{r_R} & \frac{2h_M}{r_R} & \frac{2h_M}{r_R} \end{bmatrix}_{\xi\eta z} \quad (1.38)$$

$$\tilde{\mathcal{W}}^{-1} = \frac{h_M}{3r_R} \begin{bmatrix} 1 & -\sqrt{3} & -\frac{r_R}{h_M} \\ 1 & \sqrt{3} & -\frac{r_R}{h_M} \\ -2 & 0 & -\frac{r_R}{h_M} \end{bmatrix}_{\xi\eta z} \quad (1.39)$$

Eliminating the last row of $\tilde{\mathcal{W}}$ and the last column of $\tilde{\mathcal{W}}^{-1}$ yields the matrices \mathcal{W} and \mathcal{W}^{-1} , respectively, because they are related to the inert orientation α . The rows of \mathcal{W} and the corresponding columns of \mathcal{W}^{-1} add up to zero, because the entries of $\mathbf{G}_{\text{shift}}$ always add up to zero:

$$\mathcal{W} \stackrel{\text{def}}{=} \frac{r_R}{2h_M} \begin{bmatrix} 1 & 1 & -2 \\ -\sqrt{3} & \sqrt{3} & 0 \\ 0 & 0 & 0 \end{bmatrix}_{\xi\eta\alpha} \quad (1.40)$$

$$\mathcal{W}^{-1} \stackrel{\text{def}}{=} \frac{h_M}{3r_R} \begin{bmatrix} 1 & -\sqrt{3} & 0 \\ 1 & \sqrt{3} & 0 \\ -2 & 0 & 0 \end{bmatrix}_{\xi\eta\alpha} \quad (1.41)$$

1.2.3 Equation of Motion

After having defined the kinematic and kinetic properties, the equation of motion can now be derived. Figure 1.6 shows the masses and moments of inertia contained in the robot. The robot is modelled as a set of four distinct rigid bodies. The robot chassis is illustrated as a gray circle. Its mass m_C contains the chassis, motors, battery, and all other parts rigidly attached to it, its moment of inertia with respect to the α -axis is $\Theta_{\alpha,C}$. Its mass center is assumed to be located on the α -axis. The three wheels are identical. They have a mass of m_W each, their mass center is located on their axis of rotation at the distance r_V from the robot's center. Their moments of inertia of their axis of rotation and of their principle axis parallel to the α -axis are $\Theta_{ax,W}$ and $\Theta_{\alpha,W}$, respectively. The robot total mass m_R consists of the chassis mass plus the wheel masses. Its center of mass is assumed to be located on the α -axis at the height h_M above the ground:

$$m_R = m_C + 3m_W \quad (1.42)$$

The equation of motion can easily be derived with *Lagrange's Equations*

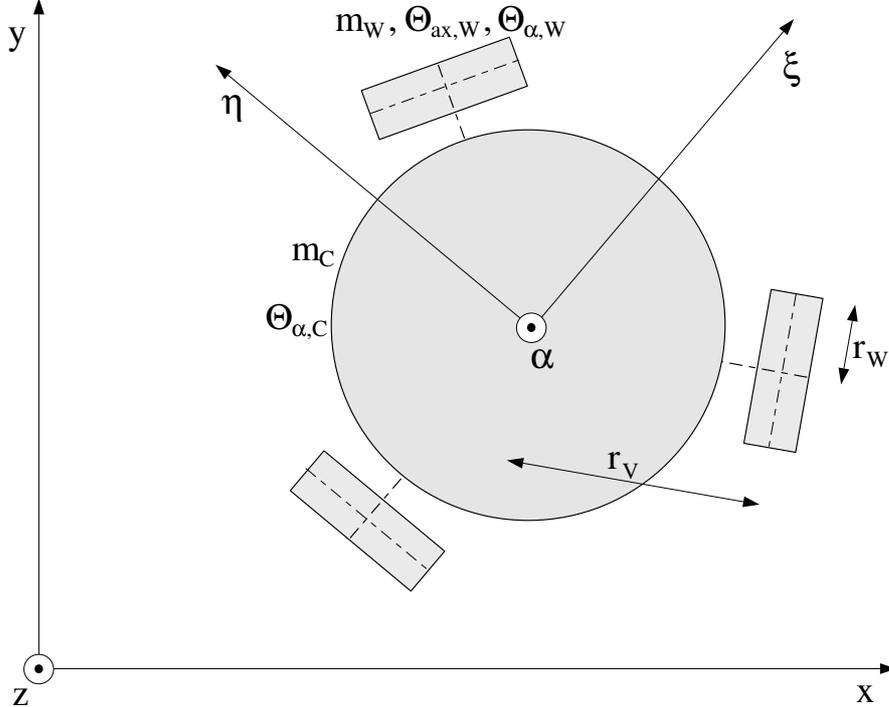


Figure 1.6: Robot mass distributions

expressed in global coordinates.

$$\frac{d}{dt} \left(\frac{\partial E_{\text{kin}}}{\partial \dot{\mathbf{x}}_R} \right) - \frac{\partial E_{\text{kin}}}{\partial \mathbf{x}_R} = \mathbf{F}_R \quad (1.43)$$

The robot's state is expressed with its generalized global position \mathbf{x}_R and its global velocity $\dot{\mathbf{x}}_R$ according to equation (1.3), \mathbf{F}_R is the generalized global force vector according to equation (1.23). The robot's kinetic energy E_{kin} is

$$E_{\text{kin}} = \frac{1}{2} \left(\dot{\mathbf{x}}_R^T \mathcal{M}_{\text{glide}} \dot{\mathbf{x}}_R + \Theta_{\text{ax,W}} \dot{\boldsymbol{\varphi}}^T \dot{\boldsymbol{\varphi}} \right) = \frac{1}{2} \dot{\mathbf{x}}_R^T \mathcal{M} \dot{\mathbf{x}}_R \quad (1.44)$$

in which the mass matrix \mathcal{M} consists of two parts:

$$\mathcal{M} = \mathcal{M}_{\text{glide}} + \mathcal{M}_{\text{roll}} \quad (1.45)$$

The matrix $\mathcal{M}_{\text{glide}}$ represents the complete robot system in case it was skidding over the floor without the wheels rotating. The wheel rotation is contributed by $\mathcal{M}_{\text{roll}}$, for its derivation equations (1.9), (1.8), (1.21), and (1.19) where utilized. All matrices \mathcal{M} , $\mathcal{M}_{\text{glide}}$, and $\mathcal{M}_{\text{roll}}$ are diagonal matrices with $d_1 = d_2$ according to equation (1.17). They can either carry the indices $xy\alpha[\dots]_{xy\alpha}$ or $\xi\eta\alpha[\dots]_{\xi\eta\alpha}$.

$$\mathcal{M}_{\text{roll}} \stackrel{\text{def}}{=} \Theta_{\text{ax,W}} \mathcal{Y}^T \mathcal{Y} \quad (1.46)$$

$$\mathcal{M}_{\text{glide}} \stackrel{\text{def}}{=} \begin{bmatrix} m_R & 0 & 0 \\ 0 & m_R & 0 \\ 0 & 0 & \Theta_{\alpha,R} \end{bmatrix} \quad (1.47)$$

In equation (1.47), the parameter $\Theta_{\alpha,R}$ is the moment of inertia of the α -axis of the robot and the wheels as a rigid system:

$$\Theta_{\alpha,R} = \Theta_{\alpha,C} + 3(\Theta_{\alpha,W} + m_W r_V^2) \quad (1.48)$$

The equation of motion of the robot system in global coordinates is:

$$\boxed{\mathcal{M} \ddot{\mathbf{x}}_R = \mathbf{F}_R} \quad (1.49)$$

It is useful to express the equation of motion in terms of the wheel torques because this represents the physical system. Equations (1.27) and (1.28) can be used to replace the global force vector \mathbf{F}_R with the wheel torques \mathbf{T} .

$$\ddot{\mathbf{x}}_R = \mathcal{M}^{-1} \mathcal{O}^T(\alpha) \mathcal{Y}^T \mathbf{T} \quad (1.50)$$

Note that equation (1.50) is linear except for the time dependent orientation matrix $\mathcal{O}(\alpha(t))$. It can be solved for the wheel torque vector \mathbf{T} . This is useful to dimension the motors for each wheel and for trajectory design. For a given trajectory $\mathbf{x}_R = [x_R(t), y_R(t), \alpha(t)]^T$, the wheel torques can be calculated in order to check for wheel slipping and to compute motor loads. Note that \mathcal{M} and \mathcal{O} switched places in equation (1.50) utilizing equation (1.20). Accelerations are mapped between the coordinate systems like velocities according to equations (1.8) to (1.10). This is because all moving coordinate systems are natural, that means they are moving with the robot.

$$\mathbf{T} = \mathcal{Y}^{-T} \mathcal{M} \mathcal{O}(\alpha) \ddot{\mathbf{x}}_R \quad (1.51)$$

$$\mathbf{T} = \mathcal{Y}^{-T} \mathcal{M} \ddot{\boldsymbol{\xi}}_R \quad (1.52)$$

During acceleration, the wheel circumferential forces do not compute according to equation (1.51) because a portion of the total torque \mathbf{T} created by the propulsion motors is required to accelerate the moment of inertia of each wheel with respect to its own axis of rotation. In analogy to equation (1.45), the torque consists of two contributions, $\mathbf{T}_{\text{glide}}$ and \mathbf{T}_{roll} :

$$\mathbf{T} = \mathbf{T}_{\text{glide}} + \mathbf{T}_{\text{roll}} \quad (1.53)$$

$\mathbf{T}_{\text{glide}}$ is responsible for accelerating the robot chassis, and \mathbf{T}_{roll} for accelerating the wheel masses in the rotation about their respective axes. Only $\mathbf{T}_{\text{glide}}$ creates wheel circumferential forces. For a given acceleration $\ddot{\boldsymbol{\xi}}_R$, the wheel torques $\mathbf{T}_{\text{glide}}$ can be obtained with equation (1.52) by replacing the mass matrix \mathcal{M} with the matrix $\mathcal{M}_{\text{glide}}$.

$$\mathbf{T}_{\text{glide}} = \mathcal{Y}^{-T} \mathcal{M}_{\text{glide}} \mathcal{O}(\alpha) \ddot{\mathbf{x}}_R \quad (1.54)$$

$$\mathbf{T}_{\text{glide}} = \mathcal{Y}^{-T} \mathcal{M}_{\text{glide}} \ddot{\boldsymbol{\xi}}_R \quad (1.55)$$

The wheel circumferential forces now compute according to equation (1.29):

$$\mathbf{t}_{\text{glide}} = \mathcal{U}^{-T} \mathbf{T}_{\text{glide}} \quad (1.56)$$

The dynamic wheel loads in equation (1.34) can also be expressed in terms of the acceleration in global and robot coordinates:

$$\mathbf{G}_{\text{dyn}} = \mathbf{G}_{\text{stat}} + \mathcal{W}^{-1} \mathcal{M}_{\text{glide}} \mathcal{O}(\alpha) \ddot{\mathbf{x}}_R \quad (1.57)$$

$$\mathbf{G}_{\text{dyn}} = \mathbf{G}_{\text{stat}} + \mathcal{W}^{-1} \mathcal{M}_{\text{glide}} \ddot{\boldsymbol{\xi}}_R \quad (1.58)$$

The actual coefficients of friction μ_i at each wheel are constituted by the ratios of the circumferential forces $t_{i,\text{glide}}$ to the dynamic wheel loads $G_{i,\text{dyn}}$:

$$\mu_i \stackrel{\text{def}}{=} \frac{t_{i,\text{glide}}}{G_{i,\text{dyn}}}, \quad i = 1, 2, 3 \quad (1.59)$$

1.3 Robot Hardware

Figure 1.7 shows a drawing created with the computer aided design application Unigraphics (UG). It gives a good impression of the robot's appearance and it also illustrates the locations of some major components. The robot is equipped with three single roller type wheels in this figure, they are mounted directly on the motor gearbox output shafts. The three electric motor-gearbox units are hidden in this view because they are mounted to the bottom of the central plate. Below this plate, there is a bottom plate that has a clearance of only a few millimeters to the floor. This plate was originally designed to carry an optional fuel cell system, a hydrogen storage device, and a DC/DC converter to recharge the battery by the fuel cell during operation. This fuel cell system is still under development because its installation was not the primary focus of this work. The bottom plate served two purposes during the experimental phase: It was used to determine the robot's reference position with a measuring tape, and it has a central hole that allows to place the robot on a fixture in order to rotate it about its α -axis while blocking its translational degrees of freedom. This was useful during camera calibration on the narrow carpet. On the upper side of the central plate are the three servoamplifiers, each carrying a battery pack consisting of 12 cells. In the center of the robot, the PC/104 is located. It is a stack of miniature PC components consisting of a mainboard, three frame grabber cards, a multi I/O card, a CAN card, and a card containing two PCMCIA drive slots. The cameras are mounted in the highest position that is allowed by the RoboCup regulations for middle size robots, they are facing downwards as described in section 5.1.1. The long mast that carries the cameras is used to attach the wireless CAN communication device because no package space was found underneath the cover which can be seen in figure 1.8. During the experiments, it turned out that the radio contact was very sensitive to disturbances so it was good to have the device with its antenna in a free location. All chassis parts of the robot are made of aluminum or plastics to reduce weight. Note that figures 1.7 and 1.8 do not show any nuts, bolts, electric wiring, or switches.

The dribbling mechanism is not part of the robot design in the UG model. The reason for this is that the dribbling mechanism was manufactured by hand and is still in a very premature state. Refer to figure 2.3, p. 38, for photographs of the mechanism. It was the purpose of this work to pay special attention to the development of the omnidirectional propulsion system to solve the Two Body Problem, rather than a ball handling mechanism.

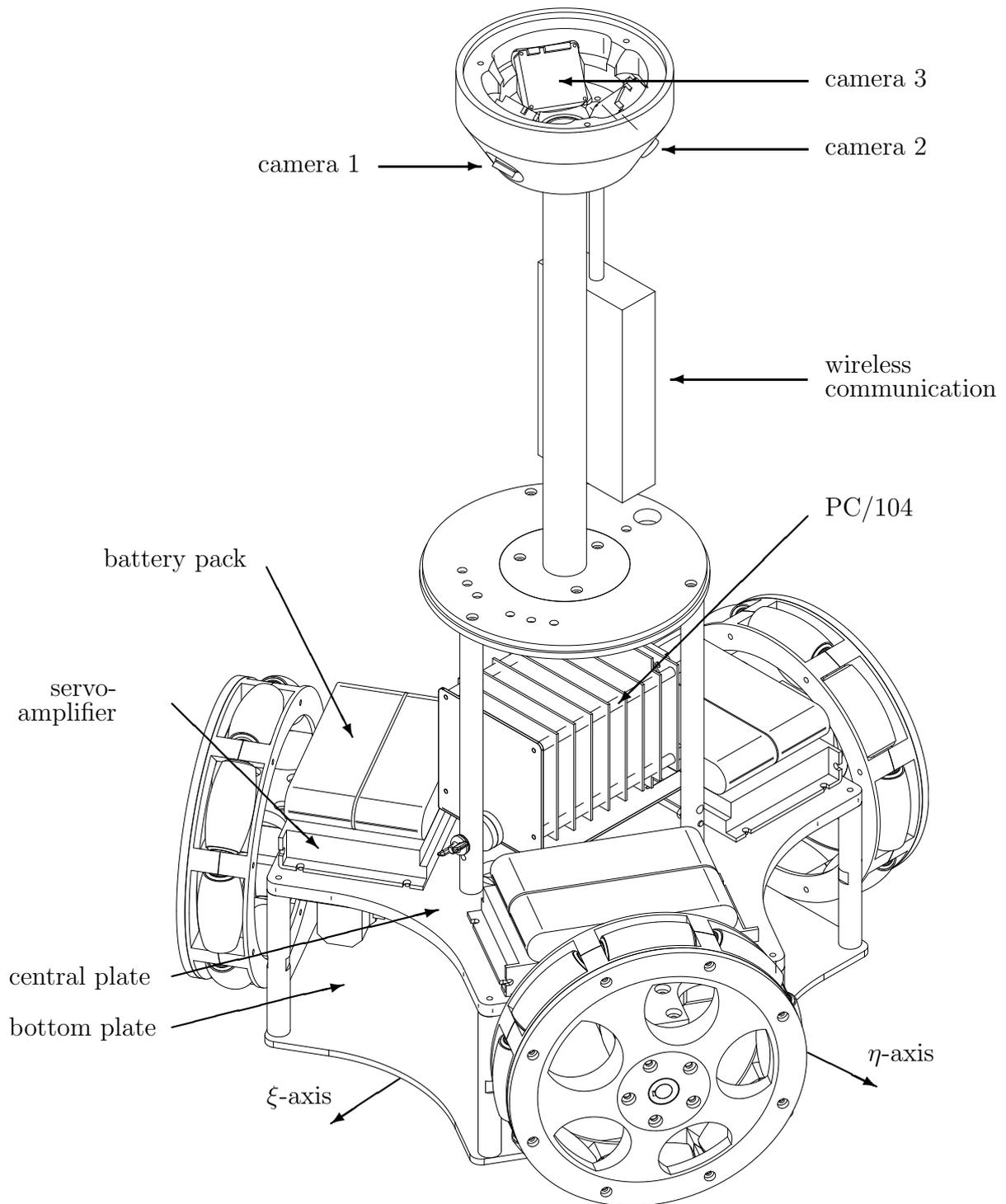


Figure 1.7: Robot assembly without cover

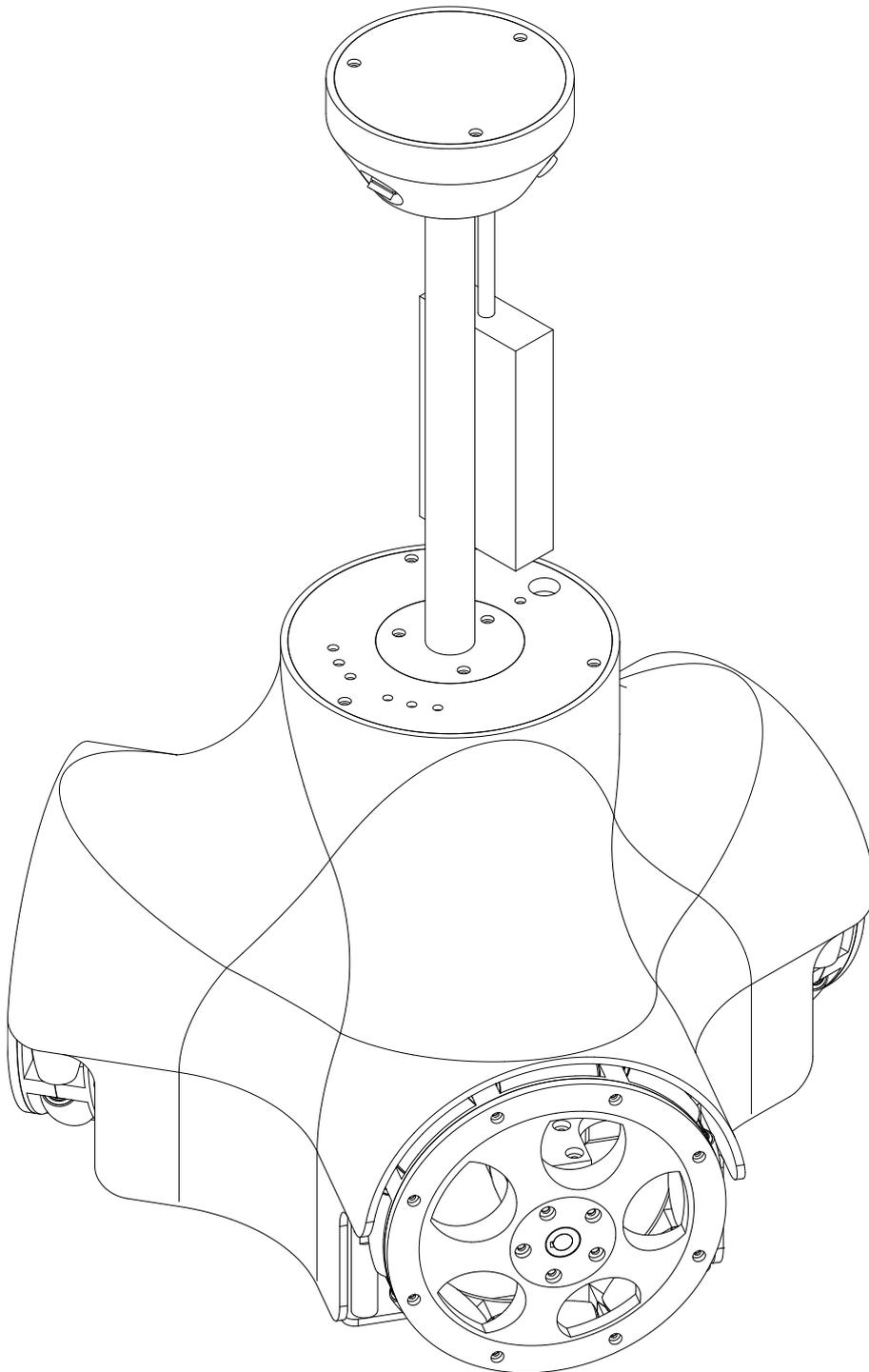


Figure 1.8: Robot assembly with cover

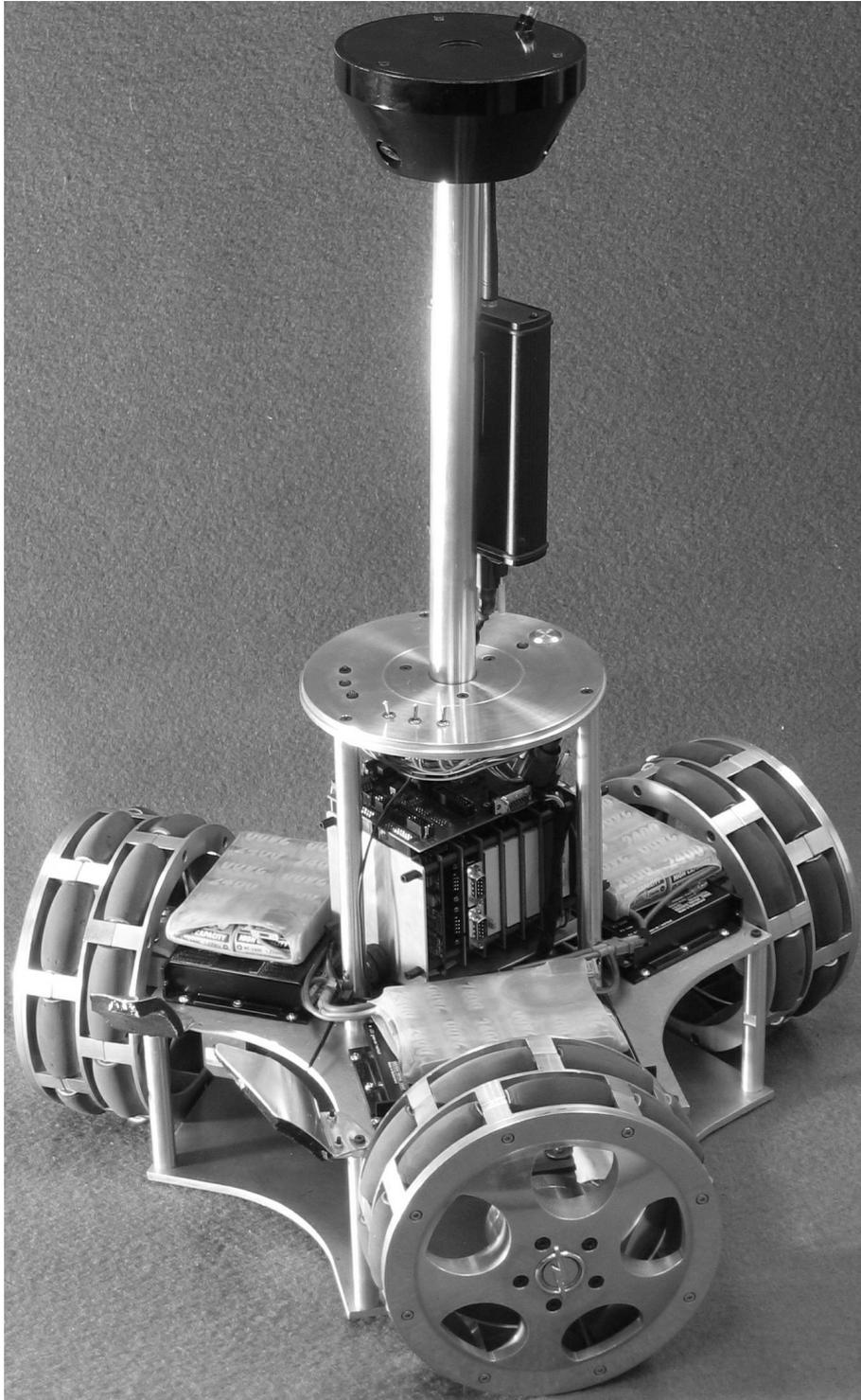


Figure 1.9: Robot photograph without cover

Chapter 2

Actuators

This chapter describes the robot's actuators that constitute the propulsion system. It provides the technical specifications on which the motor dimensioning is based. Furthermore, the dribbling mechanism is briefly discussed since it is the interface used by the robot to act on the ball.

2.1 Technical Specifications

In order to define the technical requirements used for motor dimensioning, some basic assumptions must be made. As the concept of the mobile robot was inspired by the RoboCup competition, the general rules and regulations for the contest in the year 1999 were applied.

Geometrical and Gravimetric Specifications

The RoboCup Middle Size Robot League (F-2000) dictated the following geometrical restrictions for the competition in 1999: The robot's diameter must not exceed 0.63 m. The area covered by a vertical projection of the robot's hull must not exceed 0.2025 m^2 . The robot must be less than 0.8 m high. There is a weight limitation to a maximum of 80 kg. In addition to these restrictions, the robot must be able to move a FIFA standard size 4 or 5 soccer ball that has a radius of about 0.11 m and weighs approximately 0.45 kg.

Both geometrical and weight specifications strongly depend on the hardware parts used for the robot. Defining reasonable values is an iterative procedure because the motors including battery have a significant impact on the overall robot system size and weight.

The robot height will be set to the maximum height allowed by the RoboCup regulations in order to create a good angle of view for the vision

system which needs to be positioned as high as possible. The robot diameter is determined by the length of the motors which are mounted coaxially with the wheels. The wheel diameter is determined by manufacturing issues predominantly. These include roller bearing size and gap size between adjacent rollers.

#	Part	Weight / kg
1	robot chassis	6.0
2	3 wheels	4.5
3	3 motors	6.0
4	battery	2.0
5	electronic devices	1.5
6	fuel cell system	5.0
sum		25.0

Table 2.1: Estimated masses of robot components

Table 2.1 lists the estimated masses of the heaviest parts in the robot. These can be divided into fixed and variable masses. Fixed masses are electronic devices such as computers and cameras, and the fuel cell system including hydrogen storage and a fuel cell stack. Note that the fuel cell system was only considered an option to recharge the battery during operation. It is still under development because it was not the primary focus of this work. The number of cells in the battery is also fixed because the motor controls require a minimum operation voltage that can only be produced by connecting battery cells in series or by changing the voltage with a DC/DC converter. The robot chassis, wheels, and motors can be considered variable masses because they scale with the size of the system. The mass of the motors in table 2.1 is based on the actually selected motors. The mass of the wheels is estimated by taking into consideration the expected robot total mass which determines the size of the roller bearings. The estimated mass of the robot chassis is based on the expected robot diameter and height using aluminum as material.

Dynamical Specifications

The robot's dynamics is defined by its maximum velocity and maximum acceleration. The maximum velocity is only limited by the motors' capabilities. The maximum acceleration has two physical limitations: The first one is the coefficient of friction between the wheels and the ground constraining the force that each wheel can transmit to the ground. Due to the robot's symme-

try, the ξ - and η -directions are outstanding and it is sufficient for both kinetic and kinematic examinations to consider these two directions exclusively. A maximum coefficient of friction of $\mu_{\max} = 1.0$ is assumed. This is conservative compared to the value of $\mu = 1.2$ which a rubber tire can reach on asphalt [49]. The second limitation is the height of the robot's mass center. If the mass center is too high, the robot may fall over at large accelerations. The following list summarizes the assumptions made for calculating the dynamic operating conditions:

- The robot undergoes a pure translational motion, this excludes rotation about the vertical axis, $\dot{\alpha} = 0$.
- Only the ξ - and η -directions are considered for kinetics and kinematics.
- The friction coefficient between wheels and ground is $\mu_{\max} = 1.0$.
- The robot mass center is at the height h_M above the ground.

Table 2.2 shows the maximum theoretical accelerations in both directions ξ and η that are possible for different robot setups. Due to the robot's symmetry, it is not necessary to consider negative acceleration in η -direction. Tables 2.2(a), (b), and (c) were used for dimensioning the robot, the parameters $m_R = 25$ kg and $r_R = 0.225$ m according to tables 2.1 and 2.3 were assumed to calculate the forces, while the height of the center of mass was varied to examine its influence. The accelerations in table 2.2 are limited by two constraints, either the largest absolute entry of the friction coefficient vector $\boldsymbol{\mu}$ reaches the maximum of $|\hat{\boldsymbol{\mu}}| = 1$, or a wheel load becomes zero. Acceleration values are provided in multiples of the gravitational acceleration g .

Table 2.2(a) assumes a zero height of the center of mass, this results in all wheel loads remaining constant and equal to the static wheel load \mathbf{G}_{stat} for any acceleration. It indicates that the robot can accelerate with $0.58g$ in positive and negative ξ -direction, wheels 1 and 2 reach maximum circumferential force while wheel 3 is idle in that case. The maximum acceleration in η -direction is only $0.5g$ and in this case wheel 3 transmits the limiting force, while wheels 1 and 2 only transmit half as much force each.

Tables 2.2(b) and (c) assume non-zero mass center heights h_M of 0.10 m and 0.13 m, respectively. This causes the wheel loads to vary dynamically depending on the direction and magnitude of the robot acceleration. For $h_M = 0.10$ m, the maximum acceleration in positive ξ -direction reduces to $0.46g$ because the load on wheels 1 and 2 is partially shifted to wheel 3. For the same reason, the maximum acceleration in negative ξ -direction increases to $0.78g$. Acceleration in η -direction is not influenced because the load on the restricting wheel 3 does not shift due to the symmetry. The mass center

Acceleration $\ddot{\xi}_R^T / g$	Circumfer. force \mathbf{t}^T / N	Wheel load $\mathbf{G}_{\text{dyn}}^T / \text{N}$	Friction coeff. $\boldsymbol{\mu}^T / -$
0.58 0 0	-81.8 81.8 0	81.8 81.8 81.8	-1 1 0
-0.58 0 0	81.8 -81.8 0	81.8 81.8 81.8	1 -1 0
0 0.5 0	-40.9 -40.9 81.8	81.8 81.8 81.8	-0.5 -0.5 1
(a) $h_M = 0 \text{ m}$			
0.46 0 0	-65.1 65.1 0	65.1 65.1 115.1	-1 1 0
-0.78 0 0	110.0 -110.0 0	110.0 110.0 25.3	1 -1 0
0 0.5 0	-40.9 -40.9 81.8	113.2 50.3 81.8	-0.36 -0.81 1
(b) $h_M = 0.10 \text{ m}$			
0.43 0 0	-61.3 61.3 0	61.3 61.3 122.6	-1 1 0
-0.87 0 0	122.5 -122.5 0	122.6 122.6 0	1 -1 0
0 0.5 0	-40.9 -40.9 81.8	122.6 40.9 81.8	-0.33 -1 1
(c) $h_M = 0.13 \text{ m}$			
0.42 0 0	-64.1 64.1 0	64.1 64.1 138.5	-1 1 0
-0.75 0 0	114.8 -114.8 0	133.3 133.3 0	0.86 -0.86 0
0 0.46 0	-41.1 -41.1 82.2	136.6 41.1 88.9	-0.30 -1 0.93
(d) actual values			

Table 2.2: Wheel loads during acceleration

height $h_M = 0.13 \text{ m}$ is a special case with respect to two aspects. First, it represents the maximum height h_M for which an acceleration in η -direction of $0.5g$ is possible. If the mass center is higher than 0.13 m , wheel 2 will be restrictive rather than wheel 3 due to its reduced wheel load. Second, the acceleration in negative ξ -direction is limited by the zero wheel load of wheel 3 in addition to the maximum coefficients of friction at wheels 1 and 2.

For comparison, table 2.2 (d) represents the actual robot system as it was built. The physical parameters are provided in table 6.6, p. 127. The mass center is even higher compared to table 2.2 (c). Therefore, the acceleration in negative ξ -direction is limited by the zero wheel load of wheel 3, and the acceleration in η -direction is limited by the friction coefficient of wheel 2 due to its reduced wheel load.

2.2 Motor Dimensioning

Table 2.3 summarizes the geometrical, weight, and dynamical specifications that were used to dimension the propulsion motors of the robot.

Parameter	Symbol	Value
robot radius	r_R	0.225 m
robot height	h_R	0.800 m
wheel radius	r_W	0.110 m
max. weight	m_R	25 kg
max. acceleration	a_{\max}	5 m/s ²
max. velocity	v_{\max}	5 m/s

Table 2.3: Robot technical specifications for design

The specified acceleration a_{\max} is very demanding. It is equal to the smaller value of the static wheel load cases in table 2.2 (a), and it is equal to the greater value of the dynamic wheel load cases in positive directions in tables 2.2 (b) and (c). The acceleration in negative ξ -direction was neglected because the robot was not designed for accelerating backwards, if the robot is dribbling a ball, the longitudinal deceleration must be soft in order not to loose the ball. The lateral acceleration in η -direction is required for driving curves, this is the most important case because the centripetal forces will limit the robot's maximum velocity. The specified velocity was chosen according to the following goal: The longest straight distance on a RoboCup soccer field is approximately 10 m. The fastest way to travel this distance is accelerating for the first half followed by decelerating for the second half with a_{\max} for 1.424 s, respectively, reaching a peak velocity of 7.071 m/s. This peak velocity can be reduced to $v_{\max} = 5$ m/s by accelerating with a_{\max} , holding constant velocity of v_{\max} , and decelerating with a_{\max} for 1 s, respectively. The whole trip now lasts 3 s rather than 2.828 s. This is a reasonable tradeoff.

The motors driving the wheels are dimensioned according to the specifications in table 2.3. Note that these values are only estimates, some of them changed with the actual robot hardware. For a list of the actual robot physical parameters refer to table 6.6, p. 127. According to the operating conditions defined on page 29, only translational motion in the ξ - and η -directions is considered. Table 2.4 combines the maximum velocities with the maximum accelerations in both directions to determine the maximum power required for each motor. The three entries of all vectors are listed horizontally in a row. Each row represents a different load case, they are sorted by robot velocity and acceleration.

Robot			Wheel								
Velocity	Accel.		Speed			Torque			Power		
$\dot{\xi}_R^T / \frac{m}{s}$	$\ddot{\xi}_R^T / \frac{m}{s^2}$		$\dot{\varphi}^T / \frac{rad}{s}$			\mathbf{T}^T / Nm			\mathbf{P}^T / W		
5 0 0	5 0 0		-39.4	39.4	0	-8.3	8.3	0	326	326	0
5 0 0	0 5 0		-39.4	39.4	0	-4.8	-4.8	9.6	188	-188	0
0 5 0	5 0 0		-22.7	-22.7	45.4	-8.3	8.3	0	188	-188	0
0 5 0	0 5 0		-22.7	-22.7	45.4	-4.8	-4.8	9.6	109	109	436

Table 2.4: Motor dimensioning: Robot load cases

The maximum wheel speed $\dot{\varphi}_{\max} = 45.4 \text{ rad/s}$ according to equation (1.9), p. 16, is reached by wheel 3 in the last load case. The maximum torque according to equation (1.52), p. 22, is $T_{\max} = 9.6 \text{ Nm}$. It also occurs at wheel 3 in the same load case. Using equation (1.32), p. 18, this causes the maximum power demand $P_{\max} = 436 \text{ W}$ to occur at wheel 3 as well.

In the following, the components of the propulsion system are described. They are also listed in tables A.1, p. 169, and A.2, p. 171. Power supply to the propulsion motors is limited by the battery and the digital servoamplifiers (DES). Figure 2.1 shows the electric power scheme of the propulsion system. It illustrates the electric currents and voltages used for dimensioning.

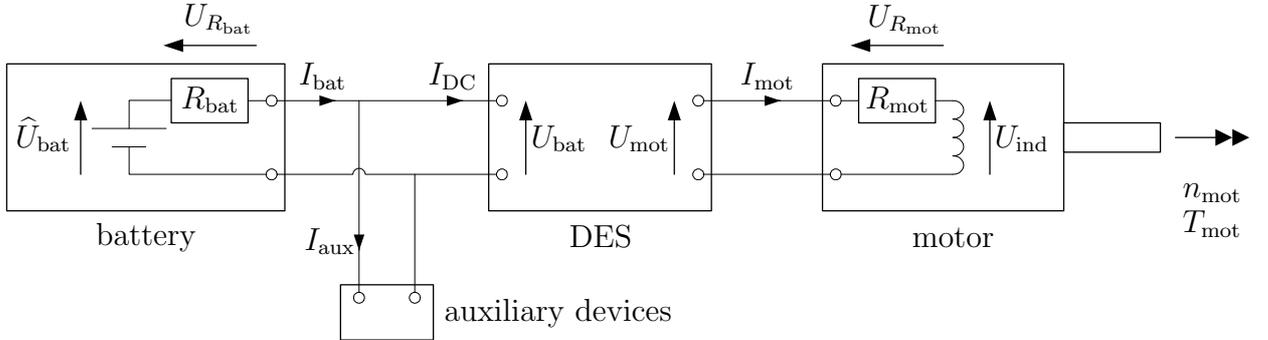


Figure 2.1: Motor dimensioning: Electric power scheme

The battery consists of 36 rechargeable Ni-Cd cells connected in series. Each cell reaches a maximum idle voltage of 1.4 V when fully charged, but during operation, the cell idle voltage quickly drops to the nominal cell voltage 1.2 V. The battery pack therefore operates at idle voltages between 40 and 50 V with a nominal value $\hat{U}_{\text{bat}} = 43.2 \text{ V}$. The internal resistance of the whole pack was determined in an experiment to be $R_{\text{bat}} = 0.618 \Omega$. Therefore,

at high battery currents the actual supply voltage U_{bat} can drop significantly:

$$U_{\text{bat}} = \hat{U}_{\text{bat}} - R_{\text{bat}} \cdot I_{\text{bat}} \quad (2.1)$$

It is assumed that the auxiliary consumers (e.g. computer, cameras, communication device) draw the constant current $I_{\text{aux}}=1$ A from the batteries. The propulsion system is supplied with the electric current I_{DC} . Hence, the battery current computes as:

$$I_{\text{bat}} = I_{\text{DC}} + I_{\text{aux}} \quad (2.2)$$

The digital servoamplifiers (DES) are responsible for controlling and commutating the electric motors propelling the robot. Each wheel is operated by an individual motor, and each motor is controlled by an individual servoamplifier. This is described in section 4.1, p. 71. The DES operate at voltages between 24 and 70 V. They can provide maximum continuous and peak currents of $I_{\text{mot}}=10$ A and 30 A, respectively. Their maximum efficiency is specified as $\eta_{\text{DES}}=92\%$:

$$U_{\text{mot}} \cdot I_{\text{mot}} = U_{\text{bat}} \cdot I_{\text{DC}} \cdot \eta_{\text{DES}} \quad (2.3)$$

The maximum effective output voltage U_{mot} created by the DES is specified by the supplier as $v_{\text{DES}}=90\%$ of the supply voltage U_{bat} . This allows the following calculation of the effective motor current I_{mot} :

$$U_{\text{mot}} = U_{\text{bat}} \cdot v_{\text{DES}} \quad (2.4)$$

$$I_{\text{mot}} = I_{\text{DC}} \cdot \eta_{\text{DES}} / v_{\text{DES}} \quad (2.5)$$

The motors that were chosen for the robot's propulsion are electronically commutated DC brushless electric motors. Their rotor carries neodymium permanent magnets and the stator holds the 3 phase Y-connected windings with an ohmic resistance of $R_{\text{mot}}=0.64\Omega$ and an inductivity of 0.260 mH from phase to phase. The maximum continuous torque is 0.3 Nm at 5000 rpm. This draws the maximum continuous current $I_{\text{mot}}=6$ A. The maximum speed is 6000 rpm at the rated voltage $U_{\text{mot}}=36$ V. A higher voltage will allow a proportionally higher motor speed which can be computed with the motor constants $k_M=0.054$ Nm/A and $k_N=175$ rpm/V specified by the manufacturer. The slope of the motor characteristic is $\Delta n/\Delta T=2100$ rpm/Nm. The maximum motor efficiency is specified as $\eta_{\text{mot}}=85\%$. At maximum speed and zero load, the motor draws the current $I_{\text{mot},0}=0.370$ A. The following expressions allow the calculation of the motor speed n_{mot} and the motor torque T_{mot} :

$$n_{\text{mot}} = k_N \cdot U_{\text{mot}} - \frac{\Delta n}{\Delta T} \cdot k_M \cdot I_{\text{mot}} \quad (2.6)$$

$$T_{\text{mot}} = k_M \cdot I_{\text{mot}} - T_{\text{fric}} \quad (2.7)$$

Equation (2.6) can easily be derived from the effective voltage U_{mot} of the motor. This voltage splits in two contributions, the ohmic voltage $U_{R_{\text{mot}}}$ and the voltage induced by the motor coils U_{ind} :

$$U_{\text{mot}} = U_{R_{\text{mot}}} + U_{\text{ind}} \quad (2.8)$$

$$= R_{\text{mot}} \cdot I_{\text{mot}} + n_{\text{mot}} / k_N \quad (2.9)$$

Solving for n_{mot} and substituting the following definition of the motor characteristic $\Delta n / \Delta T$ yields equation (2.6). The motor friction T_{fric} reduces the usable motor torque T_{mot} . It can be estimated from the motor no-load current $I_{\text{mot},0}$:

$$T_{\text{fric}} = k_M \cdot I_{\text{mot},0} \cdot \text{sign}(n_{\text{mot}}) \quad (2.10)$$

$$\frac{\Delta n}{\Delta T} \stackrel{\text{def}}{=} \frac{60 \text{ rpm}}{2\pi \frac{\text{rad}}{\text{s}}} \cdot \frac{R_{\text{mot}}}{k_M^2} \quad (2.11)$$

A two stage planetary gear reduces the motor speed by a gear ratio of $i_{\text{gear}} = 12.25$. The gearbox output shaft is coaxial with the motor input shaft. It can carry a radial force of 150 N which is greater than the maximum dynamic wheel load occurring in table 2.2. Therefore, the wheels can be attached to the gearbox output shaft directly allowing a simple mechanical design. According to the manufacturer, the gearbox has a maximum efficiency of $\eta_{\text{gear}} = 81\%$.

Taking the gearbox efficiency and ratio into consideration, the wheel torque and speed can be computed for motor dimensioning as

$$T = T_{\text{mot}} \cdot i_{\text{gear}} \cdot \eta_{\text{gear}} \quad (2.12)$$

$$n_W = n_{\text{mot}} \cdot i_{\text{gear}}^{-1} \quad (2.13)$$

$\frac{I_{\text{mot}}}{\text{A}}$	$\frac{I_{\text{DC}}}{\text{A}}$	$\frac{I_{\text{bat}}}{\text{A}}$	$\frac{U_{\text{bat}}}{\text{V}}$	$\frac{U_{\text{mot}}}{\text{V}}$	$\frac{T_{\text{mot}}}{\text{Nm}}$	$\frac{n_{\text{mot}}}{\text{rpm}}$	$\frac{T}{\text{Nm}}$	$\frac{\dot{\varphi}}{\text{rad/s}}$
0.37	0.4	1.4	42.4	38.1	0.0	6648	0.0	56.8
6.74	6.6	7.6	38.5	34.7	0.347	5311	3.4	45.4
10.0	9.8	10.8	36.5	32.9	0.524	4628	5.2	39.6
18.15	17.8	18.8	31.6	28.4	0.967	2918	9.6	24.9
30.0	29.3	30.3	24.4	22.0	1.612	433	16.0	3.7

Table 2.5: Motor dimensioning: Results for propelling motors

Table 2.5 shows the calculated values for different operating conditions. The effective motor current I_{mot} was varied as an independent variable and all

other values were calculated subsequently. The first row shows the results for the case of zero motor load. This requires the effective current $I_{\text{mot},0} = 0.370$ A to the motors. The battery current includes the fixed current to the auxiliary devices. The battery voltage drops due to its internal resistance from $\hat{U}_{\text{bat}} = 43.2$ V to 42.4 V. This allows a maximum motor voltage of 38.1 V due to the restriction on the DES output voltage. Both motor torque and wheel torque are zero according to the assumption made for this load case. The maximum wheel speed in this case is 56.8 rpm, this is above the requirement. Row two represents the maximum torque for which the required motor speed can be reached. Row four represents the maximum speed for which the required motor torque can be reached. Rows three and five show the results for the maximum continuous and peak current allowed by the DES. In the peak current case, the battery voltage drops significantly and it almost drops to the lower operating limit of the DES which is 24 V. In this case, the maximum wheel torque of 16 Nm can be created.

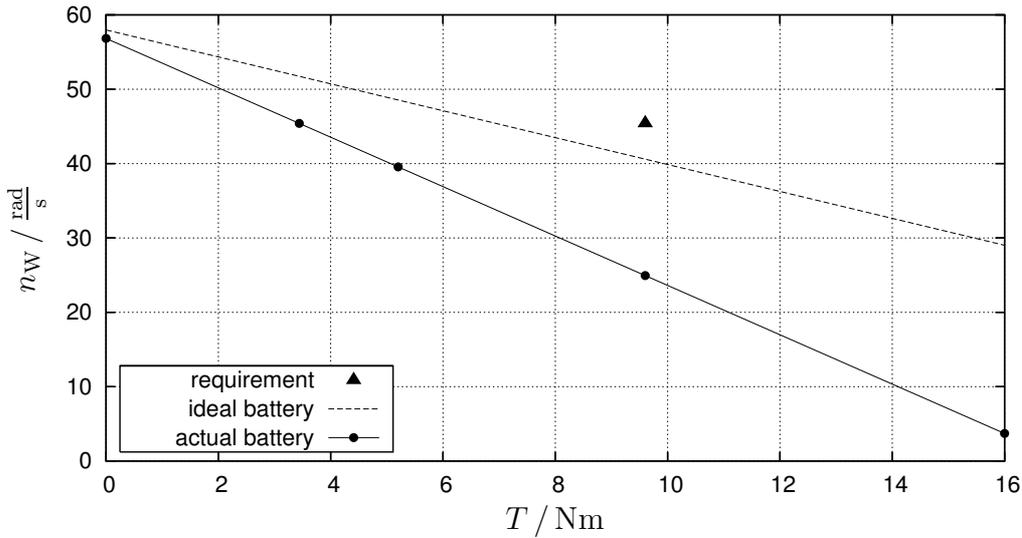


Figure 2.2: Motor dimensioning: Speed–torque characteristic

Figure 2.2 plots the results of the motor dimensioning calculations. The two curves represent pairs of maximum wheel speed and maximum torque for different operating conditions. The dashed curve shows the results with an ideal battery with a constant supply voltage of 43.2 V and zero internal resistance. The solid curve shows the results with the actual battery. The five points on this curve marked with bullets indicate the operating conditions listed in table 2.5. The required maximum wheel speed and torque obtained from table 2.4 is plotted as a triangle. Both curves fail to meet

the required wheel speed and torque at the same time. However, the motor can reach both requirements separately. The maximum achievable torque and speed for these two cases are listed in rows two and four in table 2.5. The selected propulsion components therefore do not totally meet the design specifications. They are regarded as an acceptable tradeoff for the following reasons: First, it should be kept in mind that the dynamic load cases used in table 2.4 are not absolutely representative. Driving with maximum velocity in η -direction and accelerating with maximum acceleration at the same time is a very unlikely operating condition for the robot. In addition to that, the load cases include only translational motion, but the robot will typically also rotate and accelerate about its α -axis. Second, the physical parameters assumed in tables 2.3 and 2.2 were preliminary estimates. Especially table 2.2 (d) shows that for the actual implementation of the robot, the acceleration of 5 m/s^2 can not be reached due to the height of the robot's mass center.

During the driving experiments, the wheel torque was calculated from the measured effective motor currents I_{mot} :

$$T = \begin{cases} (I_{\text{mot}} \cdot k_M - T_{\text{fric}}) \cdot i_{\text{gear}} \cdot \eta_{\text{gear}} & , P > 0 \quad (\text{propel}) \\ (I_{\text{mot}} \cdot k_M - T_{\text{fric}}) \cdot i_{\text{gear}} / \eta_{\text{gear}} & , P < 0 \quad (\text{generate}) \end{cases} \quad (2.14)$$

Two operating conditions exist for the motors. They can be distinguished by the mechanic power P at the wheel defined in equation (1.32), p. 18: If the power is positive, $P > 0$, the motor is propelling the wheel, while $P < 0$ means the wheel is propelling the motor. In the latter case the motor is acting as a generator recharging the battery. All previous considerations for dimensioning the motors were made for the propelling case. However, it should be kept in mind that in the generating case the same absolute motor current can balance a much higher torque at the wheel because 19% of the torque is dissipated by the gearbox efficiency. Refer to section 6.4 for the experimental results that validate the motor dimensioning results.

2.3 Dribbling Mechanism

It was the primary objective of this work to develop a robot platform that is able to solve the Two Body Problem defined in the introduction. Special attention was paid to the agility of the robot that allows adjusting to different driving demands, this was solved with the omnidirectional propulsion system. It was hoped that this agility in combination with a sophisticated planning of the robot's trajectory could solve the Two Body Problem without the support of a complicated ball handling mechanism. Therefore, the

dribbling mechanism was kept as simple as possible, this allowed to optimize the trajectory planning methods. It will always be possible to refine the mechanism in the future to further improve the robot's ball skills.

It was also tried to dribble the ball without any mechanism at all, the ball was only restricted by the natural concave groove between the robot's front wheels, compare figures 1.7 and 1.9, p. 24. Preliminary experiments were conducted without the dribbling mechanism, but it was impossible to dribble the ball with that configuration. Two reasons could be identified for the failure. First, the ball was a leather soccer ball according to the RoboCup regulations, its skin consisted of 12 pentagonal and 20 hexagonal patches connected with seams. Therefore, it significantly deviated from a perfect sphere. As a result, the ball often rolled sideways and consequently got lost since there was no guidance. Second, the gyroscopic forces that were neglected for the derivation of the equation of motion for a rolling ball in section 3.2.1 do play a role. In contrast to a sliding disk that will change its direction of motion if an external force is applied, a rolling ball stabilizes itself. Therefore, a ball will be persistent in keeping its rolling direction, and this contributes to its tendency to get lost in curves.

The situation could be improved significantly with little effort. The dribbling mechanism that was used for the experiments is depicted from different viewpoints in figure 2.3. It consists of two pieces of aluminum sheet metal that are curved like the ball and are attached to the robot's central plate. On their inner side, they are covered with porous cellular India rubber to increase friction between the mechanism and the ball. The plates are shaped such that their front end touches the ball first, they are flexible and bend a bit upwards if the ball is in its neutral position. Their length was trimmed such that they meet the RoboCup geometrical regulations for ball handling. They are designed such that the ball rolls freely when dribbled by the robot, this is another RoboCup requirement. Note that the two plates provide very little guidance to the ball in lateral direction. They do not hold the ball sideways, but they apply friction at their top ends to stabilize and damp the ball's agitated rolling motion. For a discussion of the experimental results obtained with this mechanism, refer to section 6.6, p. 155.



(a) Side view



(b) Front view



(c) Isometric view



(d) Top view

Figure 2.3: Ball dribbling mechanism photographs

Chapter 3

Trajectory Planning

This chapter describes the approaches that have been developed to provide an analytical expression of the robot trajectory including its derivatives as functions of time. There are two modes of robot motion: The robot stand-alone mode without a passive object, and the dribbling mode in which the robot affects the trajectory of a passive object. The latter mode is referred to as the Two Body Problem.

3.1 Two Body Problem

The dribbling problem that arises from the RoboCup competition can be stated verbally as follows:

”Based on the physical model of a passive object, formulate an analytical expression of the robot state as a function of time that will push the passive object along a predefined two dimensional planar trajectory.

The trajectory is provided in terms of control points that need to be passed or surrounded.”

The difficulty herein is the restriction that the passive object must only be pushed, it is not possible for the robot to apply a tensile force to the object. It is therefore required that the robot always stays behind the object in a sense that the vector of force applied to the passive object is always pointing from the robot to the object’s center. This will be elaborated in detail in section 3.2.

3.2 Interaction between Passive Object and Robot

The passive object is assumed symmetrical with respect to rotation about a vertical axis. It is therefore characterized sufficiently by two independent coordinates, unlike the robot which requires a third coordinate for its orientation.

Using the unit vectors $\{\mathbf{e}_x, \mathbf{e}_y\}$ defined in section 1.2.1, the positions of both the passive object and the robot relative to the fixed global reference frame can be expressed as a two dimensional vector:

$$\vec{x}_B \stackrel{def}{=} x_B \mathbf{e}_x + y_B \mathbf{e}_y = [x_B \ y_B]^T \quad (3.1)$$

$$\vec{x}_R \stackrel{def}{=} x_R \mathbf{e}_x + y_R \mathbf{e}_y = [x_R \ y_R]^T \quad (3.2)$$

The two dimensional vector representing the force acting on the passive object follows the same formalism, the reacting force on the robot is ignored in all calculations because the ball mass can be neglected compared to the robot mass.

$$\vec{F}_B \stackrel{def}{=} F_{x,B} \mathbf{e}_x + F_{y,B} \mathbf{e}_y = [F_{x,B} \ F_{y,B}]^T \quad (3.3)$$

3.2.1 Equation of Motion for the Passive Object

Two types of passive objects are considered, sliding and rolling objects.

For a sliding passive object, the equation of motion is trivial because it is equal to that of a rigid body undergoing a pure translational motion. The equation of motion is stated in equation (3.9).

For a rolling ball, rotational energy contributions have to be considered as well as gyroscopic forces that occur when the orientation of the axis of rotation is altered. The derivation presented here is based on a pure energetic consideration. Gyroscopic forces need to be considered separately but are neglected here. The principal moment of inertia with respect to any axis through the ball's center of gravity is

$$\Theta_B = \frac{2}{3} m_B r_B^2. \quad (3.4)$$

This assumes that the ball is hollow and all mass m_B is homogenously distributed on the ball's circumference, at a constant radius r_B . The kinetic energy of a rolling ball with a translational velocity $\dot{\vec{x}}_B$ and an angular speed ω is

$$E_{\text{kin}} = \frac{1}{2} m_B (\dot{\vec{x}}_B)^T (\dot{\vec{x}}_B) + \frac{1}{2} \Theta_B \omega^2. \quad (3.5)$$

The following rolling condition applies when the ball rolls in the direction of the translational motion and corresponds to zero slip:

$$(\omega r_B)^2 = (\dot{\vec{x}}_B)^T(\dot{\vec{x}}_B) \quad (3.6)$$

The kinetic energy of the rolling ball therefore is

$$E_{\text{kin}} = \frac{1}{2} \left(\frac{5}{3} m_B \right) (\dot{\vec{x}}_B)^T(\dot{\vec{x}}_B). \quad (3.7)$$

A Lagrangian approach according to equation (1.43), p. 21, with \vec{x}_B as generalized coordinate and \vec{F}_B as force leads to the conservative formulation of the equation of motion for the rolling object

$$M_B \ddot{\vec{x}}_B = \vec{F}_B. \quad (3.8)$$

A non-conservative viscous damping term $D_B \dot{\vec{x}}_B$ is introduced additionally to account for sliding and rolling friction of the passive object on the ground and on its contact points with the robot. This leads to the common equation of motion for both sliding and rolling passive objects which will be used in the following:

$$\boxed{M_B \ddot{\vec{x}}_B + D_B \dot{\vec{x}}_B = \vec{F}_B} \quad (3.9)$$

The generalized mass M_B is different for sliding and rolling objects:

$$M_B \stackrel{\text{def}}{=} \begin{cases} m_{\text{disk}} & \text{for sliding disk} \\ \frac{5}{3} m_B & \text{for rolling ball} \end{cases} \quad (3.10)$$

In the following sections, the passive object will be referred to as ball.

3.2.2 Kinematics and Kinetics of Interaction

Figure 3.1 illustrates the kinematics of the interaction between robot and ball. The ball is guided relative to the robot by two elastic springs with a stiffness k . The springs are perpendicular to each other which leads to an effective stiffness of k in both ξ - and η -direction. The dashed circle around the point $(\xi = \xi_0, \eta = 0)$ indicates the ball's neutral position for which the springs are relaxed. This neutral position relative to the robot center expressed in global coordinates \vec{x}_0 depends on the robot's orientation α :

$$\vec{x}_0 \stackrel{\text{def}}{=} \xi_0 (\cos(\alpha) \mathbf{e}_x + \sin(\alpha) \mathbf{e}_y) = \xi_0 [\cos(\alpha) \sin(\alpha)]^T \quad (3.11)$$

This allows to express the ball's dislocation $\Delta\vec{x}$ from the neutral position in terms of robot and ball position:

$$\Delta\vec{x} = \vec{x}_B - \vec{x}_R - \vec{x}_0. \quad (3.12)$$

At this point it is assumed that during dribbling the ball does not leave the robot but always compresses the springs, because the springs cannot pull the ball towards the robot. This leads to the following kinetic expression for the force acting between springs and ball.

$$\vec{F}_B = -k \Delta\vec{x} \quad (3.13)$$

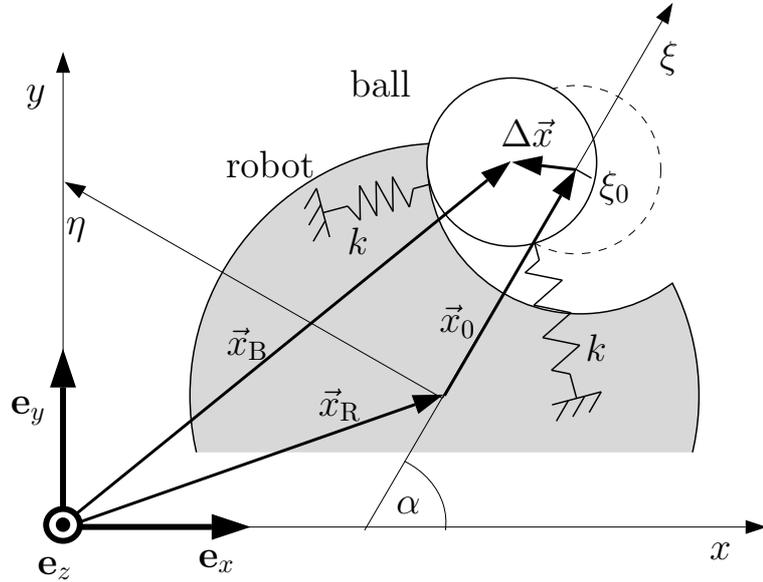


Figure 3.1: Interaction between robot and ball

Now the best position and orientation for the robot to guide the ball can be deduced. Solving equation (3.12) for \vec{x}_R and considering equations (3.13) and (3.9) yields the robot's position

$$\vec{x}_R = \vec{x}_B + \frac{M_B}{k} (\ddot{\vec{x}}_B + \delta \dot{\vec{x}}_B) - \vec{x}_0(\alpha) \quad (3.14)$$

with the viscous damping ratio δ defined as

$$\delta \stackrel{def}{=} \frac{D_B}{M_B}. \quad (3.15)$$

It has been mentioned before that the robot must always stay 'behind' the ball in a sense that it only pushes it because the springs cannot apply tensile forces. The best choice for the robot is to keep the ball always in the middle of its groove because that way it is most unlikely that the ball gets lost. In other words, the ball's center should always be located on the robot's ξ -axis. Therefore, the springs must always be compressed evenly, which leads to a resulting force \vec{F}_B along the ξ -axis. The robot's orientation α is consequently determined by the direction of \vec{F}_B according to equation (3.9)

$$\tan(\alpha) = \frac{F_{y,B}}{F_{x,B}} = \frac{\ddot{y}_B + \delta \dot{y}_B}{\ddot{x}_B + \delta \dot{x}_B} \quad (3.16)$$

3.3 Trajectory Parameterizations

It was postulated that, based on the physical equations of motion of ball and robot, the respective two dimensional trajectories are solved for as analytical functions of time. The following sections describe three different approaches that were developed to provide a feasible solution for the Two Body Problem.

Two cases of robot motion can be distinguished, driving freely and dribbling a ball. The first case is much simpler because there is no interaction between two objects, and for all approaches discussed here it can easily be derived from the more complicated Two Body Problem. The focus of this chapter is therefore dedicated to the dribbling of a ball, and the simplification steps for free robot motion are interspersed where necessary. The trajectory design in all approaches will follow these three objectives:

- The trajectory will always pass through or surround a set of control points that are used as landmarks to define the course from start to end.
- The time required from start to end of the course will be minimized.
- Constraints on the robot's velocity and acceleration capabilities must not be violated.

All approaches were developed with the primary goal to yield an algorithm that is robust, easy to implement and fast to compute. Therefore, determining the robot trajectory by solving differential equations numerically was not considered from the beginning because of the expected complexity of the formulation and the solving methods. It seemed much more pragmatic to start with a given explicit parameterization for the trajectory coordinates and optimize the occurring coefficients.

Starting with the two non-linear equations (3.14) and (3.16) describing the interaction between robot and ball, a remarkable property can be observed that will be exploited in the following. In equation (3.16), the only robot coordinate is the orientation α , it occurs only once on the left hand side of the equation. In equation (3.14), α appears on the right hand side while the robot coordinates x_R and y_R only show up on the left hand side. The ball's coordinates x_B and y_B occur in form of three different derivatives with respect to time, these are \vec{x}_B , $\dot{\vec{x}}_B$, and $\ddot{\vec{x}}_B$, on the right hand side of the equations. This fact makes the following procedure suited for solving the Two Body Problem:

First, determine an analytical formulation of the ball's two dimensional trajectory \vec{x}_B as a function of time that allows the computation of velocity $\dot{\vec{x}}_B$ and acceleration $\ddot{\vec{x}}_B$, as well as higher order derivatives with respect to time. Second, use equation (3.16) to compute the robot's orientation α , and third, determine its position \vec{x}_R with equation (3.14). This procedure is illustrated in figure 3.2.

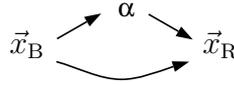


Figure 3.2: Procedure for solving the Two Body Problem

In case of free driving, the second step can be omitted, the trajectory is used for the robot's position \vec{x}_R directly, and a suitable expression for $\alpha(t)$ needs to be found independently. In order to allow this distinction, the trajectory will be treated as generic in the following sections, likewise the subscripts for the robot or the ball are omitted.

3.4 Piecewise Polynomial Approach

In the parameterization of the two dimensional trajectory according to the approach discussed in this section, the coordinates x and y are expressed as piecewise polynomials in powers of time t . This is a very simple and straightforward parameterization and it allows a direct computation of the leading coefficients $c_{k,i}$. In this parameterization, n_i is the number of intervals, $n_{c,i}$ is the number of coefficients for each coordinate in interval i , and n_p is the number of control points. As each interval is located between two adjacent control points, $n_i = n_p - 1$. The control points have the coordinates (x_p, y_p) , they are located on the trajectory. The piecewise polynomials are defined in intervals between two subsequent control points:

order to find a root of the residual, this is the case when the acceleration constraint for the robot is met exactly once in each interval. The velocity constraint is neglected in this preliminary parameterization. This procedure will result in a minimal end time t_{n_p-1} .

The piecewise polynomial approach has several advantages, the most outstanding of which is the simplicity of equations (3.17). They allow a direct solving for the coefficients $c_{k,i}$. However, it creates a large number of coefficients to be solved for in each iteration which requires some computational effort and time. It turned out that the preliminary approach presented here is not capable of finding a solution to certain arrangements of control points and correlated trajectories. The *eight* sample course used to test the trajectory algorithm in chapter 6 can not be generated satisfactorily. The reason for this is that for a varying end time not only the velocity of an object along the trajectory changes, but also the shape of the trajectory itself. If the end time is decreased for example, the trajectory might warp between two control points and grow longer in such a way that either a minimization of the end time while meeting all acceleration constraints is impossible or the shape of the resulting trajectory is not acceptable. Obviously, the parameterization according to equations (3.17) was too rigorous and limited the function space too severely. For that reason, the piecewise polynomial approach was dropped and no longer pursued at an early stage. Its basic concept with the polynomial representation of the trajectory coordinates and the method of computing the leading coefficients however was carried over to the minimal curve length approach discussed in the subsequent section.

3.5 Minimal Curve Length Approach

The second parameterization approach is an improvement of the piecewise polynomial approach. It utilizes all its advantages and avoids the primary weakness by decoupling the trajectory shape from the end time. This is achieved by constructing the trajectory in two distinct steps. In the first step, the mere shape of the trajectory is determined. There is no time dependence. The result of the first step only satisfies the condition to pass all control points. Again, the trajectory curve is defined explicitly and analytically by polynomials, but it is defined in a single interval from start to end. Therefore, there is no longer the need to meet transition conditions for the derivatives of the trajectory coordinates. The optimum shape of the trajectory is obtained with two objectives, minimization of the curve length (MCL) and minimization of its curvature in an integral sense. The second step involves time dependency of the motion along the trajectory. It minimizes

the arrival time while satisfying both velocity and acceleration constraints. Table 3.1 summarizes the steps in which a trajectory is created.

<p>Step 1: Trajectory Shape for ball</p> <ul style="list-style-type: none"> • no time dependency • pass through control points • determine optimum shape <p>Step 2: Trajectory Speed for ball</p> <ul style="list-style-type: none"> • time dependency • minimize end time • satisfy velocity and acceleration constraints <p>Step 3: Robot Trajectory</p> <ul style="list-style-type: none"> • time dependency • satisfy dribbling constraints

Table 3.1: MCL approach: Trajectory design in three steps

3.5.1 Trajectory Shape

This section describes the first step of the parameterization approach according to table 3.1. Figure 3.3 illustrates this step with an example of four control points. These are provided in global coordinates (x_p, y_p) with $p=0, \dots, n_p-1$. The number of control points is n_p , in this example $n_p=4$. The trajectory starts at point 0, passes the intermediate points 1 and 2 in ascending order, and ends at point $n_p-1=3$. It is parameterized in a single interval, the independent variable along the curve is r which ranges from $r_0=0$ at the start to $r_{n_p-1}=1$ at the end. Control point p is passed for a value $r=r_p$. The coordinate functions x and y are parameterized as polynomials of equal order with powers of r , the number of coefficients for the parameterizations of x and y is n_c each.

$$x(r) \stackrel{def}{=} \sum_{k=0}^{n_c-1} c_k \cdot r^k \quad , \quad 0 \leq r \leq 1 \quad (3.21)$$

$$y(r) \stackrel{def}{=} \sum_{k=0}^{n_c-1} c_{n_c+k} \cdot r^k \quad , \quad 0 \leq r \leq 1 \quad (3.22)$$

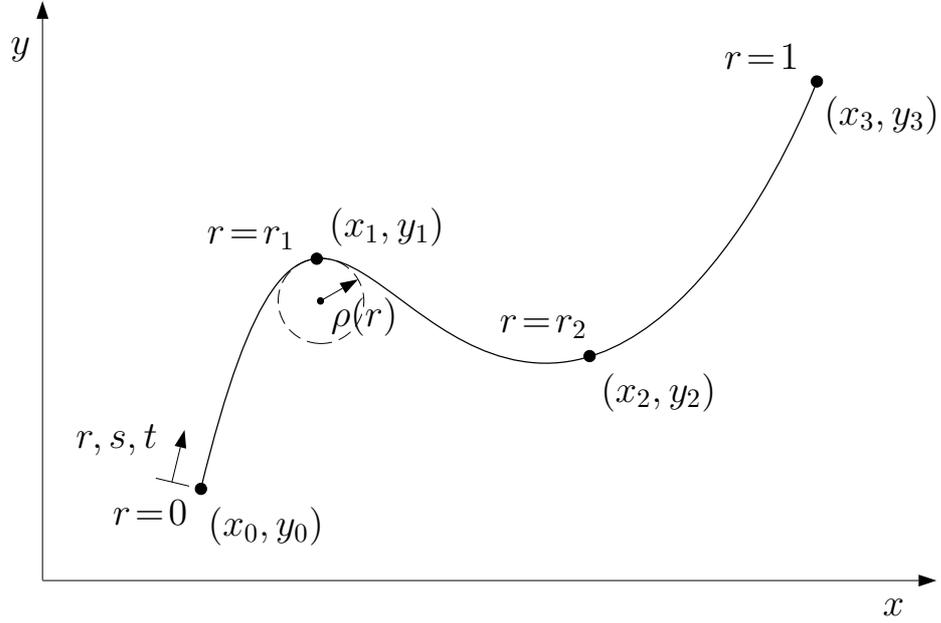


Figure 3.3: MCL approach: Trajectory through control points

Determining the coefficients c_k with $k=0, \dots, 2n_c-1$ such that the trajectory passes the control points for given values of r_p is the first task:

$$\mathbf{x}(r_p) = x_p \quad , \quad y(r_p) = y_p \quad \forall \quad p=0, \dots, n_p-1 \quad (3.23)$$

If there are no boundary conditions on the trajectory's direction or curvature at start or end, the number of coefficients required equals the number of control points, $n_c = n_p$.

The second task is to adjust the values r_p in such a way that the trajectory meets a criterion for optimality. Two criteria are considered, minimization of curve length and minimization of integral curvature. These criteria are described in more detail later in this section. The optimization can not be performed in a single step. Therefore, the two tasks need to be iterated in order to approach the optimal solution.

The first task is solved as follows. The n_p equations (3.21) for the x -coordinate of all control points are

$$\sum_{k=0}^{n_c-1} c_k \cdot r_p^k = x_p \quad , \quad p=0, \dots, n_p-1. \quad (3.24)$$

They can be written in matrix form

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & r_1 & r_1^2 & r_1^3 & \dots & r_1^{n_c-1} \\ 1 & r_2 & r_2^2 & r_2^3 & \dots & r_2^{n_c-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{n_p-2} & r_{n_p-2}^2 & r_{n_p-2}^3 & \dots & r_{n_p-2}^{n_c-1} \\ 1 & 1 & 1 & 1 & \dots & 1 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n_c-1} \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n_p-2} \\ x_{n_p-1} \end{bmatrix} \quad (3.25)$$

and in index notation

$$R_{pk} \cdot c_k = x_p. \quad (3.26)$$

The same can be done for the y -coordinate separately. There is no coupling of the equations for x - and y -coordinates if there are no start or end conditions on the curve's direction or curvature. Equation (3.26) can be solved for c_k by inversion of matrix R_{pk} :

$$c_k = (R_{pk})^{-1} \cdot x_p = R_{kp}^{-1} \cdot x_p \quad (3.27)$$

For the iterative optimization of the curve shape in task two, the derivatives of c_k with respect to the solution vector r_p will be required:

$$\frac{dc_k}{dr_p} = \frac{dR_{ki}^{-1}}{dr_p} \cdot x_i \quad (3.28)$$

Here the derivative of the inverse R_{ki}^{-1} is obtained from the subsequent equation which can be derived from the equality $R_{pk} \cdot R_{ki}^{-1} = I_{pi}$, with I_{pi} being the identity matrix:

$$\frac{dR_{ki}^{-1}}{dr_p} = -R_{kj}^{-1} \cdot \frac{dR_{jl}}{dr_p} \cdot R_{li}^{-1} \quad (3.29)$$

The second task is to determine the solution vector r_p with $p=1, \dots, n_p-2$ such that a criterion for the optimal shape of the trajectory is met. As mentioned before, two criteria have been developed, they are combined as two contributions to the objective function z^* :

$$z^*(r_p) \stackrel{def}{=} (1-w^*) \cdot s^*(r_p) + w^* \cdot \kappa^*(r_p) \text{ is minimal} \quad (3.30)$$

This corresponds to the postulation that the derivatives of z^* with respect to r_p must be zero:

$$\frac{dz^*}{dr_p} \stackrel{!}{=} 0 \quad (3.31)$$

Both contributions to the objective function z^* are multiplied with weight factors $(1-w^*)$ and w^* , respectively. These are described at the end of this section. The first contribution to the objective function z^* is the curve length s^* . It is calculated by integration of the arc length s' over the trajectory:

$$s^* \stackrel{def}{=} \int_0^1 s'(r) dr \quad (3.32)$$

The integrand is expressed in terms of $x(r)$ and $y(r)$.

$$s' = \sqrt{(x')^2 + (y')^2} \quad (3.33)$$

Note that the prime indicates a derivative with respect to the free variable r :

$$\square' \stackrel{def}{=} \frac{d\square}{dr}, \quad \square'' \stackrel{def}{=} \frac{d^2\square}{dr^2} \quad \forall \quad \square \in \{x, y, s\}$$

The second contribution to the objective function has been added in order to avoid bends with a small curve radius because they will reduce the velocity due to large centripetal forces. A so called curvature penalty term κ^* is introduced that is constituted by the integral of the trajectory's squared curvature from start to end:

$$\kappa^* \stackrel{def}{=} \frac{1}{2} \int_0^1 \kappa^2(r) dr \quad (3.34)$$

The integrand contains the curvature κ as a function of $x(r)$ and $y(r)$.

$$\kappa = \frac{x'y'' - x''y'}{(s')^3} \quad (3.35)$$

Both equations (3.32) and (3.34) need to be integrated numerically. A second order approximation scheme has been implemented that uses a large number of integration points. This is elaborated in appendix B.2.

The objective function z^* is a function of r_p because $x(r) = x(c_k(r_p), r)$. Table 3.2 summarizes the complete algorithm for computing the trajectory shape. It starts with the initialization of the solution vector $r_p^{(0)}$. The values $r_p^{(0)}$ for the intermediate control points may be picked arbitrarily but in ascending order. The values of r_p are initialized equally spaced, $r_p^{(0)} = \frac{p}{n_p-1}$ for $p = 1, \dots, n_p-2$. Then the loop over the iterative search for the optimal set of r_p begins. The matrix R_{jl} is assembled with the current values of r_p .

After inversion of R_{jl} , the coefficients $c_k(r_p)$ are solved for. They are functions of r_p . Using these coefficients, the objective function and its derivatives with respect to r_p are obtained by integration over the whole curve from $r=0$ to $r=1$. A complete description of the numerical treatment of the derivatives with respect to r_p can be found in appendix B.1. When the algorithm has converged, the iteration loop can be terminated. Otherwise, the solution vector r_p is adjusted according to the Newton method which is applied to solve equation (3.31). With the new set of r_p , the loop is started over by assembling the R-matrix again.

The weight factor w^* in equation (3.30) should range from 0 for pure curve length minimization to 0.5 for taking the curvature penalty into account. It should not exceed a value of 0.5 because various examples have shown poor or no convergence if the curvature penalty term is emphasized too heavily. This is due to the fact that the smallest curvature is obtained with the greatest curve radii which will blow up the trajectory curve to great spirals through the control points, a behavior that is not desired at all.

A better convergence is achieved by computing the trajectory shape in two passes of the algorithm presented in table 3.2. The first pass should only minimize the curve length. This is achieved by setting w^* to zero. This objective has a strong convergence towards a stable solution which is necessary because the initial guess of the solution vector $r_p^{(0)}$ will always be very rough. The solution r_p of the first pass can be used to initialize the second pass which will include the curvature penalty term by setting w^* to a value between 0 and 0.5.

<pre> initialize $r_p^{(0)}$ LOOP over iterations (m) assemble R-Matrix: $R_{jl}, \frac{dR_{jl}}{dr_p}, \frac{d^2R_{jl}}{dr_p dr_q}$ solve for coefficients: $c_k, \frac{dc_k}{dr_p}, \frac{d^2c_k}{dr_p dr_q}$ compute objective function: $z^*, \frac{dz^*}{dr_p}, \frac{d^2z^*}{dr_p dr_q}$ check convergence: $\left \frac{dz^*}{dr_p} \right < \text{Tol} ?$ perform Newton step: $r_p^{(m+1)} = r_p^{(m)} - \left[\left(\frac{d^2z^*}{dr_q dr_p} \right)^{-1} \cdot \frac{dz^*}{dr_q} \right]_{r_p^{(m)}}$ END LOOP over iterations (m) </pre>

Table 3.2: MCL approach: Trajectory shape algorithm

3.5.2 Trajectory Speed

The second step of the parameterization approach according to table 3.1 is the introduction of a time dependency of the object moving along the trajectory. It is repeated here that this object is either the robot in case of pure robot motion or the ball in case of the Two Body Problem.

An explicit time dependency is expressed for the distance along the curve as $s=s(t)$, rather than for the independent curve parameter r . This formulation has the advantage that the derivatives of s with respect to time, velocity and acceleration, can be obtained directly. The disadvantage of this approach is that now there exist two explicit parameterizations of the distance along the curve, $s=s(r)$ and $s=s(t)$. The coordinates x and y as functions of t can only be obtained in an implicit way, this is described at the end of this section.

For the parameterization of the distance along the curve $s(t)$ as a function of time, several issues need to be taken into consideration. First of all, it is not reasonable for the ball to move in reverse direction along the trajectory because this will be detrimental to the goal of minimizing the end time. Second, a deceleration of the ball greater than that imposed by its natural damping will force the robot to travel from a position behind the ball to a position in front of it. In order to do so, the robot has to go all the way around the ball, and after the deceleration maneuver it has to go back to its position behind the ball in order to accelerate again. If the total length of the trajectory is short compared to the dimensions of the robot, the maneuver described above can consume more time than travelling in a slower pace all the time. This is the case for the RoboCup competition where the length of the soccer field is approximately 10 m, this is about 20 times the robot diameter. Therefore, the maximum trajectory length will be relatively small compared to the robot size. The chosen parameterization of $s(t)$ rules out the danger of both travelling backwards and decelerating because its first derivative inherently increases monotonically for increasing time. It is defined in equation (3.36). There are two parameters, τ and v_∞ :

$$s(t) \stackrel{def}{=} v_\infty \frac{t^2}{\tau + t} \quad (3.36)$$

$$\dot{s}(t) = v_\infty \left[\frac{2t}{\tau + t} - \frac{t^2}{(\tau + t)^2} \right] \quad (3.37)$$

$$\ddot{s}(t) = 2v_\infty \left[\frac{1}{\tau + t} - \frac{2t}{(\tau + t)^2} + \frac{t^2}{(\tau + t)^3} \right] \quad (3.38)$$

Evaluating equation (3.36) at the end time t^* with the total trajectory curve length s^* and solving for t^* leads to the following expression for the

end time t^* . It is a function of the two parameters τ and v_∞ :

$$t^* = \frac{s^*}{v_\infty} \left[\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{\tau v_\infty}{s^*}} \right] \quad (3.39)$$

The properties of the parameterization are illustrated in figure 3.4 dimensionless. The time t on the abscissa and all values on the ordinate are scaled by v_∞ and powers of τ . The thin solid line represents the distance along the trajectory $s(t)$, it asymptotically approaches the thick solid line $\frac{t}{\tau} - 1$ representing uniform motion. The thin dashed line displays the velocity $\dot{s}(t)$. Starting from zero, it asymptotically approaches a constant velocity v_∞ plotted as a thick dashed line. The velocity increases monotonically, this is independent of the choice of the parameter τ . The acceleration profile in tangential direction $\ddot{s}(t)$ is plotted as a dash-dotted line. The ball's acceleration starts with a maximum value of $\ddot{s}(t) = \frac{2v_\infty}{\tau}$ and asymptotically approaches zero as time proceeds.

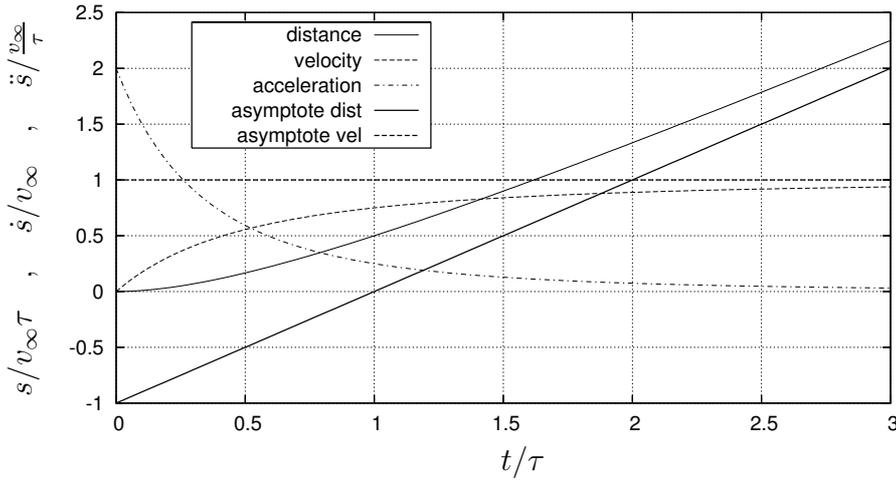


Figure 3.4: MCL approach: Velocity profile along trajectory

The choice of the parameter τ depends on the dynamic constraints of the robot's performance. A thorough examination of the true constraints imposed by the robot's actuators as described in chapter 2 requires the following steps: First, the robot's trajectory needs to be determined according to section 3.5.3. Second, the wheels' angular speeds can be determined with equations (1.8) and (1.9), p. 16. The wheel torques can be computed by equation (1.51), p. 22. Due to the high degree of nonlinearity of the equations used in the two steps, it is necessary to use an iterative process to determine the set of τ and v_∞ that minimize the end time. In order to avoid this, a

much simpler approach has been developed which seems to be a reasonable tradeoff.

In the approach that has been implemented, not the robot's capabilities are constrained, but the motion of the ball. This rids the problem both of the complicated conversion from the ball's to the robot's trajectory, and of the robot specific three wheeled geometry. The ball's motion is confined by a maximum velocity v_{\max} and two maximum accelerations, the lateral acceleration a_{lat} and the longitudinal acceleration a_{acc} :

$$0 \leq v(t) = \dot{s}(t) \leq v_{\max} \quad (3.40)$$

$$0 \leq a_t(t) = \ddot{s}(t) \leq a_{\text{acc}} \quad (3.41)$$

$$0 \leq a_n(t) = \left| \frac{v^2(t)}{\rho(r(t))} \right| \leq a_{\text{lat}} \quad (3.42)$$

In these constraints, $v(t)$ is the velocity along the curve, a_t and a_n are the accelerations in tangential and normal directions, respectively. The centripetal acceleration a_n is always perpendicular to the trajectory and is determined by the velocity $v(t)$ and the local curve radius $\rho(r(t))$.

Dividing the acceleration into its two components has a practical reason. The ball is assumed to start its motion with zero velocity and high acceleration a_t in tangential direction. In the beginning, the acceleration a_n perpendicular to the course can be neglected due to the small velocity. Then the tangential acceleration decreases while the ball gains velocity and perpendicular accelerations become important in curves. The two acceleration components are therefore never significant at the same time, in the beginning the tangential component dominates and the normal component is not significant while towards the end the situation reverses.

Equation (3.41) is used to determine the parameter τ for a maximum initial acceleration $a_t(0) = a_{\text{acc}}$:

$$a_t(0) = \ddot{s}(0) = v_{\infty} \frac{2}{\tau} \quad (3.43)$$

$$\tau = \frac{2 v_{\infty}}{a_{\text{acc}}} \quad (3.44)$$

The second parameter, the asymptotic velocity v_{∞} , must satisfy both equations (3.40) and (3.42). At this point, neither of the equations can be solved. Equation (3.40) can not be solved because the greatest velocity is reached at the end of the course, $\dot{s}(t^*)$, but the end time t^* is not known yet.

Therefore, the asymptotic velocity v_∞ is used as a conservative approximation instead of equation (3.40):

$$v_\infty \leq v_{\max} \quad (3.45)$$

Equation (3.42) cannot be solved because there is no correlation yet between the time t and the trajectory parameter r . Therefore, it is not possible to relate a velocity $\dot{s}(t)$ to its corresponding curve radius $\rho(r(t))$. As a remedy, the smallest curve radius $\hat{\rho}$ of the complete trajectory and the greatest velocity v_∞ are used in equation (3.42) rather than the actual values. This is a very conservative approach:

$$\frac{(v_\infty)^2}{\hat{\rho}} \leq a_{\text{lat}} \quad (3.46)$$

The minimum curve radius is determined as

$$\hat{\rho} \stackrel{\text{def}}{=} \min_{r \in [0,1]} (\rho(r)) \quad (3.47)$$

in which the curve radius $\rho(r)$ is defined as the inverse of the curvature κ defined in equation (3.35):

$$\rho(r) = \frac{\left(\sqrt{(x')^2 + (y')^2}\right)^3}{x'y'' - x''y'} \quad (3.48)$$

Solving equation (3.46) for v_∞ yields

$$v(t) \leq \sqrt{\hat{\rho} \cdot a_{\text{lat}}} \quad (3.49)$$

Combining equations (3.45) and (3.49), v_∞ can be obtained as the smaller value out of the set of two:

$$v_\infty = \min\left(\left\{v_{\max}, \sqrt{\hat{\rho} \cdot a_{\text{lat}}}\right\}\right) \quad (3.50)$$

Now that the parameters for the velocity profile along the trajectory have been determined, the position of the ball can be expressed in terms of global coordinates as functions of time, $x(t)$ and $y(t)$, including their derivatives with respect to time.

$$x(t) = \mathbf{x}(r(s(t))) \quad (3.51)$$

The function $r(s)$ is the inverse of the function $\mathbf{s}(r)$:

$$r(s) = \mathbf{s}^{-1}(r) \quad (3.52)$$

At this point, it is no longer possible to provide an explicit expression because $r(s)$ cannot be solved for explicitly. The problem is treated numerically from this point on by defining a set of closely spaced sampling times t_s . For these times t_s , the distance s_s along the trajectory can be determined with equation (3.36):

$$s_s = s(t_s) = v_\infty \frac{t_s^2}{\tau + t_s} \quad (3.53)$$

The corresponding values of r_s can be obtained by solving the equation

$$s(r_s) = s_s = \int_0^{r_s} s'(r) dr \quad (3.54)$$

for each sample instance s_s . This is implemented by creating a lookup table with equally spaced values of r_l and corresponding values $s(r_l)$. This lookup table is created in the course of the numerical integration of equation (3.32). It is used to interpolate between two of its entries. This procedure is elaborated in appendix B.2. The trajectory coordinates x and y at the sample times t_s are then obtained with the values of r_s :

$$x(t_s) = x(r_s) \quad (3.55)$$

The first derivative of x and y with respect to time can be computed as follows:

$$\frac{dx}{dt} = \frac{dx}{dr} \frac{dr}{ds} \frac{ds}{dt} \quad (3.56)$$

or

$$\dot{x}(t_s) = x'(r_s) (s'(r_s))^{-1} \dot{s}(t_s) \quad (3.57)$$

The same can be done for the y -coordinate. For a complete derivation of expressions for all required derivatives with respect to time, refer to appendix B.2. Table 3.3 summarizes the steps presented in this section.

3.5.3 Robot Trajectory

This section describes how the robot's trajectory is obtained from a given trajectory of the ball according to section 3.2. First, the robot's orientation α is determined according to equation (3.16) for each sample time t_s .

$$\tan(\alpha(t_s)) = \frac{\ddot{y}_B(t_s) + \delta \dot{y}_B(t_s)}{\ddot{x}_B(t_s) + \delta \dot{x}_B(t_s)} \quad (3.58)$$

```

compute curve speed coefficient  $\tau$ 
determine smallest curve radius  $\hat{\rho}$ 
compute maximum velocity  $v_\infty$ 
compute end time  $t^*$ 
compute distance along curve at sample times  $s_s = s(t_s)$ 
determine respective values of  $r_s = r(s_s)$ 
compute auxiliaries for derivatives
compute  $x(t_s) = x(r_s)$ ,  $y(t_s) = y(r_s)$  and derivatives

```

Table 3.3: MCL approach: Trajectory speed algorithm

Appendix B.3 elaborates the details of the computation of α and its derivatives with respect to time. The next step is the robot's position according to equation (3.14) which is a function of the ball's position, velocity, acceleration, and the orientation α .

$$\begin{aligned}
\vec{x}_R &= \vec{x}_B + \frac{M_B}{k} \left(\ddot{\vec{x}}_B + \delta \dot{\vec{x}}_B \right) - \vec{x}_0(\alpha) \\
\dot{\vec{x}}_R &= \dot{\vec{x}}_B + \frac{M_B}{k} \left(\dot{\ddot{\vec{x}}}_B + \delta \ddot{\vec{x}}_B \right) - \dot{\vec{x}}_0(\alpha) \\
\ddot{\vec{x}}_R &= \ddot{\vec{x}}_B + \frac{M_B}{k} \left(\ddot{\dot{\vec{x}}}_B + \delta \dot{\ddot{\vec{x}}}_B \right) - \ddot{\vec{x}}_0(\alpha)
\end{aligned} \tag{3.59}$$

The robot's acceleration $\ddot{\vec{x}}_R$ is needed to check if the torques required at each wheel can be accomplished by the motors. In the above formulation, the fourth derivative of the ball's position with respect to time $\ddot{\dot{\vec{x}}}_B$ needs to be computed for the robot's acceleration.

Assuming very stiff springs $k \rightarrow \infty$ for the robot trajectory design simplifies these equations significantly and reduces the highest derivative with respect to time to the order of two. Note that this does not affect the robot's orientation which is still determined with equation (3.58). The robot trajectory at the sample times t_s is now computed as:

$$\begin{aligned}
\vec{x}_R(t_s) &= \vec{x}_B(t_s) - \vec{x}_0(\alpha(t_s)) \\
\dot{\vec{x}}_R(t_s) &= \dot{\vec{x}}_B(t_s) - \dot{\vec{x}}_0(\alpha(t_s), \dot{\alpha}(t_s)) \\
\ddot{\vec{x}}_R(t_s) &= \ddot{\vec{x}}_B(t_s) - \ddot{\vec{x}}_0(\alpha(t_s), \dot{\alpha}(t_s), \ddot{\alpha}(t_s))
\end{aligned} \tag{3.60}$$

A second modification introduces a new degree of freedom which is useful for the experimental validation as it offers another parameter to play with. It is based on the concept of a generic trajectory that is pursued more strictly

in the derivation of the next method in section 3.6. Assume now that the trajectory derived in the previous two sections is not the trajectory for the ball but it is treated more generally as a generic trajectory \vec{x}_T . That means it is only used as a guideline to construct both trajectories for robot and ball. One method to do this is introducing a weight factor ψ that determines the location of the robot and ball trajectory relative to \vec{x}_T :

$$\vec{x}_R = \vec{x}_T - \psi \vec{x}_0(\alpha) \quad (3.61)$$

$$\vec{x}_B = \vec{x}_T + (1 - \psi) \vec{x}_0(\alpha) \quad (3.62)$$

Again, the robot's orientation is determined with equation (3.58). For a weight factor $\psi=1$, nothing has changed. The generic trajectory coincides with the one for the ball.

3.6 Circle and Tangent Approach

Both previously described polynomial approaches required to invert a matrix in order to solve for the leading coefficients. In addition to that, both approaches required an iterative procedure to meet a given design criterion. The third trajectory generation approach that is presented in this section aims to cut down the computational effort even more. This can only be obtained by reducing the restrictions on the trajectory's properties. The simplification of an infinitely stiff spring according to equations (3.60) that was already used in the minimal curve length approach already reduces the need for the higher order derivatives of the ball's position. Some more rigorous simplifications will be applied in this third approach to allow an even more straightforward computation of the trajectory:

The shape of the trajectory is assembled geometrically with circular arcs and straight tangent lines, hence the name 'circle and tangent' (CAT) approach. The major drawbacks of this method are the unavoidable discontinuities at the transition points between straight lines and circular arcs. At these points there is a change in curvature which goes along with a step in the acceleration of a point moving along the trajectory, disregarding of its velocity. This automatically leads to a discontinuity in the robot orientation according to equation (3.16). A method to handle this issue by smoothing the transitions is described in section 3.6.3. With these modifications the trajectory no longer fulfills any of the equations (3.14) and (3.16) representing the actual kinematic situation. The trajectory will provide a very close approximation though which allows a new design degree of freedom that was not utilized before: In the polynomial trajectory approaches, the trajectory was always computed for the ball in the case of dribbling; the robot trajectory

was derived in a subsequent step. Now it is possible to treat the trajectory as a generic constructor or a guideline to derive both the ball's and the robot's course.

3.6.1 Trajectory Shape

As mentioned above, the trajectory according to the approach presented in this section is assembled with two geometric elements: circular arcs and straight lines. It is assumed here that the control points are provided as the centers of discrete, distributed, non-coherent obstacles that are to be avoided. In addition to that, the chronological order and the orientation in which each obstacle has to be surrounded is provided. The trajectory is now created as the union of circular arcs around the control points and the straight tangent lines connecting the arcs in the respective directions. This procedure is adapted to the obstacle avoidance requirement in the RoboCup competition because the centers of the opposing robots can be chosen as control points.

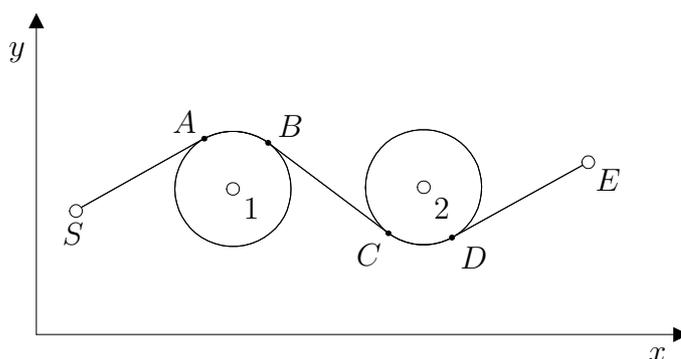


Figure 3.5: CAT approach: Trajectory shape

Figure 3.5 depicts this approach. The required information to create the trajectory qualitatively reads as:

- Start at point S
- Surround object 1 in clockwise direction
- Surround object 2 in anticlockwise direction
- Head for point E

The points S , 1, 2, and E are the control points, they are plotted as empty circles in figure 3.5. The tangent points A , B , C , and D separate

the straight and the circular sections, they are plotted as filled dots. For a detailed description of the computation of the tangent point coordinates refer to appendix B.4.

3.6.2 Trajectory Speed

As discussed above, the continuity restrictions imposed by equations (3.14) and (3.16) are already violated by the trajectory shape. Therefore, no special attention needs to be paid to continuity issues as far as the velocity along the trajectory is concerned. Thus, a piecewise constant acceleration is assumed because, based on the physical restrictions of the robot hardware, this approach will yield the shortest travelling time from start to end. The same constraints that confined the ball's motion in section 3.5.2 can be applied to the generic trajectory here, compare equations (3.40) to (3.42):

$$0 \leq v(t) \leq v_{\max} \quad (3.63)$$

$$-a_{\text{dec}} \leq a_t(t) \leq a_{\text{acc}} \quad (3.64)$$

$$0 \leq a_n(t) \leq a_{\text{lat}} \quad (3.65)$$

In analogy to the MCL approach discussed in section 3.5, motion is confined by a maximum velocity v_{\max} , a maximum lateral acceleration a_{lat} , and the maximum longitudinal acceleration a_{acc} . New in this approach is that deceleration is permitted, this requires a new confining parameter, the maximum longitudinal deceleration $a_{\text{dec}} > 0$. The lateral acceleration constraint in equation (3.65) can be converted into a velocity constraint with the respective circle radius r for each circular section:

$$v_{\text{lat}}^2 = a_{\text{lat}} \cdot |r| \quad (3.66)$$

In straight sections, the lateral acceleration constraint has no effect. The three construction rules for the velocity profile are simple:

1. Accelerate with constant acceleration a_{acc} whenever possible
2. Decelerate with constant deceleration a_{dec} whenever necessary such that no constraint will be violated in the future
3. Maintain velocity otherwise

The procedure with piecewise constant accelerations can be illustrated in the velocity squared–distance diagram as straight lines. The derivation is

very short: Given a piecewise constant acceleration

$$a = \frac{dv}{dt} = \text{const} \quad (3.67)$$

and the definition of velocity

$$v = \frac{ds}{dt} \quad (3.68)$$

one can solve for the time increment

$$dt = \frac{dv}{a} = \frac{ds}{v} \quad (3.69)$$

which can be rewritten as

$$v dv = a ds . \quad (3.70)$$

Integration of equation (3.70) yields:

$$v^2(\tilde{s}) = 2a(\tilde{s} - s_0) + v_0^2 . \quad (3.71)$$

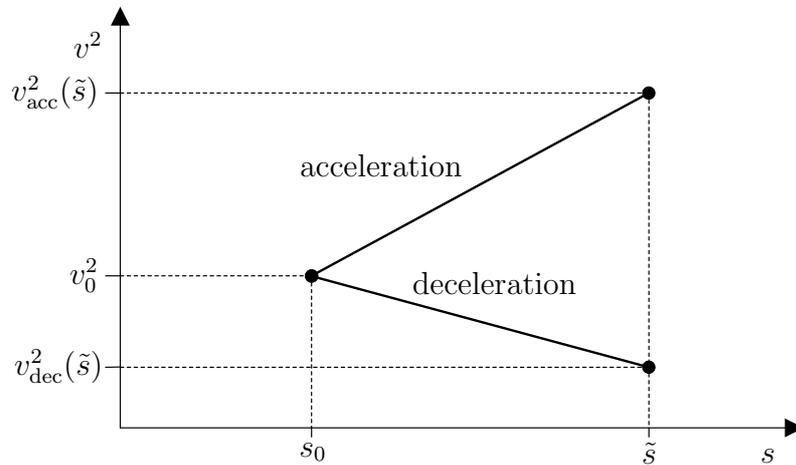


Figure 3.6: CAT approach: Velocity²-distance diagram

The graphical interpretation is illustrated in figure 3.6. Starting at a distance s_0 with velocity v_0 , the squared velocity is a linear function of the travelled distance s . For positive values of a , the velocity increases (acceleration), for negative values of a , the velocity decreases (deceleration).

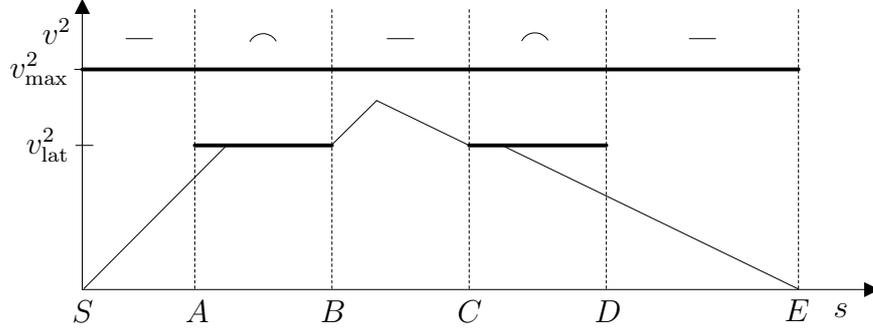


Figure 3.7: CAT approach: Trajectory speed

For the simple example in figure 3.5, the velocity profile could look like the one depicted in figure 3.7. The diagram is divided into five alternating straight and circular sections. For all sections, the velocity constraint v_{\max}^2 according to equation (3.63) is binding. For the two circular sections between points A and B and between C and D , the lateral acceleration additionally constrains the velocity to v_{lat} such that equation (3.65) is not violated. Note that in this example both circles have the same radius, this results in the same velocity constraint v_{lat}^2 for both circular sections. Typically, this constraint is harder than the maximum velocity constraint. In the example depicted in figure 3.7, both start and end velocities are zero. In the first section, rule 1 is applied to accelerate throughout the complete section. In the second section, rule 1 is applied again until rule 3 is applied to meet the constraint v_{lat}^2 in the remainder of the section. In the third section, rule 1 is applied to accelerate, then rule 2 is applied to meet the constraint v_{lat}^2 in point C . In the fourth section, rule 3 is applied first until rule 2 initiates the deceleration phase that persists throughout the last section in order to stop at the end. Appendix B.5 elaborates the details of this procedure.

3.6.3 Robot Trajectory

As mentioned above, the trajectory created in the previous two sections is generic. This means that either robot or ball or a point between both can be located on the generic trajectory because it is only a guideline. The robot location \vec{x}_R and the ball location \vec{x}_B relative to the generic trajectory \vec{x}_T are determined with the following equations:

$$\vec{x}_R = \vec{x}_T - \psi \vec{x}_0(\alpha) \quad (3.72)$$

$$\vec{x}_B = \vec{x}_T + (1 - \psi) \vec{x}_0(\alpha) \quad (3.73)$$

They require a weight factor ψ and the ball neutral position \vec{x}_0 defined in equation (3.11). This is analog to the second modification to the MCL trajectory according to equations (3.61) and (3.62). For a factor $\psi=0$ the robot trajectory is identical with the generic trajectory, the ball will be located inside the generic trajectory in curved sections. For $\psi=1$ the ball trajectory will coincide with the generic trajectory, the robot will be located outside in curved sections:

For the calculation of the robot orientation α , the secant method has been developed: For a given point \vec{x}_T on the generic trajectory, the robot orientation α is obtained as the orientation of the secant between the respective point \vec{x}_T and a point \vec{x}_S that is also located on the trajectory ahead by a given arc length \bar{s} . The arc length \bar{s} is chosen such that equation (3.16) is satisfied for the given circle radius and velocity.

The secant method starts with the assumption that in a circular section the velocity is constant. This assumption is easily justified because typically, in circular sections the lateral acceleration constraint is limiting the velocity. For maximum velocity and acceleration values of 5 m/s and 5 m/s² that were used for motor dimensioning in chapter 2, the critical circle radius is 1 m. It is foreseen that all obstacles will have a smaller radius and thus the acceleration limit will be active in circular sections rather than the velocity limit.

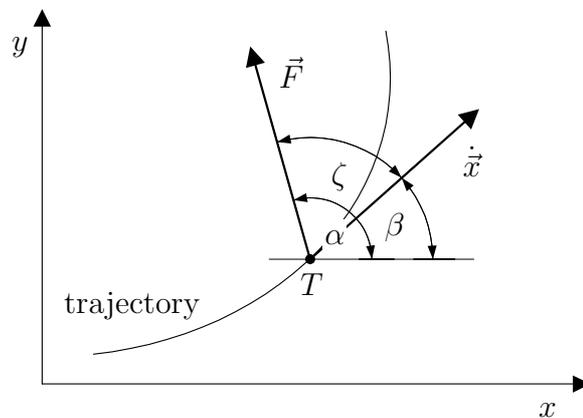


Figure 3.8: CAT approach: Angle geometry for arbitrary trajectory

Figure 3.8 illustrates the geometry of the various angles involved for an arbitrary trajectory. The velocity vector \dot{x} is tangent to the trajectory in point T . It has an angle β with the x -axis that is defined in equation (3.75). The vector \vec{F} is the force vector according to equation (3.9). It has an angle α with the x -axis because it is pursued to orient the robot in direction of \vec{F}

according to equation (3.16). If the trajectory in figure 3.8 represents the ball's motion, equation (3.16) can be formulated for the trajectory:

$$\tan(\alpha) = \frac{\ddot{y} + \delta\ddot{y}}{\ddot{x} + \delta\ddot{x}} \quad (3.74)$$

$$\tan(\beta) = \frac{\dot{y}}{\dot{x}} \quad (3.75)$$

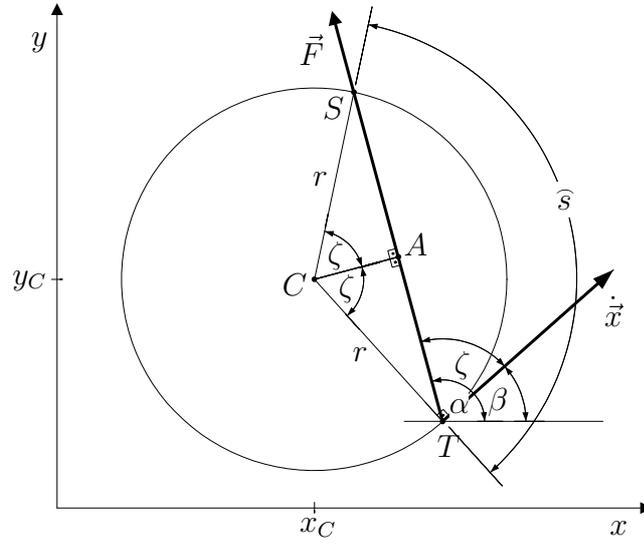


Figure 3.9: CAT approach: Secant geometry for circular sections

Figure 3.9 illustrates the geometry of the above mentioned angles and the secant for the special case of anticlockwise motion along the circular section of a trajectory with radius r and central point C . Starting at point T located on the trajectory, the force vector \vec{F} intersects the circle in the point S ahead of T . The connecting straight line \overline{TS} constitutes the secant after which this method is named. The circular arc length $\widehat{s} = |\widehat{TS}|$ is used for the calculation of the coordinates of point S :

$$\widehat{s} = 2\zeta r \quad (3.76)$$

The angle between $\dot{\vec{x}}$ and \vec{F} is $\zeta = \alpha - \beta$, it can be found in both triangles CTA and CSA . Equations (3.74) and (3.75) allow a substitution of the unknown angle ζ :

$$\tan(\zeta) = \tan(\alpha - \beta) = \frac{\dot{y}\dot{x} - \ddot{x}\dot{y}}{(\ddot{x}\dot{x} + \ddot{y}\dot{y}) + \delta(\dot{x}^2 + \dot{y}^2)} \quad (3.77)$$

Taking the circular motion into consideration by introducing polar coordinates, $x = x_C + r \sin(\beta)$ and $y = y_C - r \cos(\beta)$, this equation simplifies to:

$$\tan(\zeta) = \frac{\dot{\beta}}{\delta} \quad (3.78)$$

The angular speed of the circular motion $\dot{\beta}$ is determined by the lateral acceleration constraint $a_{\text{lat}} = \dot{\beta}^2 r$. Now the secant length can be expressed in terms of the parameters r , a_{lat} , and δ :

$$\widehat{s} = 2r \arctan\left(\frac{\sqrt{\frac{a_{\text{lat}}}{r}}}{\delta}\right) \quad (3.79)$$

The secant method is applied in the following way: It is assumed that the trajectory is provided in the form of a table containing closely spaced entries of time t , distance s , and x - and y -coordinates. For a given point T on the trajectory, the secant length is calculated according to equation (3.79). Point S is obtained by adding this length to the distance of the point T , $s_S = s_T + \widehat{s}$. The x - and y -coordinates of point S are obtained via linear interpolation of the distance s_S with the next two neighbors in the trajectory table. Finally, the angle α is calculated as

$$\alpha = \arctan\left(\frac{y_S - y_T}{x_S - x_T}\right). \quad (3.80)$$

Note that for the circular section, equation (3.74) yields the same angle α . However, this method would not account for the transitions between straight and circular sections. At least one of the robot and the ball trajectory would become discontinuous at these transitions, while the secant method produces relatively smooth continuous transitions.

Now the robot and ball positions can be calculated relative to the generic trajectory according to equations (3.72) and (3.73). The influence of the weight factor ψ and the damping ratio δ is illustrated in figure 3.10. As mentioned before, $\psi=0$ means robot, $\psi=1$ means ball is on the generic trajectory. For $\psi=0.5$, an intermediate point between robot and ball is on the generic trajectory. A damping ratio $\delta=0$ indicates no viscous friction for the ball; the robot is oriented towards the center of the actual circular section. A large value, e.g. $\delta=2\text{ s}^{-1}$, means viscous friction is dominant over centripetal forces. In that case, the robot is oriented more in the direction of its velocity.

The secant method has the following advantages:

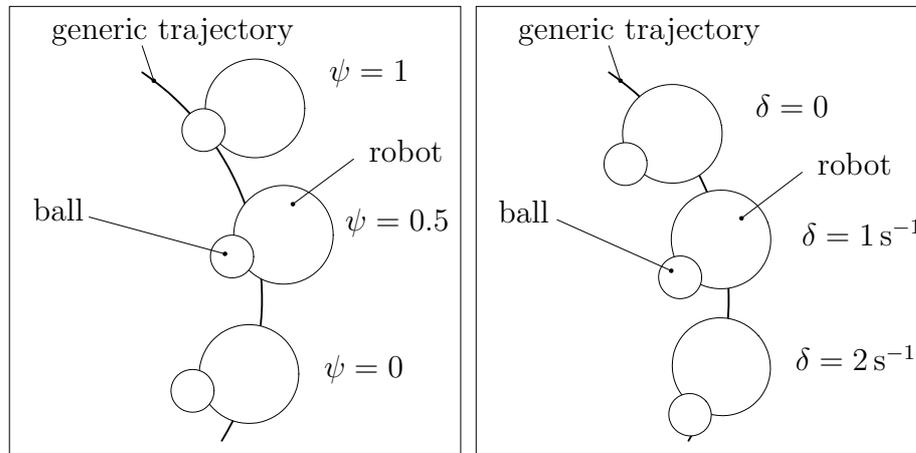


Figure 3.10: CAT approach: Weight factor ψ and damping ratio δ

- It precisely produces the angle according to equation (3.16) in a circular section with constant velocity if the ball is located on the generic trajectory
- It is precise in straight sections according to equation (3.16) when the robot is directly behind the ball
- It produces smooth transitions both from circular to straight and from straight to circular sections
- It is fast, robust, and simple to compute

Refer to appendix B.6 for more information on the numerical treatment of the computation of the robot position, velocity, and acceleration.

3.7 Weighted Polynomial Approximation

In both trajectory generation algorithms described in the previous sections the resulting robot trajectory is stored in form of a lookup table containing entries $[t_i, x_i, y_i, \alpha_i]$. These trajectory entries constitute the robot drivecycle. While the robot is driving, it interpolates the entries of this table to obtain its desired position, velocity, and acceleration according to the drivecycle. This section describes the method of interpolation, a weighted polynomial approximation (WPA).

Figure 3.11 illustrates how an interpolated position is obtained for the x -coordinate at an arbitrary time t^* using the entries x_i at times t_i provided by the drivecycle table. The procedure is described for the x -coordinate in

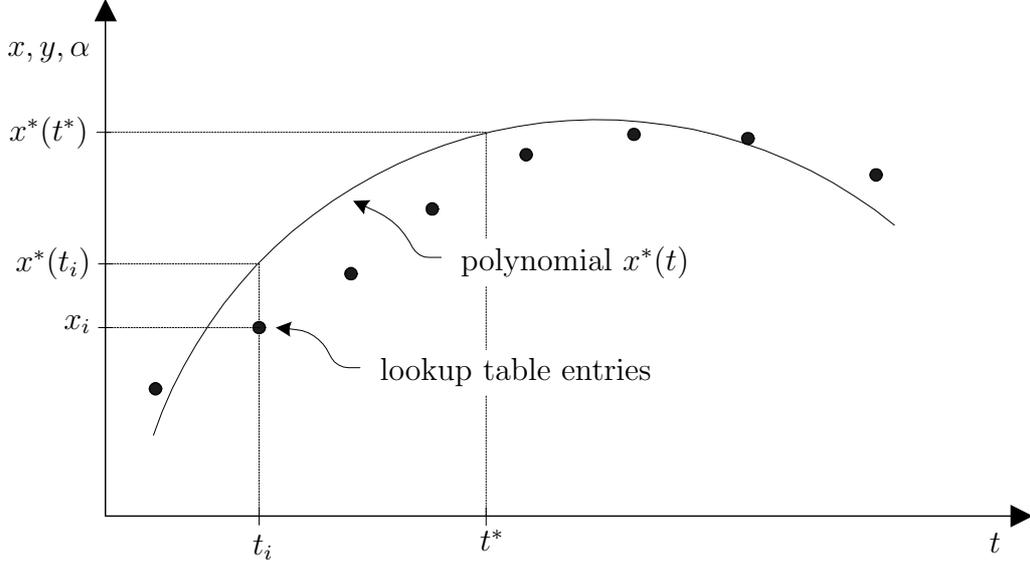


Figure 3.11: Weighted polynomial approximation

the following but the same method is applied to both y and α . Note that the angle α is typically clipped to an interval $[-\pi, \pi]$. Therefore, it is necessary to un-clip the angle prior to the polynomial fit to remove discontinuities.

The weighted polynomial approach is applied at discrete sampling times t^* to obtain the interpolated position x^* and its derivatives only for the given time t^* , not for any other time. The unity of all interpolated positions x^* constitutes the interpolated robot position $x(t)$ for the complete drivecycle:

$$x(t) \stackrel{def}{=} x^*(t^*) \quad (3.81)$$

At each time t^* , the weighted polynomial defined in equation (3.82) is computed in order to obtain the position and its derivatives at time t^* only. The polynomial order n_{WPA} must be at least 2 to be able to compute interpolated accelerations.

$$x^*(t) \stackrel{def}{=} \sum_{k=0}^{n_{\text{WPA}}} a_k^* \cdot t^k \quad (3.82)$$

$$\dot{x}^*(t) = \sum_{k=1}^{n_{\text{WPA}}} k a_k^* \cdot t^{k-1} \quad (3.83)$$

$$\ddot{x}^*(t) = \sum_{k=2}^{n_{\text{WPA}}} k(k-1) a_k^* \cdot t^{k-2} \quad (3.84)$$

Given n data points $[t_i, x_i]$, the weighted error e_i^* between the approximation polynomial x^* and the data point x_i is defined in equation (3.85):

$$e_i^* = w_i^* \cdot (x^*(t_i) - x_i) \quad (3.85)$$

$$w_i^* = w^*(t_i) \quad (3.86)$$

The weight factor w_i^* is determined for each data point individually utilizing a weight factor function $w^*(t)$ according to equation (3.86). The weight factor allows to emphasize the data points in the vicinity of time t^* and to create a smooth interpolation curve with little deviations from the original data points.

The polynomial coefficients a_k^* are obtained with a standard least error squares method. Equation (3.87) sums over the weighted errors of all data points. This sum J^* must be minimal for an optimal set of coefficients. The $n_{\text{WPA}}+1$ derivatives of J^* with respect to the coefficients must therefore be zero:

$$J^* \stackrel{\text{def}}{=} \frac{1}{2} \sum_{i=1}^n (e_i^*)^2 \stackrel{!}{=} \min \quad (3.87)$$

$$\frac{dJ^*}{da_k^*} = \sum_{i=1}^n e_i^* \cdot \frac{de_i^*}{da_k^*} \stackrel{!}{=} 0 \quad , \quad k = 0, \dots, n_{\text{WPA}} \quad (3.88)$$

The derivative of the weighted error at data point i with respect to coefficient number k is

$$\frac{de_i^*}{da_k^*} = w_i^* t_i^k \quad , \quad k = 0, \dots, n_{\text{WPA}}. \quad (3.89)$$

Writing the $n_{\text{WPA}}+1$ equations (3.88) in matrix form allows a simple formulation of the problem

$$\mathcal{T}_w \mathcal{T}_w^T \cdot \mathbf{a}_x - \mathcal{T}_w \cdot \mathbf{X}_w = 0 \quad (3.90)$$

which utilizes the following coefficient matrices:

$$\mathcal{T}_w \stackrel{\text{def}}{=} \begin{bmatrix} w_1^* \cdot 1 & w_1^* \cdot t_1 & w_1^* \cdot t_1^2 & \cdots & w_1^* \cdot t_1^{n_{\text{WPA}}} \\ w_2^* \cdot 1 & w_2^* \cdot t_2 & w_2^* \cdot t_2^2 & \cdots & w_2^* \cdot t_2^{n_{\text{WPA}}} \\ w_3^* \cdot 1 & w_3^* \cdot t_3 & w_3^* \cdot t_3^2 & \cdots & w_3^* \cdot t_3^{n_{\text{WPA}}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_n^* \cdot 1 & w_n^* \cdot t_n & w_n^* \cdot t_n^2 & \cdots & w_n^* \cdot t_n^{n_{\text{WPA}}} \end{bmatrix}^T \quad (3.91)$$

$$\mathbf{a}_x \stackrel{\text{def}}{=} \begin{bmatrix} a_0^* & a_1^* & a_2^* & \cdots & a_{n_{\text{WPA}}}^* \end{bmatrix}^T \quad (3.92)$$

$$\mathbf{X}_w \stackrel{\text{def}}{=} \begin{bmatrix} w_1^* x_1 & w_2^* x_2 & w_3^* x_3 & \cdots & w_n^* x_n \end{bmatrix}^T \quad (3.93)$$

The matrix $\mathcal{T}_w \mathcal{T}_w^T$ is quadratic and its inverse can be computed in order to solve equation (3.90) for the coefficient vector \mathbf{a}_x defined in equation (3.92). The computational effort can be optimized by computing the coefficients for the polynomials for x , y , and α at the same time. This can be achieved by replacing the weighted data vector \mathbf{X}_w according to equation (3.93) by the weighted data matrix \mathbf{XYA}_w which contains the trajectory information for all coordinates:

$$\mathbf{XYA}_w \stackrel{def}{=} \begin{bmatrix} w_1^* x_1 & w_1^* y_1 & w_1^* \alpha_1 \\ w_2^* x_2 & w_2^* y_2 & w_2^* \alpha_2 \\ w_3^* x_3 & w_3^* y_3 & w_3^* \alpha_3 \\ \vdots & \vdots & \vdots \\ w_n^* x_n & w_n^* y_n & w_n^* \alpha_n \end{bmatrix} \quad (3.94)$$

The polynomial coefficients for the interpolation at time t^* can finally be computed with equation (3.95). The coefficient vectors \mathbf{a}_y and \mathbf{a}_α constitute polynomials in analogy to equation (3.82):

$$[\mathbf{a}_x \ \mathbf{a}_y \ \mathbf{a}_\alpha] = \left(\mathcal{T}_w \mathcal{T}_w^T \right)^{-1} \cdot \mathcal{T}_w \cdot \mathbf{XYA}_w \quad (3.95)$$

In the current implementation of the weighted polynomial approximation routine the weight factor distribution according to equation (3.96) was successfully used. The weight factor is unity for the time t^* and decreases quadratically to both sides. The parameter f was set to a value of $f = 4 \text{ s}^{-2}$ which causes the weight factor w to reach a zero value at a difference of 250 ms from the relevant time t^* . The maximum number of data entries n used in equations (3.91) and (3.94) is 32 which is divided symmetrically into 16 entries to the left of t^* and 16 on the right side.

$$w^*(t_i) = \begin{cases} (1 - f|t^* - t_i|)^2 & \text{for } |t^* - t_i| < f^{-1} \\ 0 & \text{for } |t^* - t_i| \geq f^{-1} \end{cases} \quad (3.96)$$

Sections 4.2 and 4.3 describe how the weighted polynomial approximation method is used not only for interpolating the drivecycle table entries but also for extrapolating the calculated robot positions into the future.

Chapter 4

Controllers

This chapter describes the controller structure of the robot. It starts with the description of the control strategy of the digital servoamplifiers (DES) because these are determined by the supplier. Section 4.2 elaborates how the robot control scheme is set up to take optimal advantage of the DES capabilities. The last section describes the method of obtaining the robot position at the discrete control times by extrapolating the past video image robot positions.

4.1 Motor Controller

The robot's propulsion motors are controlled with digital servoamplifiers (DES). These are commercial products designed and optimized for the electric motors from the same supplier [51]. They take care of the electronic commutation of the motors, and they contain a speed and a current controller. The DES provide an analog interface as well as RS232 serial or Controller Area Network (CAN) protocols. They can be configured to operate in speed or current regulation mode.

Figure 4.1 illustrates the setup of a DES controller. A digital signal processor manages the controller's I/O functions and runs the software for the speed and current control algorithm. The controller is a cascade of functional blocks with a number of feedback loops. Starting at the end with the motor, each DES is connected to the three windings of one motor and supplies the electric power. A PWM generator triggers six MOSFETs that switch the supply voltage to each phase at a frequency of 50 kHz producing the voltages \mathbf{U}_{PWM} for the three phases. It creates a sine commutation on the electric currents in each phase, this requires the rotor angle θ_{mot} . Note that the MOSFETs are not shown in figure 4.1 because they are not relevant

for the issues addressed in this section. There is an additional controller eliminating the wattless currents in the D-axis. The DES does not apply field weakening, thus the wattless currents can be neglected and this feature is also not shown in figure 4.1. The effective current I_{mot} in the Q-axis is relevant for producing the motor torque.

The rotor angle is obtained with a dual channel digital encoder. This encoder contains a disk attached to the rotor shaft with two rings of 500 equally spaced holes around the circumference. The holes of the two rings are shifted by a quarter hole width creating 2000 distinct rotor positions θ_{DES} . These positions are referred to as quarter counts (qc) and they are detected photoelectrically. One extra hole ensures a readjustment after one full rotation for precise initialization. The initial rotor position at startup is estimated with two analog hall sensors.

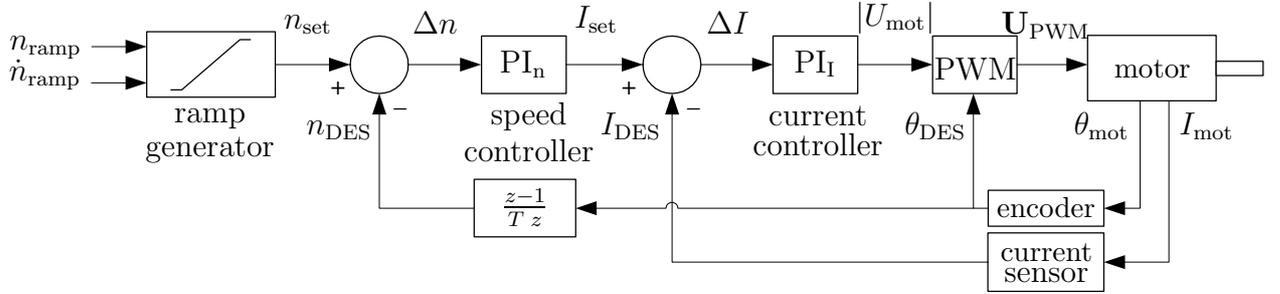


Figure 4.1: Motor controller scheme

The input for the PWM generator is the amplitude of the voltage to the windings $|U_{\text{mot}}|$ which is created by the current controller PI_I . The current controller is a simple proportional and integrative controller that runs at a sampling rate of 8 kHz. It uses the measured value I_{DES} of the actual effective current to the motor I_{mot} and tries to match it with the current set value I_{set} . Each DES is equipped with two current sensors located in two of the three motor winding circuits. They are used to determine both currents in Q- and D-axis. Type and precision of these sensors are unknown, they are not specified by the manufacturer. If the DES is operated in current regulation mode, it accepts I_{set} as input variable and disables the preceding speed regulator. The proportional and integrator gains of the current controller are $k_{I,P}$ and $k_{I,I}$, respectively. The deviation of the control variable is computed with the unfiltered value of I_{DES} as

$$\Delta I = I_{\text{set}} - I_{\text{DES}}. \quad (4.1)$$

For speed regulation mode, the speed controller PI_n in figure 4.1 is cascaded in front of the current controller. It is a simple proportional and

integrative controller just like the current controller and runs at a sampling rate of 1 kHz to produce the output I_{set} which is the set value for the subsequent current controller. The speed controller's proportional and integrator gains are $k_{n,P}$ and $k_{n,I}$, respectively, its input variable is the speed deviation

$$\Delta n = n_{\text{set}} - n_{\text{DES}}. \quad (4.2)$$

The rotor angle θ_{DES} measured with the digital encoder is used to calculate the rotor speed n_{DES} within the speed controller cycle. The sampling time of the speed controller is 1 ms and the minimum rotor angle resolution is 1 qc which equals 1/2000 rotation, the resolution of the measured speed therefore is

$$n_{\text{DES,res}} = 1 \frac{\text{qc}}{\text{ms}} = 30 \text{ rpm}. \quad (4.3)$$

For a maximum motor speed of $n_{\text{max}} = 6000$ rpm, a single rotation lasts 10 ms which leaves the speed controller 10 samples. The speed controller has an important feature that allows to create smooth speed transients. Instead of changing the controller set speed n_{set} stepwise, the DES allows to generate an internal ramp for the set speed. This is illustrated by the leftmost box in figure 4.1. The ramp generator requires the end value n_{ramp} as well as the slope of the ramp \dot{n}_{ramp} which represents the rotor acceleration. If the DES are operated in speed regulation mode, the velocity n_{ramp} is the input variable to the controller. The rotor acceleration \dot{n}_{ramp} can be initialized as a constant value or it can be adjusted dynamically together with n_{ramp} . The ramping option of the DES is expected to improve the robot's driving performance because it allows a smoother control of the wheel speeds. Without ramping the motor speed, the acceleration value \dot{n}_{ramp} must be set to a high constant value that will allow the greatest increase of the motor speed that is conceivable in the period of the robot motion controller. This will produce very high peaks in the motor currents every time the motor speed n_{ramp} is changed, resulting in peaks in the wheel torques and increased slip between wheels and floor.

In the current implementation, the robot communicates with the DES via CAN bus at a transfer rate of 512 kbit/s. The DES operate in speed regulation mode with dynamical update of the rotor acceleration parameter. During each robot control cycle four CAN messages are exchanged between robot and the three DES. The robot sends three messages to each DES: the new speed n_{ramp} , the rotor acceleration \dot{n}_{ramp} , and a request to reply some predefined DES internal states. Each DES will return one CAN message containing its internal states. These internal states are defined as: measured rotor speed n_{DES} , measured effective motor current I_{DES} , and relative rotor

encoder position θ_{DES} in quarter counts. It takes less than 1 ms to communicate the new ramp speeds and slopes from the robot to each of the three DES. Refer to section 6.5.1 for a discussion of the tuning procedure for the DES controller parameters.

4.2 Robot Motion Controller

The robot motion controller is responsible for the robot to move along a given drivecycle trajectory. The drivecycle trajectory is provided in terms of a lookup table that contains the calculated robot set state \mathbf{x}_{tab} comprising position and orientation in global coordinates for a list of discrete times t_{tab} . The creation of the trajectory is described in chapter 3. It requires a model of the robot and the ball as it has been provided in sections 1.2.3 and 3.2.1, respectively. This allows an effective implementation of the robot motion controller in two parts: a feed forward compensation that simply feeds through the calculated robot velocity, and a feedback controller that corrects course deviations based on the actual robot state. The robot state is defined by its position \mathbf{x} and its velocity $\dot{\mathbf{x}}$ according to equation (1.3), p. 15:

$$\mathbf{x} \stackrel{\text{def}}{=} x \mathbf{e}_x + y \mathbf{e}_y + \alpha \mathbf{e}_\alpha = [x \ y \ \alpha]_{xy\alpha}^T \quad (4.4)$$

$$\dot{\mathbf{x}} \stackrel{\text{def}}{=} \dot{x} \mathbf{e}_x + \dot{y} \mathbf{e}_y + \dot{\alpha} \mathbf{e}_\alpha = [\dot{x} \ \dot{y} \ \dot{\alpha}]_{xy\alpha}^T \quad (4.5)$$

Figure 4.2 shows a schematic of the robot motion controller. It starts with the drivecycle that is documented in form of a lookup table for the robot positions \mathbf{x}_{tab} in global coordinates for discrete times t_{tab} . The tabulated values are interpolated by means of the weighted polynomial approximation (WPA) described in section 3.7 to obtain the robot drivecycle state \mathbf{x}_{cyc} , $\dot{\mathbf{x}}_{\text{cyc}}$, and $\ddot{\mathbf{x}}_{\text{cyc}}$ at the actual time. The feed forward path is constituted by the WPA that computes the robot velocity $\dot{\mathbf{x}}_{\text{cyc}}$ based on the tabulated positions. The velocity $\dot{\mathbf{x}}_{\text{cyc}}$ is transformed into the velocity $\dot{\boldsymbol{\xi}}_{\text{set}}$ in robot coordinates according to equation (1.8), p. 16, and subsequently converted into the wheel speeds according to equations (1.9), p. 16, and (1.7), p. 16. The motor velocities \mathbf{n}_{ramp} for the DES are obtained by multiplying with the motor gear ratio i_{gear} . In case of a strict forward feeding, if the robot is blind, the drivecycle orientation α_{cyc} must be used for the transformation according to equation (1.8). This is not illustrated in figure 4.2. It has the disadvantage that the decoupling of the degrees of freedom can not fully be exploited if the robot is moving. If the actual robot orientation is misaligned compared to the drivecycle orientation, then the robot will drive in the wrong direction, causing a deviation in the translational x - and y -directions as well.

following section 4.3. This extrapolated robot position is called \mathbf{x}_{EXT} . It is used as actual robot position in the controller, and its orientation α_{EXT} is used for the transformations according to equation (1.8), p. 16.

The odometric feedback path is illustrated at the top in figure 4.2. The actual robot velocity $\dot{\mathbf{x}}_{\text{is}}$ is associated with the wheel speeds \mathbf{n}_{mot} . Each DES determines its respective motor speed with the digital encoders and communicates it via CAN bus to the robot. These speeds \mathbf{n}_{DES} can reversely be translated into robot velocities $\dot{\boldsymbol{\xi}}_{\text{DES}}$ using equations (1.7), p. 16, and (1.9), p. 16, and dividing by the gear ratio i_{gear} . Equation (1.8) allows the calculation of the global robot velocities $\dot{\mathbf{x}}_{\text{DES}}$ which can easily be integrated to obtain the robot position \mathbf{x}_{DES} . There is a number of problems associated with this method. First, the method can only count increments. Therefore, the robot position needs to be initialized at the beginning. Second, it allows error propagation through the robot orientation α_{DES} . This angle is required to calculate the robot global velocity. Every misalignment in the robot orientation will cause equation (1.8) to produce wrong global translational velocities and thus an orientation error will propagate in the translational positions. Third, the wheel speeds never precisely represent the robot velocity because there is always slip between wheels and floor. Fourth, there are two numeric truncation errors in the method: the motor speeds are only supplied in multiples of $n_{\text{DES, res}} = 30$ rpm, this was described in the previous section. The second truncation error concerns the integration time interval. The robot control software is implemented on a regular personal computer (PC) running MS-DOS as operating system. This is not a real time platform and the time is counted in multiples of $t_{\text{res}} = 1$ ms which is also the uncertainty of time measurement. Both truncation errors can be removed easily with a simple trick that eliminates both the numerical differentiation for the motor speeds and the numerical integration of the robot velocities. The motor rotor angle θ_{DES} is also transferred to the robot via CAN bus together with the motor speeds \mathbf{n}_{DES} . The increment of this angle $\Delta\theta_{\text{DES}}$ is used in equations (1.9) and (1.8) to obtain an increment in global coordinates:

$$\Delta\mathbf{x} = i_{\text{gear}}^{-1} \mathcal{O}^T \mathcal{Y}^{-1} \Delta\theta_{\text{DES}} \quad (4.6)$$

The video image recognition was chosen as primary sensor because it is capable of determining the absolute global position without any reference, because it does not propagate errors, and because it does not have the intrinsic slip disturbance. The only drawback of the algorithm described in chapter 5 is that it requires an initial guess of the robot position for each calculation. The odometric determination of the robot position based on the DES rotor angle θ_{mot} was also implemented. It can serve as short term

alternative if the VIR is not capable to determine the robot position. This can happen if the robot drives through an area with insufficient markings on the floor. The odometric robot position was evaluated in the experiment for determining the robot physical parameters, this is described in section 6.3.

The robot motion controller itself illustrated in figure 4.2 is a simple proportional controller. The robot deviation from the drivecycle $\Delta \mathbf{x}_{cyc}$ is obtained by subtracting the interpolated drivecycle position \mathbf{x}_{cyc} from the actual extrapolated position \mathbf{x}_{EXT} :

$$\Delta \mathbf{x}_{cyc} = \mathbf{x}_{EXT} - \mathbf{x}_{cyc} \quad (4.7)$$

The orientation angle deviation $\Delta \alpha_{cyc}$ is clipped to the interval $[-\pi, \pi]$. The controller output velocity $\dot{\mathbf{x}}_{ctr}$ is computed by multiplying the translational and rotational deviations with respective proportional gain factors:

$$\begin{aligned} \dot{x}_{ctr} &= -k_{xy} \Delta x_{cyc} \\ \dot{y}_{ctr} &= -k_{xy} \Delta y_{cyc} \\ \dot{\alpha}_{ctr} &= -k_{\alpha} \Delta \alpha_{cyc} \end{aligned} \quad (4.8)$$

The robot controller velocity $\dot{\mathbf{x}}_{ctr}$ is added to the drivecycle velocity $\dot{\mathbf{x}}_{cyc}$ from the feed forward compensation to adjust the robot's course and orientation. Note that the robot will drive in the correct translational direction even if its orientation is misaligned because the actual extrapolated angle α_{EXT} is used in the feed forward compensation path to compute the local velocities $\dot{\xi}_{set}$. The robot motion controller sampling time is 40 ms because that is the sample time of two half images determined by the video cameras. It does not make sense to use a shorter sampling time because no more information on the actual robot position is available. It is not even necessary to create interpolated motor speeds because the speed ramping function within the DES takes care of that. In the robot control software, the motion controller subroutine is called every time after the video image recognition subroutine. The robot controller is cascaded in front of the DES speed controller. Therefore, its sampling period can be greater than that of the DES speed regulator. This is the case with a factor of 40. The controller architecture has deliberately been chosen as simple as possible because a strong emphasis was put on a precise propulsion system in the development of the robot hardware. It was the purpose to build a robot that stays very accurately on its predefined trajectory even at high velocities and accelerations such that a sophisticated controller is not necessary. The only task of the motion controller is to keep the robot on track on a long-term time basis. See section 6.5.3 for the experimental performance of the robot motion controller.

4.3 Robot Position Estimation

This section describes the delay in the VIR signal chain and how it is remedied. Refer to section 5.1.1 for a detailed discussion of the sensor hardware. The robot's real position is captured every 20 ms by the exposure of three cameras synchronously. The three video images are transferred to three respective video frame grabbers. They use PAL format which means the cameras split each image into two half images, one containing all odd lines and the other containing all even lines. Only one half image is transferred to the grabbers after each exposure, the other is discarded. After the next exposure, the other half image is transferred. For the VIR process, only odd half images are used.

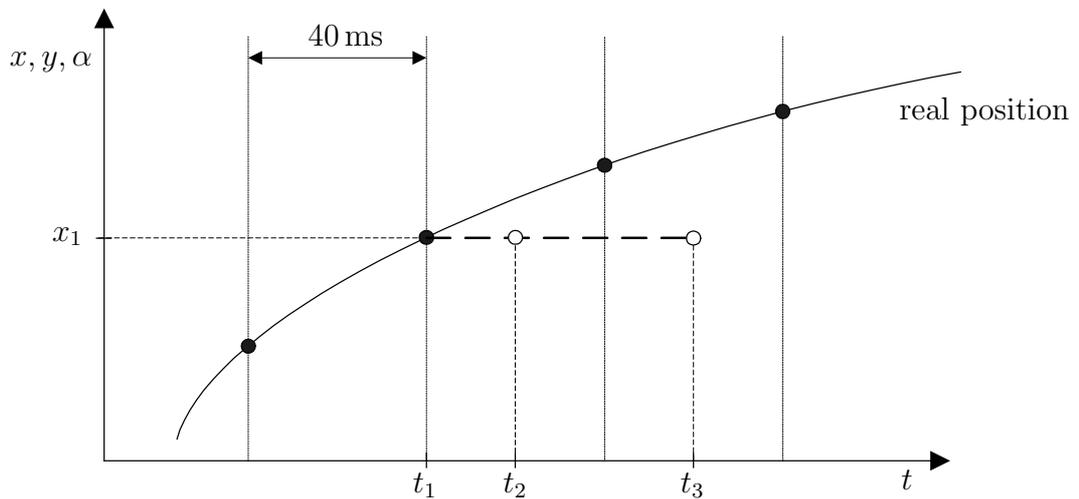


Figure 4.3: Robot position estimation: Video image time delay

Figure 4.3 illustrates the sequence of actions for the video image recognition process. Every 40 ms an odd picture is exposed in the cameras which is indicated by the filled bullets in the figure. Assume the robot has the x -coordinate x_1 at time t_1 when the odd half images are exposed in the three synchronized cameras. It takes then 20 ms to transfer the analog FBAS signals of these half images to the respective frame grabbers. This process is completed at time t_2 . The frame grabbers write each line of the digital half image directly into the mainboard's RAM in real time during digitizing so there is no significant extra delay produced by the grabbers. The video image recognition is performed by the CPU, this process can last up to 40 ms with the current hardware and video image resolution. At time t_3 , the robot position is determined. The only problem is that the information is up to

60 ms old and the robot may have moved 0.3 m if it is travelling at a velocity of 5 m/s. Note that while the actual image exposed at time t_1 is still being processed, the next image exposed at time t_0 may already be digitized, overwriting the actual one. This is a foreseen course of action that does not disturb the algorithm due to the modular architecture of the VIR algorithm described in section 5.1.

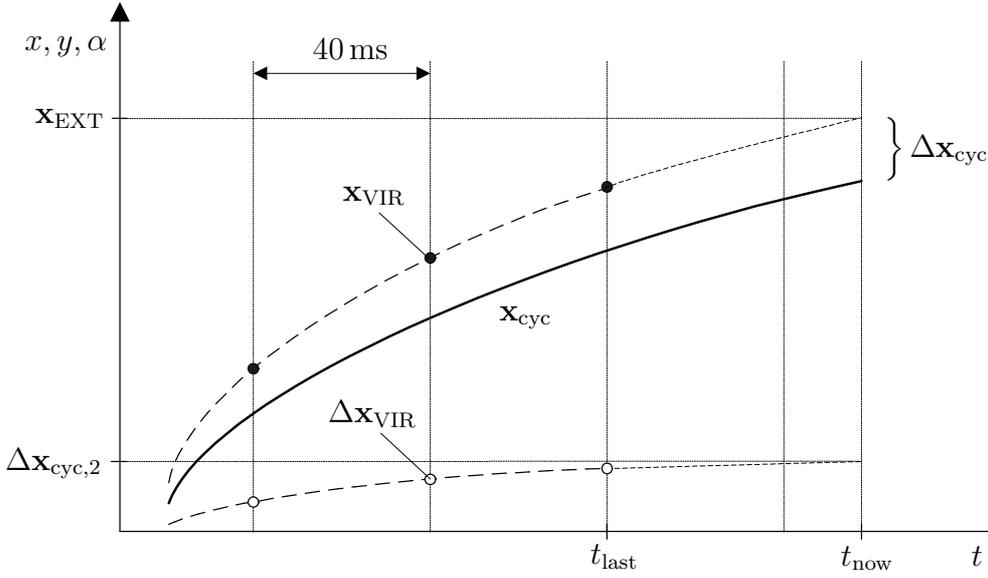


Figure 4.4: Robot Position Estimation: Position extrapolation

It was the goal to find a method to obtain an estimate of the actual robot position that is independent on any secondary sensor like the odometric system. Therefore, the projection of the past robot positions into the future is the only possibility. Figure 4.4 shows how the signal delay is dealt with by the robot motion controller. The solid thick line represents the calculated robot position \mathbf{x}_{cyc} according to the drivecycle trajectory. The filled bullets indicate the times at which the odd half images are exposed and the associated robot positions \mathbf{x}_{VIR} calculated with the VIR algorithm. At time t_{now} the VIR algorithm has been completed and the most recent robot position at time t_{last} has been determined. This time is approximately 60 ms ago as described above. The motion controller needs the actual robot position \mathbf{x}_{EXT} at time t_{now} in order to compute the actual course deviation $\Delta \mathbf{x}_{cyc}$ according to equation (4.7). This can be achieved with help of the weighted polynomial approximation described in section 3.7. There are two possibilities to apply the WPA method.

The first possibility is extrapolating the robot positions \mathbf{x}_{VIR} to obtain an extrapolated value \mathbf{x}_{EXT} at time t_{now} . Then the position deviation $\Delta\mathbf{x}_{\text{cyc}}$ can be computed according to equation (4.7).

The second possibility is to compute the deviation $\Delta\mathbf{x}_{\text{VIR}}$ for each robot position obtained with the VIR, this is illustrated in figure 4.4 by the empty bullets. These previous deviations can then be extrapolated to obtain the actual course deviation $\Delta\mathbf{x}_{\text{cyc},2}$ at time t_{now} . This second possibility is more complicated and more time consuming because the drivecycle position needs to be computed additionally at all camera exposure times and these times do not coincide with the motion controller sampling times. In addition to that, the previous deviation values $\Delta\mathbf{x}_{\text{VIR}}$ indicated by the empty bullets in figure 4.4 need to be stored in a separate array to perform the extrapolation with the WPA method.

Both methods have been implemented and compared. They showed so little difference in their predictions that the results are not discussed in chapter 6. The first possibility extrapolating the robot positions \mathbf{x}_{VIR} was used in all experiments to obtain the value \mathbf{x}_{EXT} because of its simplicity. Refer to figure 6.35, p. 146, for the test results on the performance of the WPA method for extrapolating the robot position according to the first possibility.

Chapter 5

Sensors

The two main objectives for the robot sensors are self localization and mobile object localization. All results will refer to the fixed global coordinate system. Self localization yields the robot's own position and orientation. The primary mobile object is the ball whose trajectory is to be controlled. In the RoboCup competition, the ball and other robots are considered moving objects.

There is a multitude of feasible sensors, they can be divided into the categories optical, acoustical, and contact sensors. Infrared devices, laser scanners, and video cameras are optical sensors. Ultrasonic devices are acoustic sensors, and electrical switches, potentiometers, and wire resistance strain gauges are contact sensors. All infrared, ultrasonic, and laser scanners rely on the reflection of a transmitted signal. This works for detection of mobile objects and for self localization in a fenced environment. In a completely flat environment, these sensors will fail to determine the robot's own position due to the absence of reflections. For this reason, video cameras are used as sensors because they can operate under all aforementioned conditions. For self localization in a non fenced flat environment, they require markings on the floor. They are applicable for RoboCup because there floor markings are defining the soccer field. In addition to that, colors play a vital role in RoboCup competitions, and video cameras are the only sensors that are capable of capturing colors.

5.1 Image Processing

The video cameras produce images that are processed digitally by a microprocessor in a sequence of steps. Figure 5.1 illustrates these steps and their respective intermediate products. Starting from the left, two objects are in a camera's field of view. The red circle that is crosshatched from bottom left

to top right could represent a ball while the blue square, crosshatched from top left to bottom right, is a place holder that could represent a black robot partially covering the ball, or a white field marking line. This situation is captured by the video cameras, digitized by frame grabbers, and ported into the computer memory in form of digital images. The first step executed by the software algorithm is to recognize all colors that are of interest, and map each pixel according to its membership into a color coded representation of the digital image. Then the outlines of each area in the color coded image are detected according to certain rules. At this point, it is possible to distinguish between pixels that are located on the red ball's circular circumference and outline pixels that belong to the corner created by the blue square. The last step is the recognition of the isolated objects using the outline pixel information only.

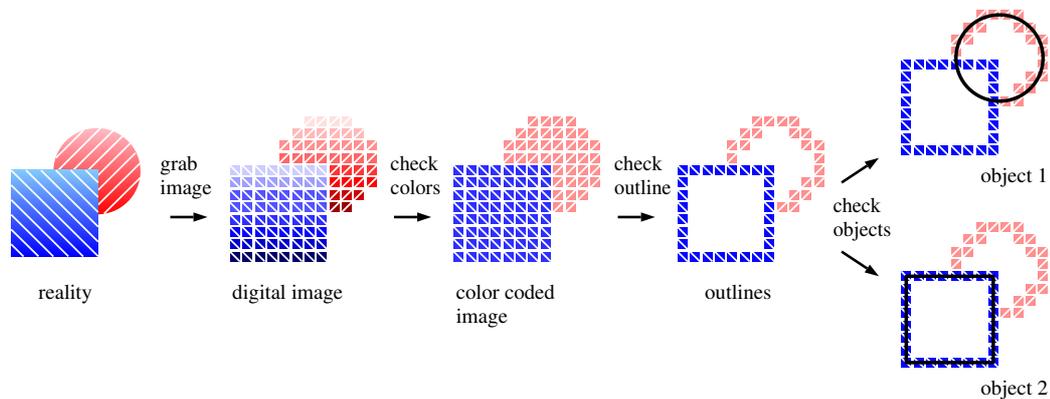


Figure 5.1: Image processing algorithm

The quality of the sensors' performance is determined by the precision and robustness of the solution as well as the computational time required. Robustness and precision of the obtained solution are achieved with a statistical method for object recognition. The algorithm presented above has been developed and implemented for real time image processing on a computer system which satisfies the computational speed criterion. It is suitable for the RoboCup benchmark where the use of colors is strictly defined: The floor is green, field markings are white. The soccer ball is red and the two goals are painted blue and yellow, respectively. All robot players must wear black color.

5.1.1 Image Capturing Hardware

The hardware setup for the video image processing system is shown in figure 5.2. Three synchronized color cameras continuously capture video im-

ages. The cameras are mounted at the highest point of the robot in order to view the field markings as vertically as possible. This is important because all outline pixels representing field markings will be mapped into a top view. The steeper the cameras view downwards, the more accurate the mapping will be.

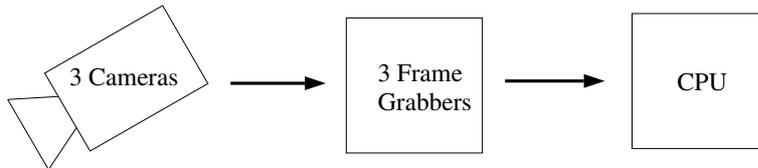


Figure 5.2: Camera hardware setup

The cameras are equipped with a 2.3 mm fixed focus wide angle lens that allows a horizontal and vertical angle of view of 120° and 90° , respectively. Refer to figure 1.7, p. 24, for an illustration of their mounting position inside the robot. Three cameras have been implemented, they enable omnidirectional vision. Their optical axes are mounted at a vertical angle of 45° ; horizontally they are oriented at angles of 120° towards each other. One camera is facing in the robot ξ -direction which is the *forward* direction, the other two cameras are facing rear left and rear right. All cameras are rigidly attached to the robot for simplicity and robustness of the system, and to avoid dead times during repositioning. The cameras have a 1/3 inch CCD video sensor with a resolution of 752*582 pixels. They produce an analog output signal at 25 fps (**f**rames **p**er **s**econd) according to the PAL standard.

Each video signal enters a separate frame grabber which is responsible for digitizing the image and transferring the data into the computer RAM via the PCI bus and direct memory access (DMA). The grabbers can handle a maximum resolution of 720*576 pixels at a maximum color depth of 24 bit or 16 Mio. colors. Each grabber can digitize and store images at full resolution in full color depth in real time at 25 fps. The grabbers can store odd and even half frames in different color formats. This allows an extremely hardware oriented color detection routine: Odd half frames are stored in the YUV color format for color detection, and even half frames are stored in RGB format for on-screen visualization and debugging. Refer to section 5.1.2 for a brief description of different color models.

The image processing algorithm runs on a Pentium 266 MMX processor with 64 MB RAM. The operating system is MS-DOS and the programming language for both video image processing and motor control is C and C⁺⁺. In order to be able to process all video data in real time with the above mentioned CPU, the resolution had to be reduced by the frame grabbers from the original

half frame size of 720*288 pixels down to 92*144 pixels. The resolution in vertical direction was reduced by a smaller factor because this direction is more important to determine the distance of objects. Refer to tables A.1, p. 169, and A.2, p. 171, for a list of the hardware components. Appendix A.3 describes the functionalities of the video image recognition carried out by the robot control program and provides screenshots of the graphical output.

5.1.2 Color Recognition

Various color models are defined as bases to express the color information of each pixel. Three of them are illustrated in figures 5.3, 5.4, and 5.5.

The RGB (**R**ed **G**reen **B**lue) color model is supported by common graphical output devices. It is therefore suited for visualization of the video images. It is not suited for color recognition because colors need to be defined in a three dimensional space which requires many parameters, is hard to visualize, and requires large computational effort. Figure 5.3 shows a three dimensional view of the color space. The three RGB base vectors origin in the black corner of the color cube. The white corner is opposite to the black one, all values R, G, and B reach their maximum value for the color white.

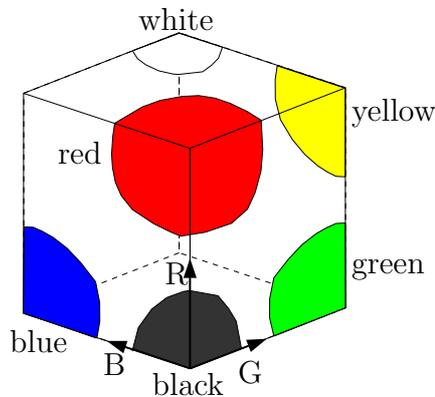


Figure 5.3: RGB color space

The HSI (**H**ue **S**aturation **I**ntensity) and YUV (Luminance **Y**, Chromance **UV**) color models are more advantageous for color recognition. In both models, the brightness information, which does not contain color information, is one base of the color space. This reduces the three dimensional color space by one dimension as far as color classification is concerned. The brightness axis corresponds to the diagonal line from the black to the white corner in the RGB color space model. It carries all shades of gray from black (zero brightness) to white (maximum brightness). The brightness axis is

perpendicular to the paper plane in both visualizations of the HSI and YUV color space in figures 5.4 and 5.5, respectively.

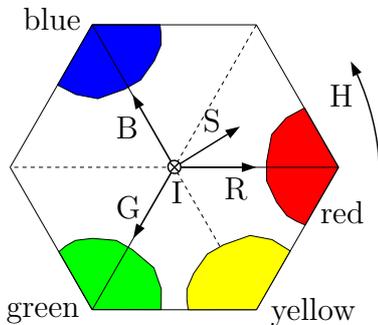


Figure 5.4: HSI color space

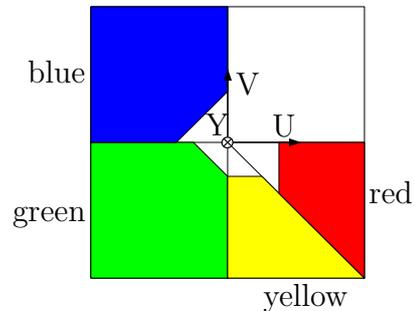


Figure 5.5: YUV color space

In the HSI model, each color is a sum of gray and a pure color contribution. The hue value H specifies the quality of the pure color contribution (e.g. red, blue, green, or yellow), it can be visualized as an angle of rotation in figure 5.4. The saturation S expresses the ratio of pure color to gray, it corresponds to a radial coordinate. The intensity I represents the color's brightness, its base vector is pointing into the paper plane. This color model is specifically suited for color recognition because it reduces the pure color information to a single dimension. The intensity value is responsible for shades of the color while the saturation is a measure for the pureness of the color.

The YUV model is an intermediate model between RGB and HSI because it reduces color detection to two dimensions by extracting the brightness information (Y value). The U and V values represent color difference signals in the red and blue axis. In figure 5.5, the brightness axis is pointing into the paper plane while the U - and V -axis define the two dimensions carrying color information. This model is the most hardware oriented because it reflects the constitution of the PAL signal produced by the cameras. It is the only color format that is supported by the frame grabbers beside the RGB format. This color model has been implemented for color detection because it allows an extremely efficient color recognition routine due to the fact that each of the colors red, green, blue, and yellow is defined by three linear limits only. The colors black and white can be detected by the brightness value Y exclusively.

5.1.3 Outline Detection

There are two reasons for extracting the object outlines from the color coded image according to figure 5.1. The first reason is reduction of information and the resulting reduction of computational time. The second reason is

improvement of data quality. In the color coded image representation, the outline of an object contains the information about its shape while its face does not add any valuable information. In addition to this, the neighborhood of the object image outline allows to determine three dimensional relations between objects in reality. Outlines created by partial covering of an object do not belong to the object's original shape and should be ignored for shape identification, this is explained with an example:

If a red ball pixel has green neighboring pixels on a RoboCup field, the red pixel must necessarily belong to the ball circumference, because the green pixels originate from the field color. If a red ball pixel has black neighboring pixels, it is most likely that a robot is partially covering the ball and the red pixel belongs to the borderline between robot and ball. This is illustrated in figure 5.1: The lower left section of the red circle is covered by the blue square. Red pixels neighboring blue ones are not considered ball outline in this example.

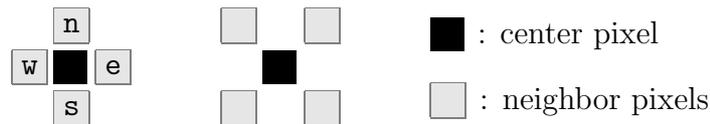


Figure 5.6: Pixel neighborhood patterns

Various patterns can be defined to determine neighborhood relations. Two patterns are illustrated in figure 5.6. The left pattern only considers north, east, south, and west pixels as neighbors, while the right pattern only uses the four diagonal neighbor pixels. Conditions on the color code of the neighbor pixels can be algebraic (e.g. *at least 3 neighbors must be green*) or boolean (e.g. *neighbor n must be green or blue*). The pseudo code in table 5.1 gives an example for boolean neighboring conditions to detect field marking and ball outlines.

```

IF (  $\blacksquare$  = white) AND ( (  $\square_w$  = green) OR (  $\square_e$  = green)
    OR (  $\square_s$  = green) OR (  $\square_n$  = green) )
    THEN objectcode( $\blacksquare$ ) = MARKING
ELSE IF (  $\blacksquare$  = red) AND ( (  $\square_w$  = green) OR (  $\square_e$  = green)
    OR (  $\square_s$  = green) OR (  $\square_n$  = green) )
    THEN objectcode( $\blacksquare$ ) = BALL
END IF

```

Table 5.1: Outline detection sample pseudo code

5.1.4 Object Localization

For shape recognition and object localization, some assumptions are made. Self localization is based on the localization of immobile objects. Only field markings are considered immobile objects. They are assumed on floor level at zero height. There are no landmarks above floor level. This is a severe restriction of generality that is appropriate for the following reasons: The omnidirectional wheels have very small ground clearance which restricts the robot operation to flat surfaces. In an artificially flat environment, landmarks for orientation can as well be located on the floor. RoboCup soccer field markings are also on floor level. The flat ground assumption allows the following simplifications: No stereo vision is required in order to determine a pixel's distance from the robot. A lookup table can store the coordinates (ξ, η) relative to the robot coordinate system for each pixel. This also allows an extremely fast processing of pixel coordinates. A prior calibration of the lookup table map removes all perspective and aberrational distortions. Localization of the field markings is carried out in ξ - η coordinates because the mapping restores lengths and angles of the original marking lines. For detection of RoboCup field markings, the exact geometry is defined in form of a parameterized model. The procedure for localizing the robot's position on the field that was developed in the course of this research project is presented in the next section. Refer to section 6.2.1 for a description of the calibration procedure, and to appendix A.2 for a presentation of the developed calibration software.

Mobile objects cannot be treated in the same way because they are not located on floor level. Therefore, these objects are localized in bitmap coordinates. After determining the geometrical center of an object whose shape is known, its projection on the floor can be used in combination with the lookup table to determine the object's position in ξ - η coordinates. This is explained in section 5.3. Due to the small size of all mobile objects, little distortion by the cameras is assumed. Therefore, the outline of an objects' video image satisfactorily represents its real shape. The ball outline will always appear as fractions of a circle.

Note that only the location of the ball is determined at this stage of development. The robot does not search for opposing players yet. Opponent robot players can vary in shape and size which makes a straightforward detection more difficult. In this case, it seems more reasonable to only consider the lowermost section of a robot outline because this part of the outline belongs to pixels on floor level. An extrapolation of this data can yield the robot's approximate position. This procedure has not been examined in detail because it is not part of the primary objective of this research project.

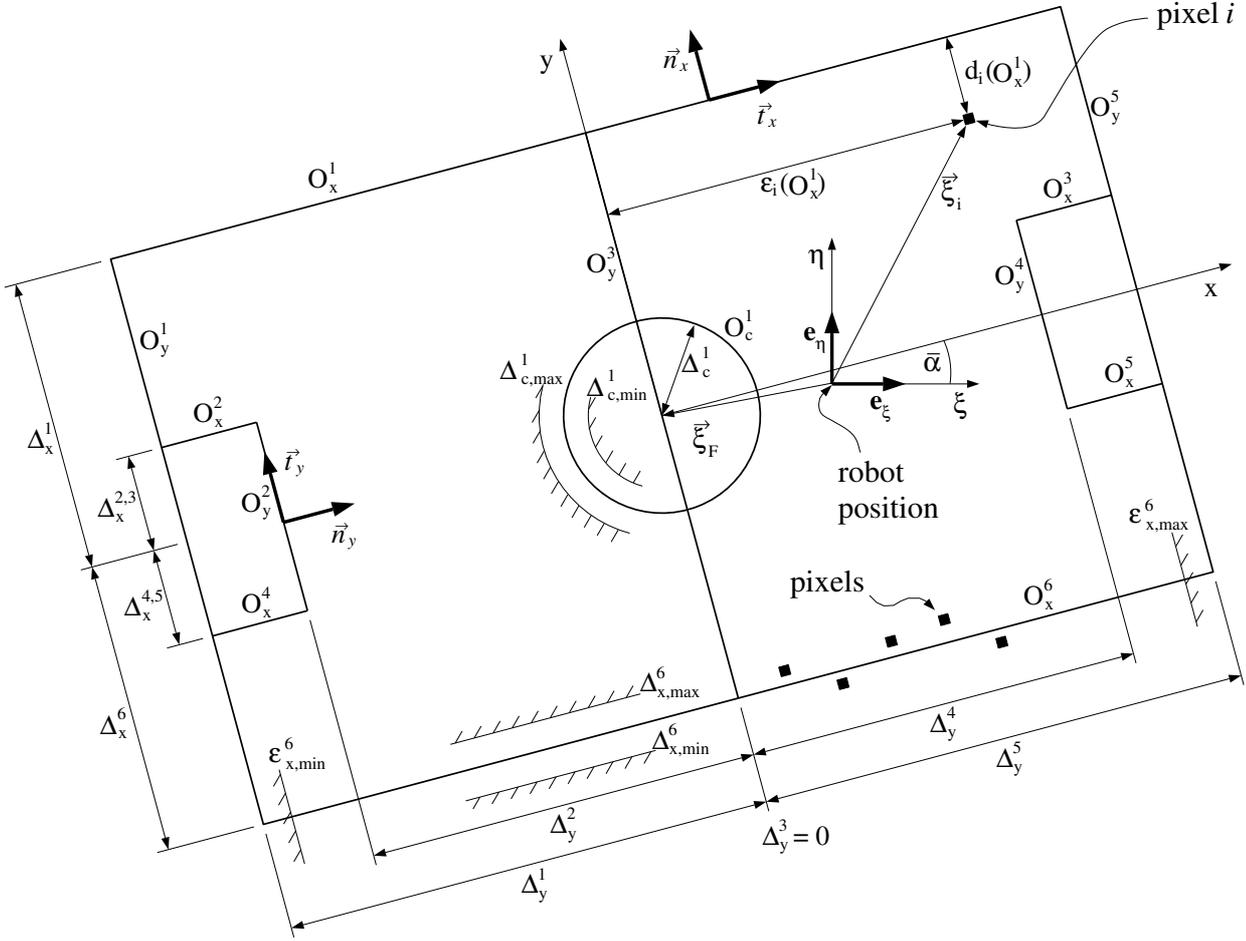


Figure 5.7: Soccer field geometry

in the current implementation: straight lines parallel to the global coordinate axes, and circles. They are referred to as:

$$O_k \in \{O_x, O_y, O_c\} \quad , \quad k = x, y, c \quad (5.6)$$

The object classes O_x and O_y represent straight lines parallel to the x - and y -axis, respectively, O_c is a class of circles whose center is located on the origin of the x - y coordinate system. It is possible to add other objects such as lines at arbitrary angles, circles at any position, or more complex one dimensional objects. These additional objects have not been considered because they do not contribute to the general understanding of the approach and they are not needed to model the RoboCup field either.

All line class objects have a common property, their orientation $\bar{\alpha}$ in robot coordinates. This is expressed in a tangent vector \vec{t} and a normal vector \vec{n}

for each class. Due to their symmetry, the circle class objects do not have this property. All vectors \vec{t}_k and \vec{n}_k point in positive axis directions, they are depicted in figure 5.7 for the object instances O_x^1 and O_y^2 .

$$\vec{t}_x(\bar{\alpha}) = \cos(\bar{\alpha})\mathbf{e}_\xi + \sin(\bar{\alpha})\mathbf{e}_\eta \quad (5.7)$$

$$\vec{n}_x(\bar{\alpha}) = -\sin(\bar{\alpha})\mathbf{e}_\xi + \cos(\bar{\alpha})\mathbf{e}_\eta \quad (5.8)$$

$$\vec{t}_y(\bar{\alpha}) = -\sin(\bar{\alpha})\mathbf{e}_\xi + \cos(\bar{\alpha})\mathbf{e}_\eta = \vec{n}_x(\bar{\alpha}) \quad (5.9)$$

$$\vec{n}_y(\bar{\alpha}) = -\cos(\bar{\alpha})\mathbf{e}_\xi - \sin(\bar{\alpha})\mathbf{e}_\eta = \vec{t}_x(\bar{\alpha}) \quad (5.10)$$

Each object class can create a number $n(O_k)$ of instances each of which having additional properties.

$$O_x^l : \Delta_x^l, \Delta_{x,\min}^l, \Delta_{x,\max}^l, \epsilon_{x,\min}^l, \epsilon_{x,\max}^l \quad , \quad l = 1, \dots, n(O_x) \quad (5.11)$$

$$O_y^l : \Delta_y^l, \Delta_{y,\min}^l, \Delta_{y,\max}^l, \epsilon_{y,\min}^l, \epsilon_{y,\max}^l \quad , \quad l = 1, \dots, n(O_y) \quad (5.12)$$

$$O_c^l : \Delta_c^l, \Delta_{c,\min}^l, \Delta_{c,\max}^l \quad , \quad l = 1, \dots, n(O_c) \quad (5.13)$$

Figure 5.7 illustrates these object instance properties. Several straight lines and a circle constitute the markings of a simple soccer field. All lines are parallel to one of the axis of the global x - y coordinate system that originates at the field center which is also center of the circle. There are 6 individual lines parallel to the x -axis, $n(O_x)=6$. Two of these lines mark the field boundary, the remaining four lines belong to the penalty areas. Five lines are parallel to the y -axis, $n(O_y)=5$, out of which two are field boundary markings, one is the field center line, and two define the penalty areas. Each of these lines and the circle, $n(O_c)=1$, represent an object instance according to their labelling. The lower field boundary line O_x^6 is displayed with its instance property information at the bottom of figure 5.7. Δ_x^l is a line's distance from the origin, $\Delta_{x,\min}^l$ and $\Delta_{x,\max}^l$ define a band parallel to the line. A pixel must be located inside this band in order to be associated to the line. The same properties are defined for lines parallel to the y -axis. The line extent in longitudinal direction is defined by $\epsilon_{x,\min}^l$ and $\epsilon_{x,\max}^l$. Circle object instances are defined by the radius Δ_c^l and the radial bounds $\Delta_{c,\min}^l$ and $\Delta_{c,\max}^l$.

The geometrical distance between pixel i and object instance O_k^l is

$$\tilde{d}_i(O_k^l) = (\vec{\xi}_i - \vec{\xi}_F) \cdot \vec{n}_k(\bar{\alpha}) - \Delta_k^l \quad , \quad k = x, y \quad (5.14)$$

$$\tilde{d}_i(O_c^l) = \sqrt{(\vec{\xi}_i - \vec{\xi}_F) \cdot (\vec{\xi}_i - \vec{\xi}_F)} - \Delta_c^l \quad (5.15)$$

The distance d_i that is used for the least squares approach is not always identical to the geometrical distance \tilde{d}_i . For object classes O_x and O_y , it is

the geometric distance, but for circular objects, it saves computational effort to modify the definition of the distance d_i such that it does not contain a square root:

$$d_i(O_k^l) \stackrel{def}{=} (\vec{\xi}_i - \vec{\xi}_F) \cdot \vec{n}_k(\bar{\alpha}) - \Delta_k^l \quad , \quad k = x, y \quad (5.16)$$

$$d_i(O_c^l) \stackrel{def}{=} (\vec{\xi}_i - \vec{\xi}_F) \cdot (\vec{\xi}_i - \vec{\xi}_F) - (\Delta_c^l)^2 \quad (5.17)$$

5.2.2 Least Squares Approach

Let $\vec{\xi}_i$ be the coordinates of field marking outline pixel number i in robot coordinates, this is illustrated in the top right corner in figure 5.7. Then d_i is the distance of this pixel to its closest geometrical object, in this case $d_i = d_i(O_x^1)$. The approach presented here minimizes the sum S_F of the squared distances d_i^2 over all pixels $n_{\text{pix}}(\mathbb{F})$ that represent field markings:

$$S_F \stackrel{def}{=} \frac{1}{2} \sum_{i=1}^{n_{\text{pix}}(\mathbb{F})} d_i^2 \stackrel{!}{=} \min \quad (5.18)$$

$$\frac{dS_F}{d\vec{\xi}_F} = \sum_{i=1}^{n_{\text{pix}}(\mathbb{F})} \left(d_i \cdot \frac{d(d_i)}{d\vec{\xi}_F} \right) \stackrel{!}{=} 0 \quad (5.19)$$

Note that equations (5.18) and (5.19) do not use the geometrical distance \tilde{d}_i but the modified distance d_i . This has the drawback that the summands of S_F have different physical units of measurement. The use of weight factors w_k^l for different object classes and instances can reduce this imbalance in the numerical treatment. Refer to appendix B.7 for more details. In order to determine d_i for a pixel, the pixel must be assigned to a geometrical object. All calculations are performed in robot coordinates, the pixel distances have units of m. Pixel i is assigned to an object of class k , instance l , if the following condition is fulfilled: The pixel must be located inside the geometrical object's bounds. The object classes O_x and O_y are bounded in tangential and normal directions, object class O_c has radial bounds:

$$\Delta_{k,\min}^l \leq (\vec{\xi}_i - \vec{\xi}_F) \cdot \vec{n}_k \leq \Delta_{k,\max}^l \quad , \quad k = x, y \quad (5.20)$$

$$\epsilon_{k,\min}^l \leq (\vec{\xi}_i - \vec{\xi}_F) \cdot \vec{t}_k \leq \epsilon_{k,\max}^l \quad , \quad k = x, y \quad (5.21)$$

$$(\Delta_{c,\min}^l)^2 \leq (\vec{\xi}_i - \vec{\xi}_F) \cdot (\vec{\xi}_i - \vec{\xi}_F) \leq (\Delta_{c,\max}^l)^2 \quad (5.22)$$

Pixels that are outside the bounds of all geometrical objects are not considered to represent field markings, they are invalid. This guarantees that pixels falsely identified as field markings do not reduce the quality of the solution. The number of valid pixels is referred to as $n_{\text{pix}}(\mathbb{F})$.

Equation (5.19) consists of three nonlinear equations that cannot be solved explicitly. Appendix B.7 provides a detailed derivation of these equations and the coefficients defining the problem. Except for the periodicity of $\bar{\alpha}$ and rotational symmetry of the field markings, there can be only one solution to these equations because the least squares problem has only one minimum that is global for a given pixel–object associativity. This property allows a straightforward numerical treatment of equation (5.19) with the Newton method. The only source for errors is a false pixel–object associativity that will lead to a wrong formulation of the problem. This can happen if the initial value $\xi_F^{(0)}$ is too bad.

The residual of the Newton method contains three entries

$$\mathbf{R} \stackrel{def}{=} [R_\xi \ R_\eta \ R_{\bar{\alpha}}]^T \quad (5.23)$$

```

initialize  $\xi_F^{(0)}$ 
LOOP over outer iterations ( $o$ )
  clear coefficients
  LOOP over pixels
    LOOP over object instances
      IF pixels is within object bounds
        THEN assign pixel to object
      END IF
    END LOOP over objects
    update coefficients with pixel coordinates  $(\xi_i, \eta_i)$ 
  END LOOP over pixels
  LOOP over Newton iterations ( $m$ )
    check convergence:  $|\mathbf{R}(\xi_F^{(m)})| < \text{Tol} ?$ 
    perform Newton step:  $\xi_F^{(m+1)} = \xi_F^{(m)} - \left[ \left( \frac{d\mathbf{R}}{d\xi_F} \right)^{-1} \cdot \mathbf{R} \right]_{\xi_F^{(m)}}$ 
  END LOOP over Newton iterations ( $m$ )
  check outer convergence:  $|\xi_F^{(o)} - \xi_F^{(o-1)}| < \text{Tol} ?$ 
END LOOP over outer iterations ( $o$ )
compute robot position:  $\mathbf{x}_R = \mathcal{P}\xi_F^{(o)}$ 

```

Table 5.2: Robot self localization algorithm

and it is computed as follows:

$$\mathbf{R} = \frac{dS_F}{d\xi_F} \stackrel{!}{=} 0 \quad (5.24)$$

The iteration of the solution vector is carried out according to

$$\xi_F^{(m+1)} = \xi_F^{(m)} - \left[\left(\frac{d\mathbf{R}}{d\xi_F} \right)^{-1} \cdot \mathbf{R} \right]_{\xi_F = \xi_F^{(m)}} \quad (5.25)$$

in which m is the Newton iteration counter. The initial value $\xi_F^{(0)}$ is extrapolated from the robot's previous positions with the WPA method. An absolute convergence criterion can be applied to terminate the Newton iteration:

$$|\mathbf{R}| < \text{Tol} \quad (5.26)$$

Table 5.2 lists the required steps of the field localization algorithm in form of a pseudo-code. The outer iterations are necessary because all pixels need to be associated with geometrical objects. A new solution for ξ_F may alter this associativity, which will yield a different solution ξ_F in the next outer iteration. After all coefficients according to table B.2, p. 208, have been reset to zero, the loop over all pixels begins. A nested loop over all geometrical objects determines which object instance the respective pixel is associated with. The coefficients for the respective object instance are updated by adding up the coordinates of the current pixel. After all pixels have been processed, the numerical coefficients are complete and the nonlinear set of equations according to equation (5.24) is solved with the Newton method in equation (5.25). The outer convergence can be determined by comparing the solution vectors of the previous outer iteration $\xi_F^{(o-1)}$ with the current outer iteration $\xi_F^{(o)}$ according to equation (5.27). The difference between the two vectors is a relative convergence criterion.

$$\left| \xi_F^{(o)} - \xi_F^{(o-1)} \right| < \text{Tol} \quad (5.27)$$

5.3 Ball Localization

The detection of the ball shape is carried out in bitmap coordinates of the digital video images. This is based on the assumption of little optical distortions due to the small size of the ball. The ball outline is therefore assumed to appear circular in the digital video images. A similar approach as for the robot self localization is employed to determine the position of the ball center as well as its radius in the digital image. The position in robot coordinates will be determined in a second step with use of the same lookup table as for the robot self localization.

5.3.1 Ball Position in Video Image

The solution vector for the ball shape detection in pixel coordinates in the digital video image is defined as

$$\mathbf{x}_B \stackrel{def}{=} [\mathbf{x}_B \ \mathbf{y}_B \ \mathbf{r}_B]^T, \quad (5.28)$$

in which \mathbf{x}_B and \mathbf{y}_B are the position of the ball center in the digital image and \mathbf{r}_B is the ball radius, all in units of pixels. This is illustrated in figure 5.8. The geometric distance between a pixel i and the circular ball outline in bitmap coordinates is

$$\tilde{\mathbf{d}}_i = \sqrt{(\mathbf{x}_i - \mathbf{x}_B)^2 + (\mathbf{y}_i - \mathbf{y}_B)^2} - \mathbf{r}_B \quad (5.29)$$

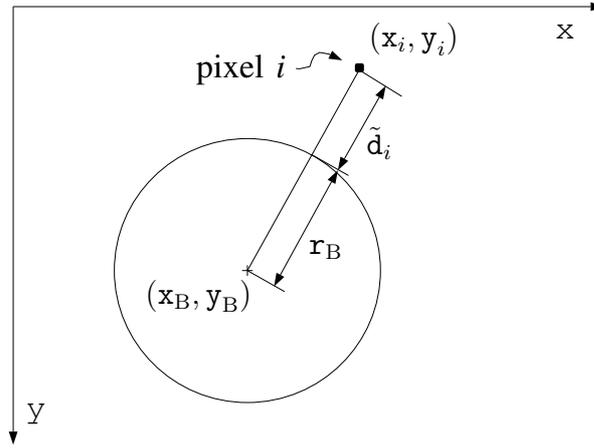


Figure 5.8: Ball geometry in video image

The sum of pixel distances and its derivatives with respect to the solution vector for ball localization over all ball pixels $n_{\text{pix}}(\mathbf{B})$ is defined as

$$S_B \stackrel{def}{=} \frac{1}{2} \sum_{i=1}^{n_{\text{pix}}(\mathbf{B})} \mathbf{d}_i^2 \stackrel{!}{=} \min \quad (5.30)$$

$$\frac{dS_B}{d\mathbf{x}_B} = \sum_{i=1}^{n_{\text{pix}}(\mathbf{B})} \left(\mathbf{d}_i \cdot \frac{d(\mathbf{d}_i)}{d\mathbf{x}_B} \right) \stackrel{!}{=} 0 \quad (5.31)$$

Using the geometrical distance $\mathbf{d}_i = \tilde{\mathbf{d}}_i$ is not practical for a solution of equation (5.31) because the square root would impede its treatment by adding nonlinear terms. Therefore, an alternative definition of the pixel

distance is used that is the square of a geometric distance and corresponds to the definition applied for field localization in equation (5.17):

$$d_i \stackrel{def}{=} (\mathbf{x}_B - \mathbf{x}_i)^2 + (\mathbf{y}_B - \mathbf{y}_i)^2 - r_B^2 \quad (5.32)$$

Equation (5.32) will not yield the exact solution as equation (5.29) but it will also minimize the average pixel distance and therefore produce a similar solution.

A detailed derivation of the solution for which equation (5.32) is used in equation (5.31) is provided in appendix B.8. It is important to mention that the solution vector \mathbf{x}_B can be obtained explicitly from the pixel coordinates \mathbf{x}_i and \mathbf{y}_i . No iterative calculation is required.

5.3.2 Ball Position in Robot Coordinates

The next step is the computation of the ball position in robot coordinates. It is possible to make use of the same pixel coordinate mapping lookup table that has previously been utilized for field localization. The ball center is provided in pixel coordinates. These coordinates are decimal numbers while the lookup table only contains entries for integer pixel coordinates. The coordinates (ξ_P, η_P) of the ball center are obtained with a bilinear interpolation of the four neighboring pixels in the lookup table. These coordinates correspond to the point P according to figure 5.9. P is located behind the ball at a distance D from the robot.

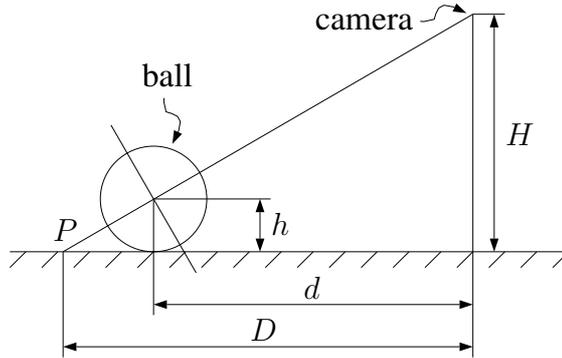


Figure 5.9: Ball projection in robot coordinates

Given the ball radius h and the camera height H , the actual ball distance d from the robot satisfies

$$\frac{D-d}{h} = \frac{D}{H} \quad (5.33)$$

This allows to express the ball distance d in terms of the known distance of point P :

$$d = D \left(1 - \frac{h}{H}\right) \quad (5.34)$$

Introducing the dimensionless ball perspective parameter

$$\chi_B \stackrel{def}{=} 1 - \frac{h}{H}, \quad (5.35)$$

the coordinates of the ball in robot coordinates can be expressed as

$$\xi_B = \chi_B \xi_P \quad (5.36)$$

$$\eta_B = \chi_B \eta_P \quad (5.37)$$

Chapter 6

Experimental Validation

This chapter describes the experimental validation of the various algorithms, methods, and components developed and presented in the previous chapters. It comprises the trajectory generation derived in chapter 3, the performance of the sensor design according to chapter 5, hardware modifications to the robot system elaborated in chapter 1, the robot motor and motion controllers described in chapter 4, the actuators dimensioned in chapter 2, and finally the robot's capabilities of solving the Two Body Problem by dribbling a ball along a predefined trajectory. The experiments are arranged by their chronological order in this chapter, not by the order in which they were described in the previous chapters. The chronological order makes most sense because the experiments depend on each other and every new test builds on the results obtained in earlier tests.

6.1 Trajectory Planning

This section examines the results produced by the two trajectory planning algorithms, the minimal curve length approach (MCL) and the circle and tangent approach (CAT) derived in sections 3.5 and 3.6, respectively. Both algorithms were implemented in Matlab apart from the robot software. The purpose was to test and evaluate both algorithms before implementing them with hardware in the loop. Besides, Matlab offers a debugging friendly high level environment for developing new algorithms. The drivecycles discussed in all following sections of this chapter were therefore created off-line with Matlab and not onboard the robot, even in experiments with the robot driving. The trajectory was created beforehand and then the table containing the robot position entries which constitute a drivecycle was communicated to the robot via the wireless CAN communication.

A time spacing of the entries in the table of 40 ms was used because the robot position calculated by the video image recognition is also sampled at this period. Since the weighted polynomial approximation described in section 3.7 is used for both interpolating the drivecycle set position and for extrapolating the actual robot position, it is useful to use the same parameter set and the same sampling period.

p	x_p / m	y_p / m
0	0.0	0.0
1	1.0	0.5
2	1.5	0.0
3	1.0	-0.5
4	-1.0	0.5
5	-1.5	-0.0
6	-1.0	-0.5
7	0.0	0.0

Table 6.1: Common points of generic MCL and CAT trajectories

It was the goal in this section to create two trajectories with the two methods that have as many common features as possible to elaborate the most substantial differences, advantages and shortcomings. The geometry of the trajectory was chosen with respect to two criteria. First, it had to be complex enough to validate the robot's agility in dribbling a ball. That means the curvature needed to change its absolute value and its direction at least one time. In contrast to that, it was considered simple to drive along a circle with constant curvature. Second, the trajectory had to fit on a small area, because space was limited in the laboratory. A compact shape of the trajectory also allows a more detailed view in a diagram. The selected geometry fulfills the two criteria very well, it is that of the figure *eight*. It has two straight sections with zero curvature at its central node, and there are two loops with opposite non-zero curvature. The *eight* cycle was therefore chosen as benchmark trajectory for all experiments that are described in this chapter, including the numerical comparison of the two generation methods discussed in this section. Both methods used the concept of the generic trajectory which served as a guideline to construct the trajectories for both robot and ball. The generic trajectories in both methods were designed to pass through a number of common points. This way, the two methods produced trajectories that were similar enough to allow a direct comparison of significant features, while each of the trajectories exhibited its individual aspects. Since the MCL method offers more degrees of freedom than the

CAT method as far as curvature is concerned, it was decided to use a curve radius of 0.5 m for the two loops of the *eight* course created with the CAT method. This radius leaves sufficient space to surround an opposing robot in a RoboCup contest. The centers of the two circular loops were located at $(x = \pm 1 \text{ m}, y = 0 \text{ m})$ for the CAT method. In order to compare the resulting trajectories produced with the two methods, it was decided that the points listed in table 6.1 needed to be passed by both methods. Start and end point was the central node located at the origin. Both methods used the robot's physical parameters listed in table 6.6, p. 127.

6.1.1 Minimal Curve Length Method

This section discusses the creation and the properties of a sample trajectory created with the minimal curve length (MCL) approach derived in section 3.5. In a first attempt, the $n_p = 8$ points listed in table 6.1 were used as control points to create the trajectory shape. The generic trajectory starts at the origin and departs in both positive x - and positive y -direction. The first loop is a right curve and it is located in the first and fourth quadrant, ending at the origin. The trajectory shape is symmetric with respect to the origin. Therefore, it is not necessary to include the origin in the list of control points. The origin is a point of inflection because the second loop is a left curve. It is located in the second and third quadrant and ends at the origin. These two loops constitute one full round along the *eight* course.

Trajectory Shape The shape was determined in two passes. The first pass only minimized the curve length with the weight factor $w^* = 0$ according to equation (3.30), p. 49. The second pass included the optimization of the curvature penalty term κ^* defined in equation (3.34), p. 50. The weight factor $w^* = 0.5$ was used in this pass. The convergence criterion according to table 3.2, p. 51, was used, the tolerance was set to $\text{Tol} = 10^{-3}$. Equations (3.32) and (3.34), p. 50, were integrated numerically using a second order integration scheme, both had the physical base unit m. Refer to appendix B.1 for a detailed discussion of the numerical treatment of the involved equations.

For the first pass, the elements r_p of the solution vector \mathbf{r} were initialized with equally spaced values:

$$\mathbf{r}^{(0)} = [0.143, 0.286, 0.429, 0.571, 0.714, 0.857] \quad (6.1)$$

Note that $r_0 = 0$ and $r_7 = 1$ are not contained in \mathbf{r} because they always represent start and end of the trajectory. After 7 iterations, the first pass converged to the solution

$$\mathbf{r}^{(7)} = [0.126, 0.244, 0.373, 0.627, 0.756, 0.874] \quad (6.2)$$

with a total curve length of $s^* = 7.733$ m. The second pass was initialized with the solution of the first pass. It failed to converge after 20 additional iterations with the solution

$$\mathbf{r}^{(27)} = [0.121, 0.238, 0.374, 0.626, 0.762, 0.879]. \quad (6.3)$$

Its trajectory geometry did not differ significantly from the solution of the first pass. A lot of effort was spent in the optimization of this trajectory, including a variation of the control points and the weight factor w^* . It turned out that the problem was badly conditioned with the specification of the control points listed in table 6.1. It seemed that three control points per loop imposed a constraint on the curvature that was so strong that the curvature penalty term in equation (3.34), p. 50, did not allow a convergence. The numerical integration error probably contributed to the problem but it was not the only problem. This was tested by refining the integration step size.

Therefore, different control points were selected that allowed a stable convergence of both passes. Only four control points were defined in addition to the start and end at the origin, two control points for each loop. Two control points per loop seems to be a rule of thumb for successful trajectory generation with the MCL method, it removes the choking constraint on the curvature penalty term. The new control points are listed in table 6.2. They were chosen such that the trajectory still hits the common points defined in table 6.1.

p	x_p / m	y_p / m
0	0.0	0.0
1	1.28	0.43
2	1.28	-0.43
3	-1.28	0.43
4	-1.28	-0.43
5	0.0	0.0

Table 6.2: MCL trajectory control points

Again, the elements r_p of the solution vector \mathbf{r} were initialized with equally spaced values for the first pass:

$$\mathbf{r}^{(0)} = [0.200, 0.400, 0.600, 0.800] \quad (6.4)$$

After 5 iterations, the first pass converged to the solution

$$\mathbf{r}^{(5)} = [0.111, 0.286, 0.714, 0.889] \quad (6.5)$$

with a total curve length of $s^*=7.701$ m and a smallest local curve radius $\hat{\rho}=0.397$ m. The second pass required only 2 additional iterations to find the solution

$$\mathbf{r}^{(7)} = [0.107, 0.283, 0.717, 0.893] \quad (6.6)$$

with a total curve length of

$$s^* = 7.705 \text{ m} \quad (6.7)$$

and a smallest local curve radius $\hat{\rho}=0.395$ m. That means the trajectory length was slightly increased by the curvature penalty term, and at the same time, the minimum curve radius was decreased. In this case, the second pass did not improve the trajectory shape, but it even reduced its quality with respect to the two aspects curve length and minimum curve radius. Despite this result, the curvature penalty term is still justified, because for other geometries that are not discussed here, a significant improvement could be achieved in the second pass. The value of the curvature penalty term was reduced during the second pass from $\kappa^{*(5)}=1.284 \text{ m}^{-2}$ to $\kappa^{*(7)}=1.275 \text{ m}^{-2}$. All trajectories created with the MCL method in this chapter are based on the control points in table 6.2. This includes all experiments with the robot driving. The only difference in a few experiments was that a second round was added by duplicating control points 1 to 4, for example in the dribbling experiment described in section 6.6. This had some effects on the curvature because in a trajectory with two rounds, the geometry of the first loop was not identical with the geometry of the third loop. Likewise, the second loop did not coincide with the fourth loop. In addition to that, neither loop of the two round trajectory was identical with a loop of the single round trajectory. This had some effects on the robot's dynamics. They can be seen indirectly in figures 6.44, p. 156, and 6.46, p. 159, by observing the calculated ball and robot trajectories. Note that the permanently increasing generic velocity also has an effect on the robot's orientation if $\psi \neq 0$, and it therefore affects the position of both ball and robot. This effect can be neglected when loop 2 is compared to loop 4 because the generic velocity has almost reached its final value in both loops.

Trajectory Speed The constraints for the trajectory speed algorithm are expressed in form of a parameter vector $\mathbf{\Lambda}$ for convenience. It contains the four parameters that determine the velocity profile along the generic trajectory, maximum velocity v_{\max} , lateral acceleration a_{lat} , longitudinal acceleration a_{acc} , and longitudinal deceleration a_{dec} :

$$\mathbf{\Lambda} \stackrel{\text{def}}{=} \left[v_{\max} / \frac{\text{m}}{\text{s}}, a_{\text{lat}} / \frac{\text{m}}{\text{s}^2}, a_{\text{acc}} / \frac{\text{m}}{\text{s}^2}, a_{\text{dec}} / \frac{\text{m}}{\text{s}^2} \right] \quad (6.8)$$

Likewise, a parameter vector $\mathbf{\Gamma}_{\text{MCL}}$ is defined, it contains the weight factor ψ that determines the location of the robot relative to the generic trajectory, and the viscous damping ratio δ :

$$\mathbf{\Gamma}_{\text{MCL}} \stackrel{\text{def}}{=} [\psi, \delta / \text{s}^{-1}] \quad (6.9)$$

This section discusses a trajectory that was created with the parameter vectors $\mathbf{\Lambda} = [1.5, 2.5, 1.5, 0.5]$ and $\mathbf{\Gamma}_{\text{MCL}} = [0.8, 5]$. These are the parameters that were used for the fast dribbling experiment in section 6.6. The only difference is that here only one round of the *eight* loop was generated while the dribbling experiment used two rounds.

The velocity is limited by the smallest curve radius $\hat{\rho}$ according to equation (3.49), p. 55, to a maximum value of 0.993 m/s. This is smaller than the specified maximum velocity v_{max} and therefore, equation (3.50), p. 55, leads to

$$v_{\infty} = 0.993 \frac{\text{m}}{\text{s}} \quad (6.10)$$

with the trajectory end time computed with equation (3.39), p. 53:

$$t^* = 9.885 \text{ s} \quad (6.11)$$

Appendices B.2 and B.3 contain a detailed discussion of the numerical treatment involved in the computation of the velocity profile along the generic trajectory speed and the robot position relative to the generic trajectory, respectively.

Discussion of Results Figure 6.1 plots the generic trajectory together with the robot and the ball trajectory in an x - y coordinate system. Due to the small trajectory weight factor $\psi = 0.2$, the ball trajectory is located inside and very close to the generic trajectory while the robot is a fair bit to the outside. The six control points listed in table 6.2 are plotted as empty circles. Three lines connecting some simultaneous robot and ball positions are added to each of the two loops. These lines represent the robot's orientation, their angles with the respective tangential directions are relatively small. This is a result of the high viscous damping ratio $\delta = 5 \text{ s}^{-1}$. It is a special property of this trajectory that the minimum curve radius $\hat{\rho}$ does not occur at the zenith of the two loops. Instead, there are two locations per loop at which the curve radius $\rho(r)$ of the generic trajectory reaches a local minimum. They are close to the four non-zero control points. Between these two locations, the curve radius has a local maximum which is located close to the zenith.

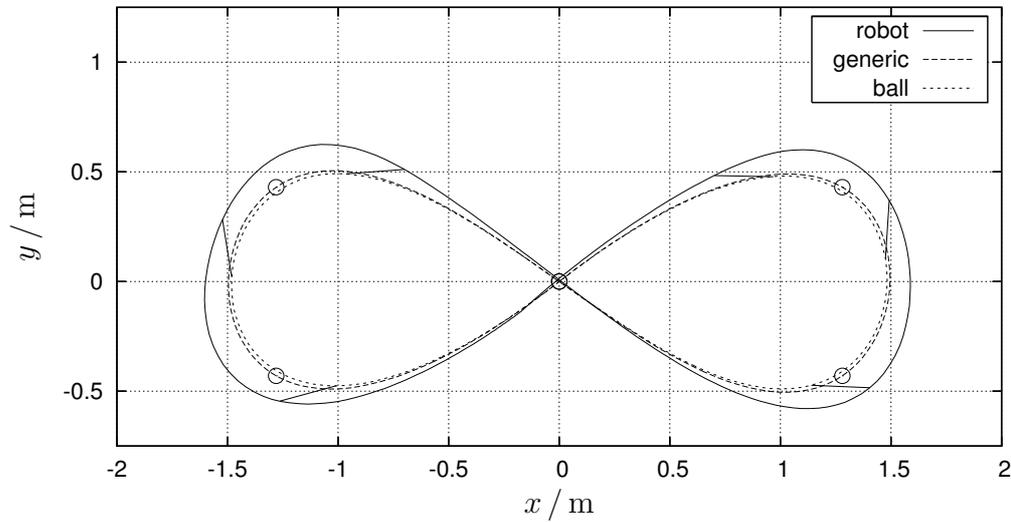
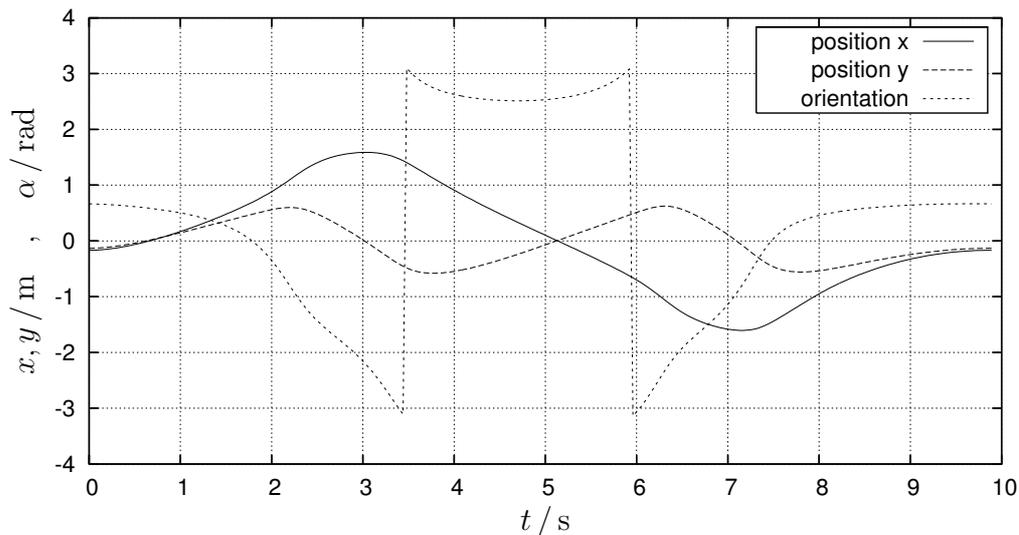
Figure 6.1: MCL trajectory: *Eight* course

Figure 6.2: MCL trajectory: Position and orientation

Figure 6.2 plots the robot coordinates with respect to time. The orientation angle α is clipped to the interval $[-\pi, \pi]$.

Figure 6.3 plots the robot velocity and total acceleration $a = \sqrt{\ddot{x}_R^2 + \ddot{y}_R^2}$ with respect to time. At the start, the robot accelerates with the longitudinal acceleration $a_t = a_{\text{acc}}$ of the generic trajectory. This acceleration is quickly reduced according to figure 3.4, p. 53. After the generic trajectory has gained

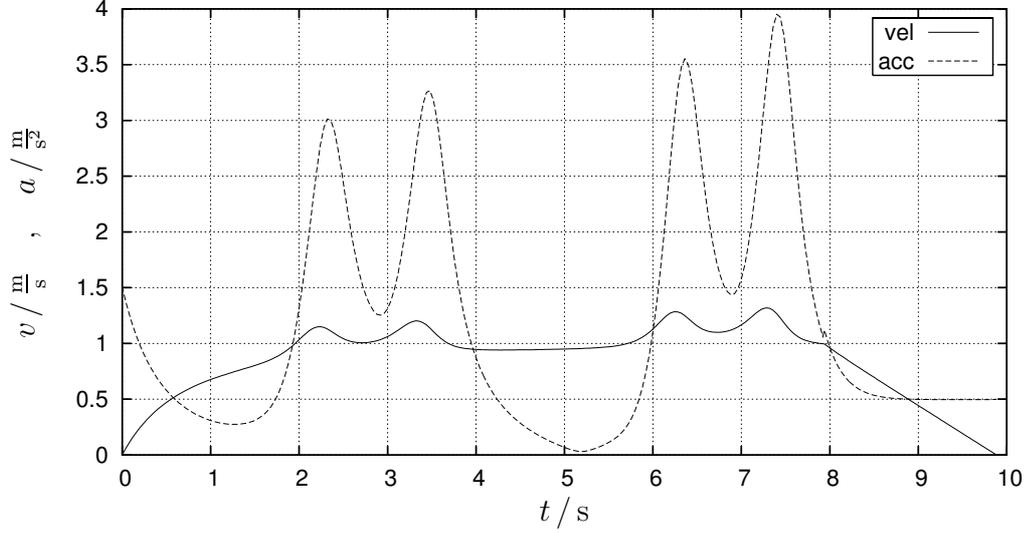


Figure 6.3: MCL trajectory: Velocity and acceleration

most of its final velocity at about 1.5s, the longitudinal acceleration a_t has almost dropped to zero. From this time on, the total acceleration is dominated by its lateral component a_n . The four peaks in the acceleration clearly indicate the locations of the minimum curve radii $\hat{\rho}$. In addition to that, the robot acquires a maximum angular speed around the respective points on the generic trajectory directly before the four minimum curve radius locations. This increases the robot's translational velocity as can be seen in figure 6.3, and adds a second contribution to the lateral acceleration. At the end, the velocity is ramped down with a constant deceleration $a_t = -a_{\text{dec}}$, starting at about 8s. Note that the ramp down was added to the velocity profile artificially to allow the robot to stop softly at the end, they are not part of the velocity profile presented in section 3.5.2. The acceleration and higher order derivatives of $s(t)$ therefore undergo a discontinuity at the beginning of the velocity ramp. This is accepted because the robot must stop at the end. The total acceleration a approaches the deceleration value a_{dec} at the end of the trajectory because both the original longitudinal acceleration a_t and the lateral acceleration a_n approach zero.

Figure 6.4 shows the angular speeds $\dot{\varphi}$ for the three wheels. They are computed according to equations (1.8) and (1.9), p. 16. Wheels 1 and 2 rotate at relatively constant speed during most of the time. Wheel 3 is subjected to many transients, it takes care of steering the robot. This is not always the case, but it happens in this trajectory, because the high damping ratio δ causes the robot to be oriented in almost tangential direction with

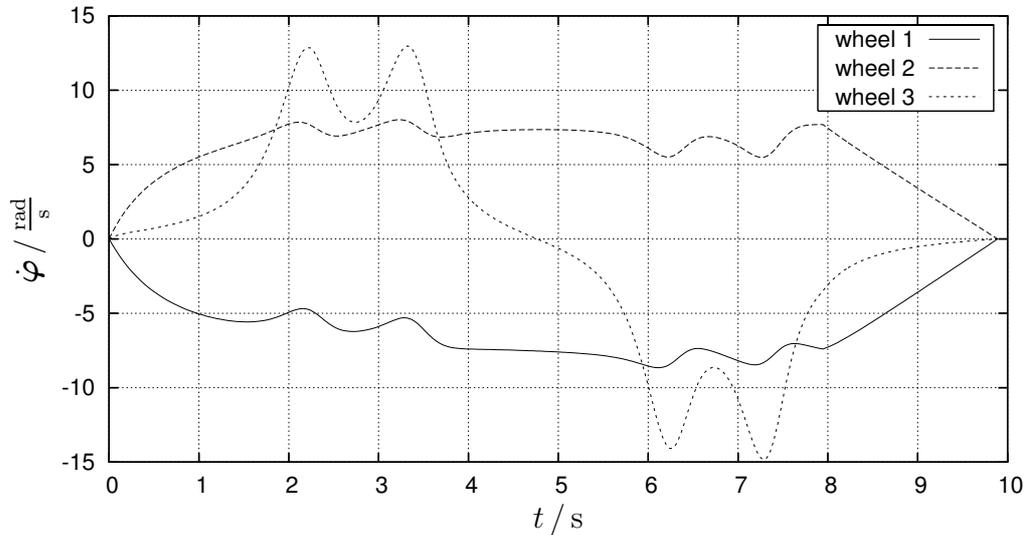


Figure 6.4: MCL trajectory: Wheel speeds

wheel 3 defining the orientation. It was mentioned before that there are four locations where the curve radius reaches a local minimum, and that they cause peaks in the robot's angular speed $\dot{\alpha}$. These peaks are mostly managed by wheel 3 as the graph indicates. The maximum absolute angular speed $|\dot{\varphi}| = 14.8 \frac{\text{rad}}{\text{s}}$ occurs at wheel 3, this is only one third of the possible maximum speed used for dimensioning the motors.

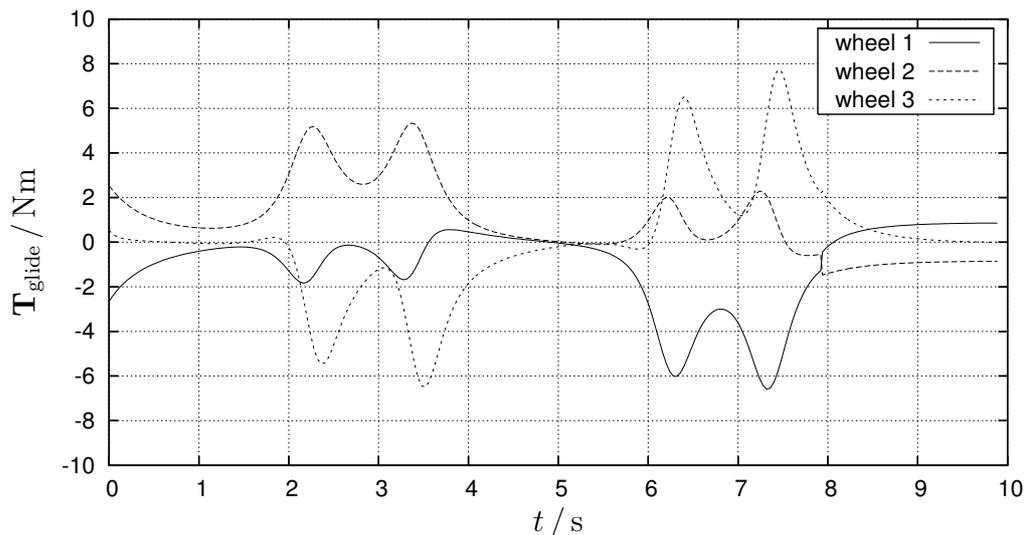


Figure 6.5: MCL trajectory: Wheel torques

Figure 6.5 plots the wheel torques $\mathbf{T}_{\text{glide}}$ according to equation (1.54), p. 22, with respect to time. They represent the circumferential forces between wheels and floor. The total wheel torques \mathbf{T} which include the rotational moments of inertia of the wheels are not plotted here, they represent the motor torques. In this drivecycle, the values of \mathbf{T} were typically 10 % higher than the respective values of $\mathbf{T}_{\text{glide}}$. Unlike the wheel speeds, all three wheels show significant transient behavior in their torques. Four peaks indicating the four locations of minimum curve radius can clearly be identified again. The maximum absolute wheel torque is $|\hat{\mathbf{T}}_{\text{glide}}|=7.7\text{ Nm}$ at wheel 3. The largest absolute value of the total wheel torque is $|\hat{\mathbf{T}}|=8.8\text{ Nm}$. This comes close to the maximum value of 9.6 Nm used for motor dimensioning in table 2.4, p. 32.

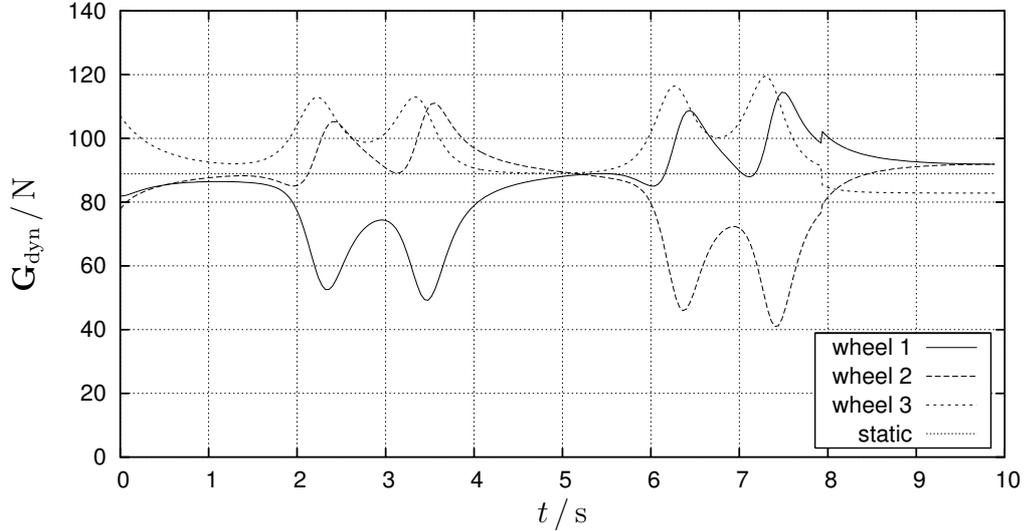


Figure 6.6: MCL trajectory: Wheel loads

Figure 6.6 plots the dynamic wheel loads \mathbf{G}_{dyn} computed with equation (1.57), p. 22, with respect to time. For comparison, the static load defined in equation (1.35), p. 18, is added. Wheel 3 always has a positive load shift $G_{3,\text{shift}}$ except at the final deceleration phase. At the beginning, the load shifts to wheel 3 because it is the rear wheel in driving direction while the robot accelerates. In the following, wheel 3 always faces towards the outside of the curves and therefore, its dynamical load increases. Wheels 1 and 2 have both positive and negative load shifts. In the first right hand bend, wheel 1 is the inner wheel while wheel 2 is the outer wheel. Therefore, the dynamic load on wheel 1 decreases while the load on wheel 2 increases. In the second left hand bend, it is the other way round. The smallest dynamic

wheel load is $\widehat{\mathbf{G}}_{\text{dyn}}=40.0\text{ N}$, this seems still uncritical. A definite problem would arise if a wheel load became negative because in that case the respective wheel would lift off and the robot might fall over.

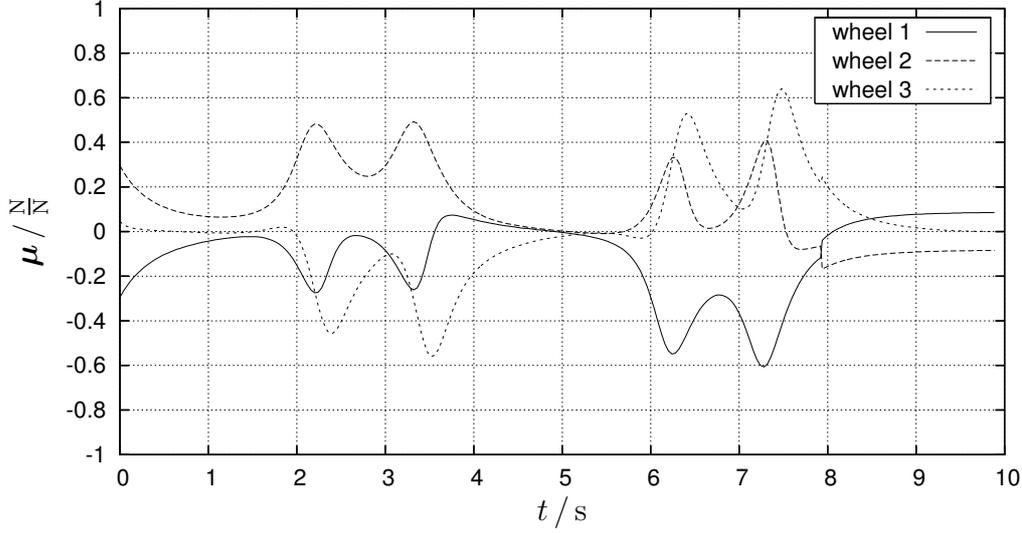


Figure 6.7: MCL trajectory: Friction coefficients

Combining the wheel torques $\mathbf{T}_{\text{glide}}$ and the dynamic wheel loads \mathbf{G}_{dyn} , the coefficient of friction $\boldsymbol{\mu}$ according to equation (1.59), p. 22, can be computed. The resulting coefficients are plotted for each wheel in figure 6.7. The sign of the friction coefficient at each wheel follows that of the wheel torques because all dynamic wheel loads are always positive. The maximum absolute value $|\widehat{\boldsymbol{\mu}}|=0.64$ occurs for wheel 3 at the last of the four points with minimum curve radius. This is indicated by the peak in the graph of wheel 3 at 7.5 s. Note that the fast dribbling experiment used the same control points and the same parameters $\boldsymbol{\Lambda}$ and $\boldsymbol{\Gamma}_{\text{MCL}}$, but the calculated maximum coefficient of friction was $|\widehat{\boldsymbol{\mu}}_{\text{MCL}}|=0.73$ there. The difference was caused by the second round that was added for the dribbling experiment. This created a different curvature for the complete trajectory. Especially at 7.5 s when the maximum value $|\widehat{\boldsymbol{\mu}}|$ is reached, the local curve radius is $\hat{\rho}=0.353\text{ m}$ for the two round trajectory. This compares to $\hat{\rho}=0.395\text{ m}$ for the single round discussed here, so the smaller curve radius creates the higher coefficient of friction.

Figure 6.8 shows the mechanical power \mathbf{P} of each wheel according to equation (1.32), p. 18, as well as the total power according to equation (1.30), p. 18, which is the sum of the three wheel powers. The powers of each wheel and the total power obtain both positive and negative signs during the drivecycle. A positive sign means that the kinetic energy of the robot is

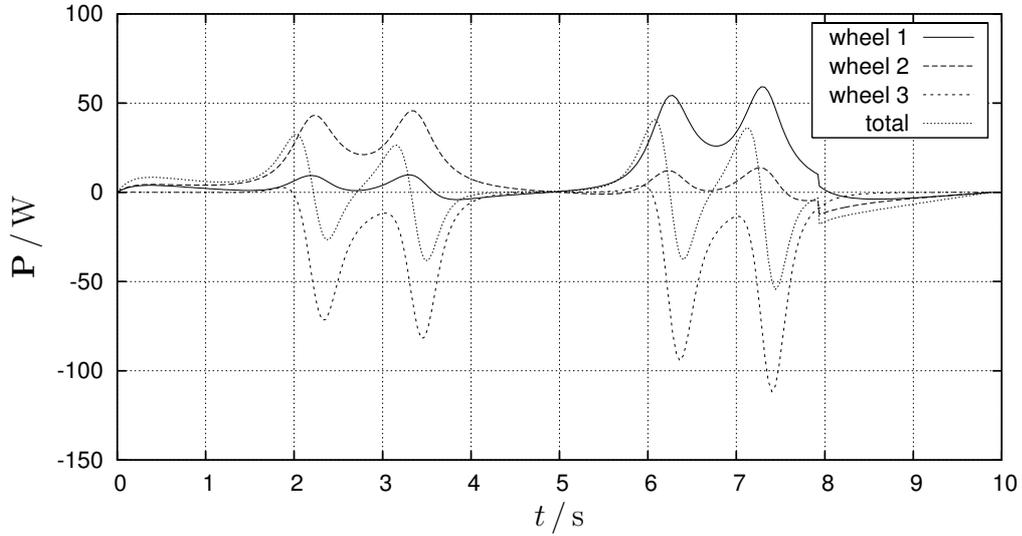


Figure 6.8: MCL trajectory: Wheel powers

increased while a negative sign means that the kinetic energy of the robot is reduced. The power at wheel 3 is negative most of the time, that means motor 3 is generating electric energy which is fed back to the battery. The maximum negative power occurs also at wheel 3, it is -112 W . This is only one quarter of the power for which the motors were dimensioned according to table 2.4, p. 32. The maximum positive power is consumed by wheel 1, it is 59 W . The total power ranges between the values -54 W and 41 W . This is the power that goes into and out of the battery if there are no efficiency losses.

6.1.2 Circle and Tangent Method

This section discusses the creation and the properties of a sample trajectory created with the circle and tangent (CAT) approach derived in section 3.6.

Trajectory Shape As mentioned before, a set of common points was defined through which both sample trajectories had to pass. These are listed in table 6.1. The CAT method allows only one solution that passes through these points. It requires two circles around the points ($x = \pm 1\text{ m}$, $y = 0\text{ m}$) with respective radii of $\mp 0.5\text{ m}$. The control points used by the CAT method are listed in table 6.3. The first control point is the origin, it has zero radius because it is the actual location where the trajectory starts. The second control point is a circle that is to be surrounded clockwise and therefore has

a negative radius. The third control point is an anticlockwise circle, hence it has a positive radius. The fourth control point denotes the trajectory end point, it has zero radius.

p	x_p / m	y_p / m	r_p / m
0	0	0	0
1	1	0	-0.5
2	-1	0	0.5
3	0	0	0

Table 6.3: CAT trajectory control points

The total curve length of the generic trajectory is

$$s_{\text{CAT}} = 7.653 \text{ m}, \quad (6.12)$$

it is slightly shorter than the one created with the MCL method. Refer to appendix B.4 for a detailed description of the intermediate steps involved in the generation of the trajectory shape with the CAT method. Note that in contrast to the MCL method, the CAT method repeats the trajectory geometry if a second round is added. This can be seen indirectly in figures 6.45, p. 156, and 6.47, p. 159, by observing the calculated ball and robot position.

Trajectory Speed In analogy to the MCL method, the constraints for the trajectory speed algorithm are combined in a parameter vector $\mathbf{\Lambda}$ defined in equation (6.8). Likewise a parameter vector $\mathbf{\Gamma}_{\text{CAT}}$ is defined. It contains the weight factor ψ that determines the location of the robot relative to the generic trajectory, the viscous damping ratio δ , and a third parameter d_0 that did not exist for the MCL method:

$$\mathbf{\Gamma}_{\text{CAT}} \stackrel{\text{def}}{=} [\psi, \delta / \text{s}^{-1}, d_0 / \text{m}] \quad (6.13)$$

The parameter d_0 was introduced as a result of the dribbling experiments, it defines a deceleration free zone at the end of every straight section that is followed by a circular section. It was observed that the most critical spots for losing the ball were the transitions points from a straight to a circular section. According to the construction rules for the velocity profile defined on page 60, the velocity of the generic trajectory will be increased whenever possible. Consequently, the velocity is often too high before a circular section is entered. Therefore, it is required to decelerate. This deceleration creates an additional disturbance on the ball and increases the chances that the ball gets lost while a curve is entered. This effect is described in section 6.6.

The trajectory that is discussed here was created with the parameter vectors $\mathbf{\Lambda} = [1.5, 2.5, 1.5, 0.5]$ and $\mathbf{\Gamma}_{\text{CAT}} = [0.8, 5, 0.2]$. Like for the MCL sample trajectory, these are the same parameters as in the fast dribbling experiment in section 6.6. The only difference is that here only one round of the *eight* loop was generated while the dribbling experiment used two rounds. The smallest curve radius here is $r = 0.5$ m by definition, it limits the velocity in the curved section according to equation (3.66), p. 60, to a maximum value of

$$v_{\text{lat}} = 1.118 \frac{\text{m}}{\text{s}} . \quad (6.14)$$

This is smaller than the specified maximum velocity v_{max} and therefore, an acceleration on the middle straight section between the circular sections is possible. The complete drivecycle lasts

$$t_{\text{CAT}} = 8.109 \text{ s} . \quad (6.15)$$

The CAT trajectory is about 18 % faster than the MCL method. This can be explained by four reasons. First, the curve velocity v_{lat} due to lateral acceleration for the CAT method is 12.6 % higher than the respective value v_{∞} for the MCL method. This is because of the larger CAT curve radius. Second, the CAT trajectory foresees a limiting velocity v_{lat} that is constant throughout both circular sections, while the velocity of the MCL method gradually increases towards its asymptotic limit v_{∞} . Third, the CAT method allows to accelerate with the maximum constant value a_{acc} at the start, while the longitudinal acceleration \ddot{s} of the MCL method rapidly decreases. Fourth, the CAT method inserts an intermediate sprint in the straight third section which the MCL method does not. If this sprint is suppressed in the CAT trajectory by limiting the maximum velocity v_{max} to the value of v_{lat} , the end time increases to 8.336 s which is still about 16 % faster than the MCL trajectory.

The arc length used in the secant method is identical for both circular sections, it is calculated according to equation (3.79), p. 65, and has the value

$$\hat{s} = 0.356 \text{ m} . \quad (6.16)$$

The angle between robot orientation and trajectory tangent in the circular section is

$$\zeta = 20.4^{\circ} . \quad (6.17)$$

Refer to the discussion of figure 6.11 for more details. The process of generating the velocity profile along the generic trajectory is described thoroughly in appendix B.5.

Discussion of Results The resulting generic trajectory is geometrically simple, it is symmetrical with respect to both coordinate axes. The two straight tangent lines cross each other in the origin and have an angle of $\pm 30^\circ$ with the x -axis. Figure 6.9 shows the resulting generic trajectory together with ball and robot trajectory. The four control points listed in table 6.3 are plotted as empty circles. The four tangent points of the generic trajectory are located at the coordinates $(x = \pm 3/4 \text{ m}, y = \pm \sqrt{3}/4 \text{ m})$, they are plotted with filled dots. The shapes of both ball and robot trajectory are symmetric with respect to the y -axis. Since the same parameters were used as with the MCL method, there are some similarities. The ball trajectory is located inside and very close to the generic trajectory while the robot is a fair bit to the outside. Again, some lines connecting the respective robot and ball positions are added to the graph, they reflect the value of the angle ζ . It can be seen that the secant method smoothly introduces the robot into the circular section. Approaching a curve, the robot departs from the generic trajectory almost 0.5 m before the respective tangent points at $y = 0.433 \text{ m}$ and swings to its outer orbit. When the robot leaves the circular sections at the tangent points at $y = -0.433 \text{ m}$, it smoothly returns back to the generic trajectory. Note that the robot does not swing out when leaving a circle like it does when entering it. The reason is the secant method which looks ahead of a given point T on the generic trajectory to compute the orientation α , this is illustrated in figure 3.9, p. 64. Then the robot position is determined by going backwards from point T in that direction by the distance $\psi \xi_0$ as expressed in equation (3.72), p. 62.

Figure 6.10 plots the robot coordinates with respect to time. The orientation angle α is clipped to the interval $[-\pi, \pi]$.

Figure 6.11 shows the velocity squared–distance diagram for both generic and robot trajectory. It reflects the elements and construction rules of the velocity profile of the generic trajectory. The locations of the four tangent points have been added to the diagram with filled dots and vertical lines. They denote the limits between straight and circular sections according to figure 3.7, p. 62. Figure 6.11 has many similarities with figure 3.7, p. 62, both represent three straight sections interspersed with two circular sections. In both figures, the velocity constraint v_{\max}^2 is larger than the constraint v_{lat}^2 in the curved sections which allows an extra acceleration in the third section.

In the first section in figure 6.11, the robot permanently accelerates to a velocity $v^2 = 1.44 \text{ m}^2/\text{s}^2$ before it has to decelerate to meet the constraint v_{lat}^2 in the second section. Two things can be observed with respect to this deceleration period in the first section between 0.48 m and 0.67 m. First, it ends 0.2 m before the tangent point is reached at 0.87 m, this creates the deceleration free zone defined by the parameter d_0 . Second, the velocity

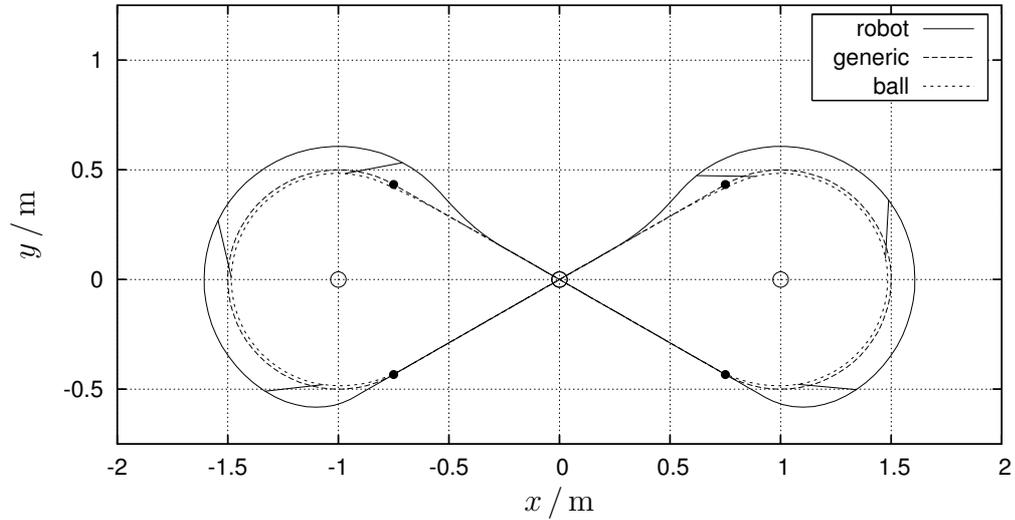
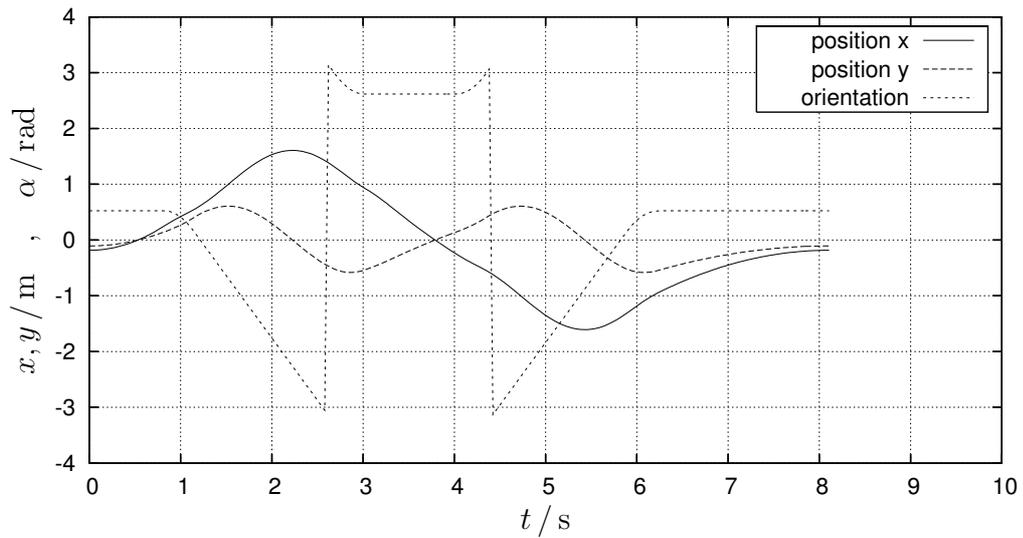
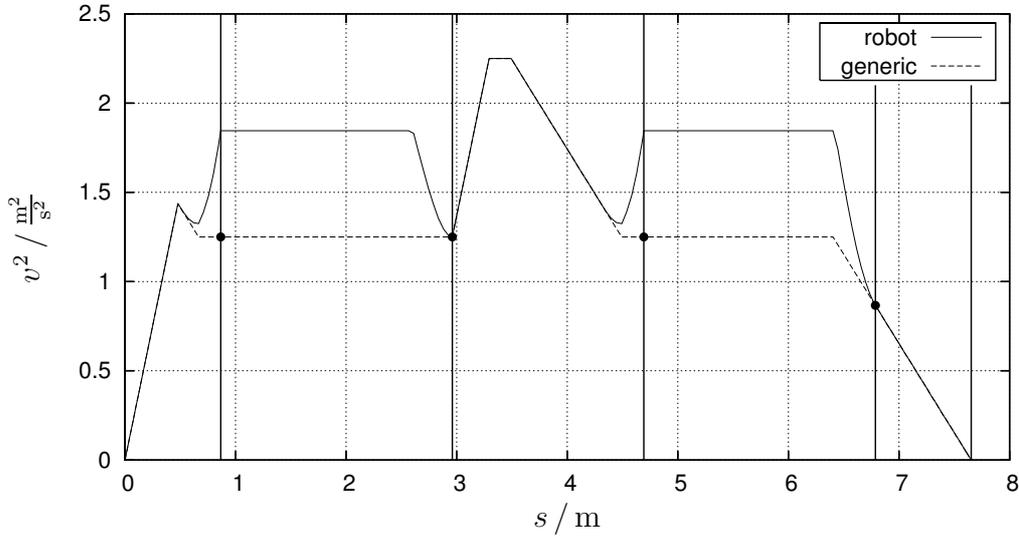
Figure 6.9: CAT trajectory: *Eight* course

Figure 6.10: CAT trajectory: Position and orientation

of the robot starts increasing while that of the generic trajectory is still decreasing. The reason for that can be found in the secant method again. The arc length \hat{s} reaches across the deceleration free zone d_0 because it is almost twice as long. Therefore, the robot starts swinging outwards before the deceleration free zone is reached on the generic trajectory, and this adds a velocity component to the robot trajectory. In section 2, the robot's velocity

Figure 6.11: CAT trajectory: Velocity²–distance diagram

is significantly higher than that of the generic trajectory. This is because the robot's orbit has a larger radius than the generic trajectory, but the robot must travel at the same angular speed around control point 1. At the end of section 2, the robot's velocity quickly approaches that of the generic trajectory because on the subsequent straight section, the robot is moving on the generic trajectory with the same velocity. In the third section, the robot accelerates until the maximum velocity limit v_{\max}^2 is reached, it travels for a short distance at this velocity before it starts decelerating to meet the velocity constraint v_{lat}^2 for the following section 4. The fourth section is identical to the second, only that now the velocity is reduced before the end of the section is reached. This is necessary to be able to stop at the end of section 5.

Figure 6.12 plots the robot velocity and total acceleration $a = \sqrt{\ddot{x}_R^2 + \ddot{y}_R^2}$ with respect to time. The velocity profile was discussed in the previous figure. Therefore, only the acceleration will be mentioned here. At the start, the robot accelerates with the constant longitudinal acceleration $a_t = a_{\text{acc}}$ of the generic trajectory. After 1.14 s, the robot has reached its orbit in the second section, and the total acceleration contains only the lateral component $a_n = 3.04 \text{ m/s}^2$. At 3.01 s, the robot enters the straight third section with consecutive acceleration $a = a_{\text{acc}}$, constant velocity $a = 0$, and deceleration $a = a_{\text{dec}}$. Section 4 is a repetition of section 2. At the end, the velocity is ramped down with a constant deceleration $a = a_{\text{dec}}$. The peak accelerations reach a value $\hat{a} = 4.24 \text{ m/s}^2$. They occur at the end of the two circular sec-

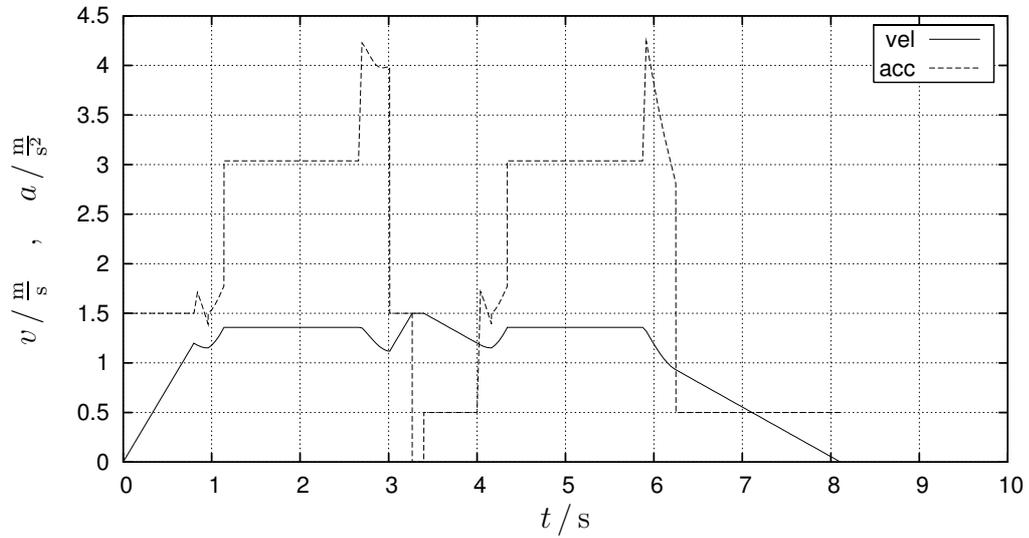


Figure 6.12: CAT trajectory: Velocity and acceleration

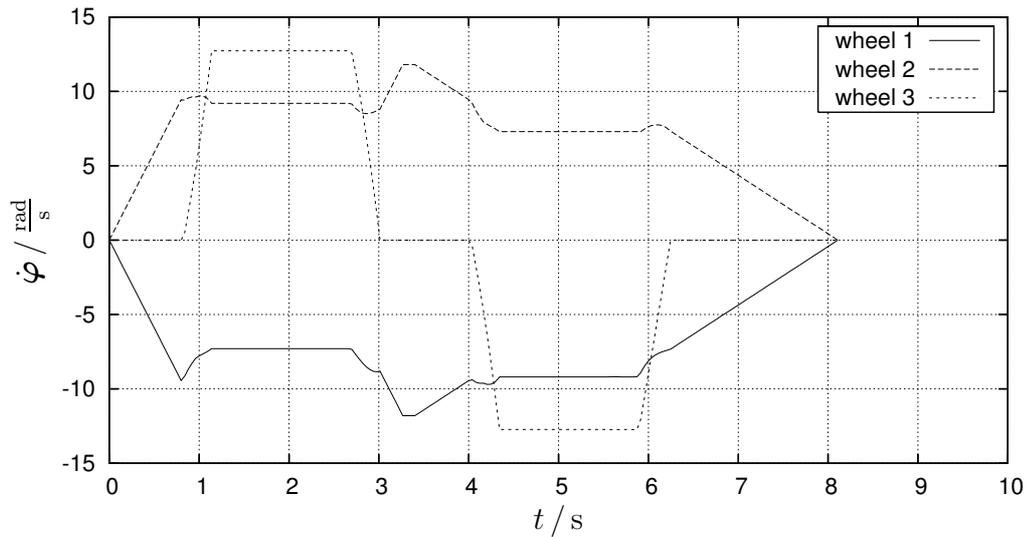


Figure 6.13: CAT trajectory: Wheel speeds

tions when the robot leaves its orbit towards the inside to enter the straight section. This reduces the local curve radius of the robot trajectory while the robot's velocity only decreases a little.

Figure 6.13 shows the angular speeds $\dot{\varphi}$ for the three wheels during the complete drivecycle, they are computed according to equations (1.8) and (1.9), p. 16. Like in the MCL trajectory, wheels 1 and 2 drive at rel-

atively constant speed while wheel 3 takes care of steering the robot and changes its direction of rotation between the two loops. The graphs of the wheel speeds in figure 6.12 reflect the velocity profile. Except for some transients, the velocity is either constant or it ramps up or down with a constant acceleration, and so do the wheel speeds. Again, the maximum absolute angular speed occurs at wheel 3, its value is $|\hat{\varphi}|=12.7$ rad/s. This is a bit less than the maximum wheel speed in the MCL trajectory.

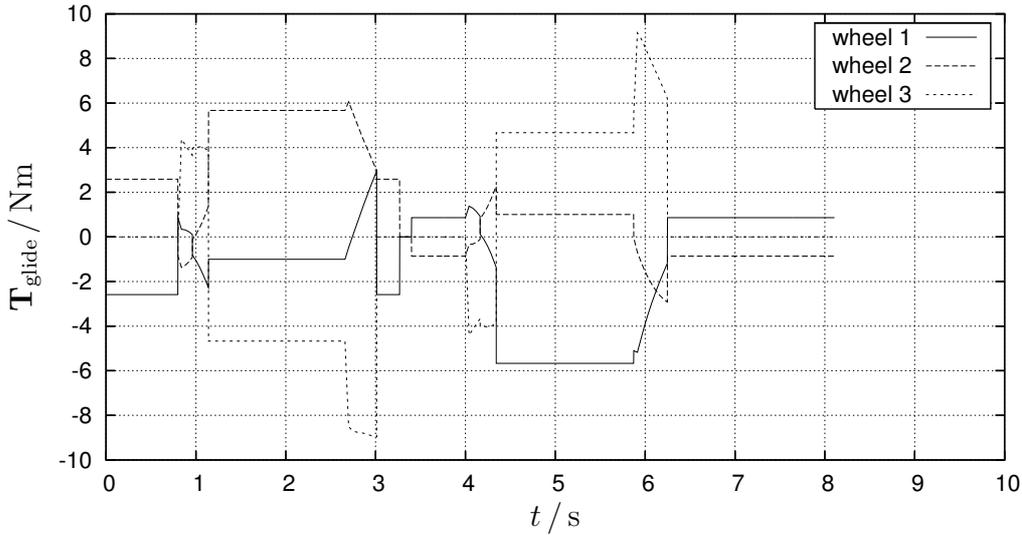


Figure 6.14: CAT trajectory: Wheel torques

Figure 6.14 plots the wheel torques $\mathbf{T}_{\text{glide}}$ according to equation (1.54), p. 22, with respect to time. Like in the MCL trajectory, the values of \mathbf{T} including the wheels' moments of inertia were typically 10% larger than the respective values of $\mathbf{T}_{\text{glide}}$. The graphs of the wheel torques neatly reflect the periods of time with acceleration, constant velocity, and deceleration. The maximum absolute wheel torque is $|\hat{\mathbf{T}}_{\text{glide}}|=9.1$ Nm at wheel 3. The largest absolute value of the total wheel torque is $|\hat{\mathbf{T}}|=10.4$ Nm, this exceeds the value 9.6 Nm used for motor dimensioning in table 2.4, p. 32. Note that according to table 2.5, p. 34, this wheel torque can still be handled by the motors because the wheel velocity is not very large. In addition to that, motor 3 is in generating mode at the time of both torque peaks because the angular speed of wheel 3 has an opposite sign compared to the torque. It is interesting to mention that both this sample drivecycle and the pure translational load case for dimensioning the motors produce their extreme values at wheel 3, that is both the maximum wheel speed and the maximum wheel torque. The MCL trajectory produced the same result.

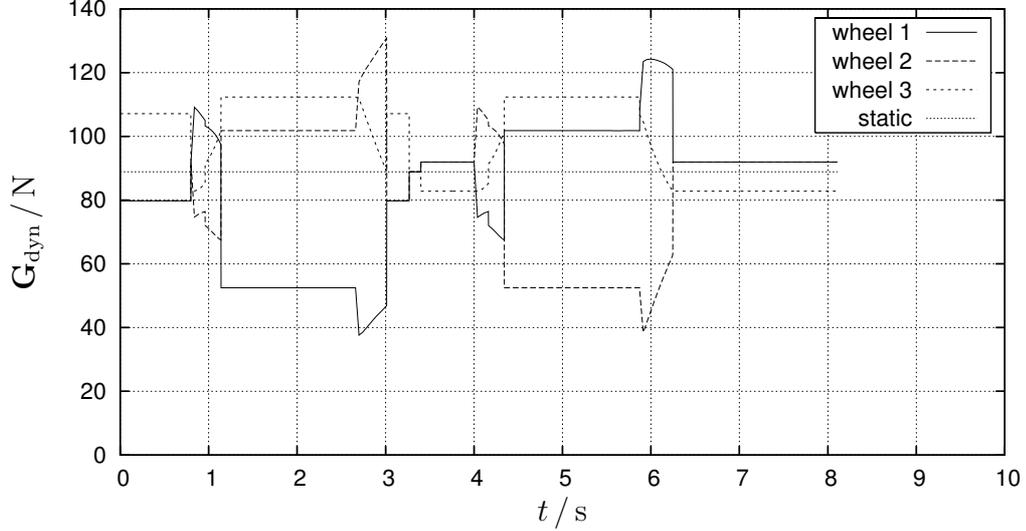


Figure 6.15: CAT trajectory: Wheel loads

Figure 6.15 plots the dynamic wheel loads \mathbf{G}_{dyn} computed by equation (1.57), p. 22, together with the static load defined in equation (1.35), p. 18. Except for the deceleration phases, wheel 3 always has a positive load shift $G_{3,\text{shift}}$. This was already observed with the MCL trajectory. At the beginning, the load shifts to wheel 3 because it is the rear wheel in driving direction while the robot accelerates, and in curved sections it always faces towards the outside of the curves, both causing an increased dynamical load. Wheels 1 and 2 have both positive and negative load shifts, depending on which is the inner and outer wheel in a curve. The smallest dynamic wheel load is $\widehat{\mathbf{G}}_{\text{dyn}} = 37.6 \text{ N}$. Therefore, the wheel loads do not limit the maximum velocity yet.

The friction coefficients are plotted for each wheel in figure 6.16. The maximum absolute value $|\widehat{\mu}| = 0.92$ occurs for wheel 3 at the end of the first curve. The value is not reached at the end of the second curve because the velocity started ramping down earlier.

Figure 6.17 shows the mechanical power \mathbf{P} of each wheel according to equation (1.32), p. 18, as well as the total power according to equation (1.30), p. 18. The power at wheel 3 is negative most of the time, that means motor 3 is generating electric energy. Only when the robot enters a circular section, the power at wheel 3 becomes positive because it pushes the robot around. The maximum negative power occurs also at wheel 3, it is -125 W . This is about 30% of the power the motors were dimensioned for according to table 2.4. The maximum positive power is consumed by wheel 2, it is 59 W

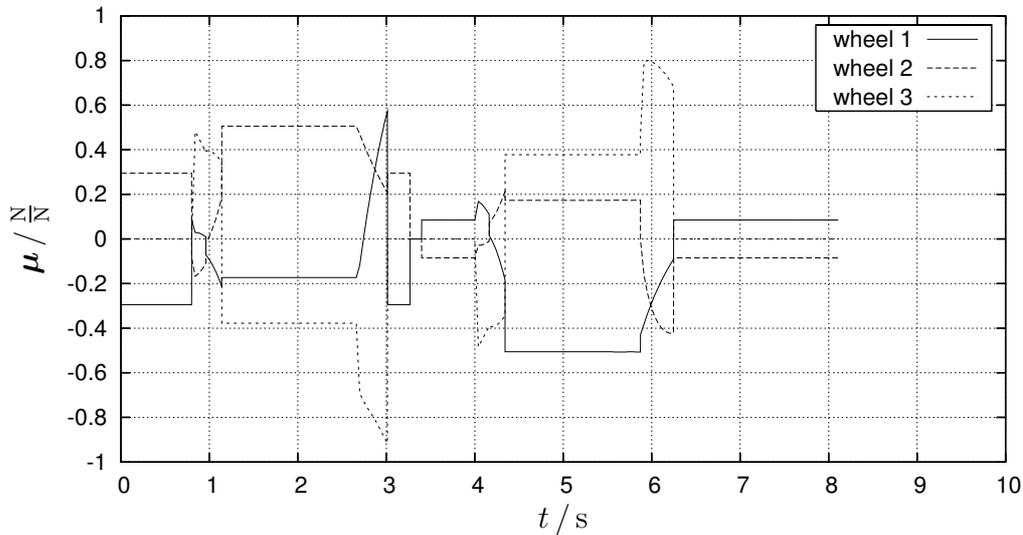


Figure 6.16: CAT trajectory: Friction coefficients

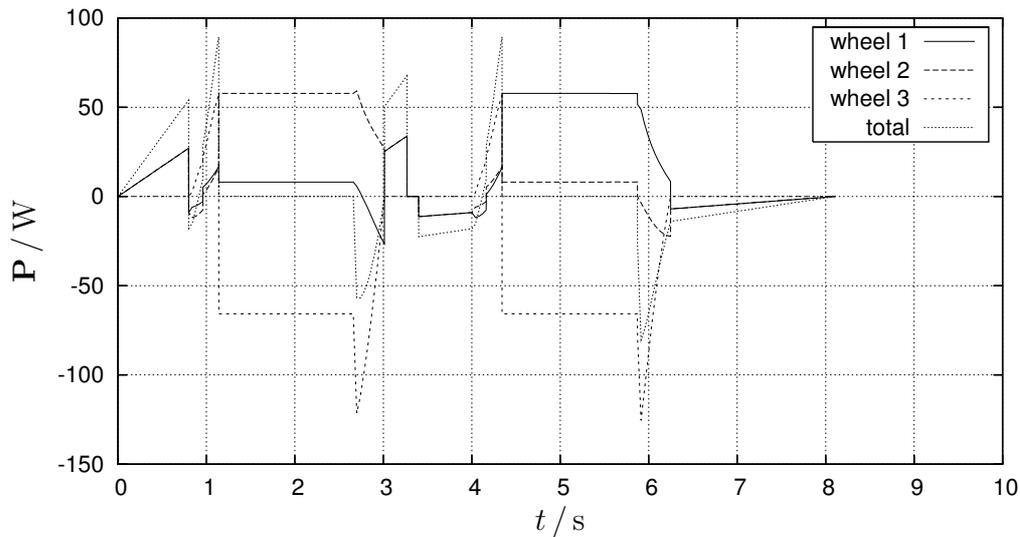


Figure 6.17: CAT trajectory: Wheel powers

or 14%. The total power ranges between the values -80 W and 89 W. Note that theoretically, no battery supply power is required while the robot travels around the circular sections, the power is transferred from motor 3 to motors 1 and 2. Of course the component efficiencies do not allow this perfect behavior, but efficiencies are not included in the graphs in Figure 6.17.

6.2 Sensors

This section describes the calibration process and the resulting precision of the robot self and ball localization based on the video image recognition derived in chapter 5. The video cameras are used as the only object localization sensor while the robot is driving, and all subsequent driving experiments will compare the calculated trajectory positions with the actual robot and ball positions obtained with the video cameras. No special attention was paid to the illumination. There were windows in the ceiling of the laboratory that could not be shaded and which allowed sunlight to enter during day time. In addition to that, the room was illuminated with fluorescent tubes which created a relatively balanced and homogeneous light on the floor. It is worth mentioning that the video image hardware and software was never significantly disturbed by neither the changing lighting conditions at different times of the day nor by the weather outside. In some experiments however some unexplainable reproducible deviations occurred that may have been caused by changing lighting conditions.

6.2.1 Calibration

As already explained in section 5.1.4, a large lookup table stores the ξ - and η -coordinates of every pixel in every camera image. It is the task of the calibration procedure to create reasonable coordinate entries for this table. A special software was developed for this purpose, it is the camera calibration program described in appendix A.2. Figures A.1, A.2, and A.3 (p. 173, 174, and 176), show screenshots of the graphical user interface of the program. The program allows to calibrate every camera individually, and to align the images of each camera with respect to each other.

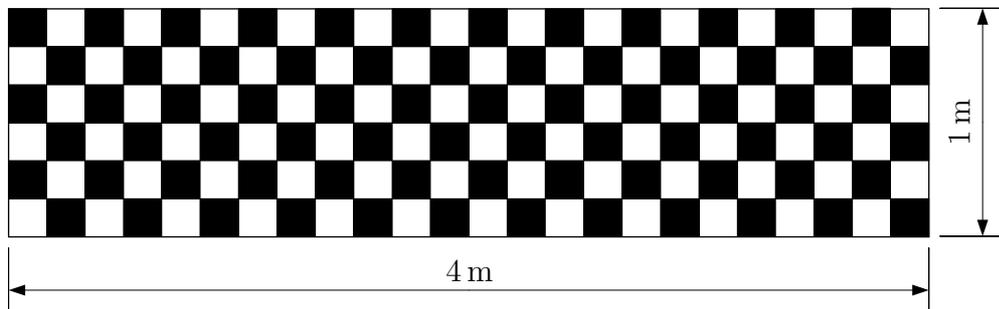


Figure 6.18: Sensor testing: Video camera calibration pattern

In order to perform an accurate calibration of the camera sensors, a reference calibration pattern is required. Figure 6.18 illustrates the calibration pattern that was used. It is a rectangular linoleum carpet that is covered with black and white squares, each square has a length of 0.167 m. Its short and long sides have lengths of 1 m and 4 m, respectively. The carpet does not cover the complete robot view. Therefore, it is not possible to perform the calibration in one step. Typically, the robot was placed in the middle of a short side of the carpet such that the long side was oriented parallel to the ξ -direction. Now, the complete carpet was in the vision of camera 1 and all grid nodes of camera 1 that were located on the video image of the calibration carpet were calibrated. The robot was then turned in increments of about 10° and the new grid nodes located on the pattern were calibrated such that they matched the already calibrated nodes. After the robot had been turned by $\pm 60^\circ$, the complete range up to a horizon of 4 m was calibrated for camera 1. Then the complete procedure was repeated for cameras 2 and 3. The final calibration step was to align the images of the three cameras with respect to each other. This was achieved by placing the robot in the center of the calibration carpet with the long carpet side oriented parallel to the ξ -direction again. Then camera 1 was assigned the angle 0° in robot coordinates because it looks in positive ξ -direction. Cameras 2 and 3 were assigned angles 120° and 240° , respectively because they look to the rear left and rear right. If any camera was not mounted into the robot with perfect alignment, then the camera image of this camera would not align with the other cameras' images of the calibration pattern and its angle could be corrected by the calibration software. Of course it would be more accurate and faster to utilize a large calibration pattern that renders it unnecessary to perform the calibration in steps of 10° but due to limited space in the laboratory this more tedious variant had to be employed.

The calibration error grows larger with increasing distance from the robot because less information is available from the cameras at large distances. Two more factors contribute to the error, the wide angle objectives on the one hand and the reduced camera resolution on the other hand. The image recognition algorithm described in chapter 5 utilizes the outlines of the color coded representation of the video image. These outlines are determined with gradients between pixel rows and columns. The spacing between two pixel rows becomes very coarse at large distances and the error in the predictions increases. Therefore, the calibration horizon is introduced, which is the radius of a circle around the robot inside which a pixel must be located to be considered by the video image recognition. Pixels with coordinates (ξ, η) outside the calibration horizon are ignored because their position error is too large. The calibration horizon was set to $d_{\text{cal}}=3.6$ m. The pixels in the

two topmost pixel rows that are just inside the calibration horizon represent a distance of approximately 0.2 m in reality. As a consequence of d_{cal} being 3.6 m, the robot, located on a short side, cannot see the far end of the calibration pattern.

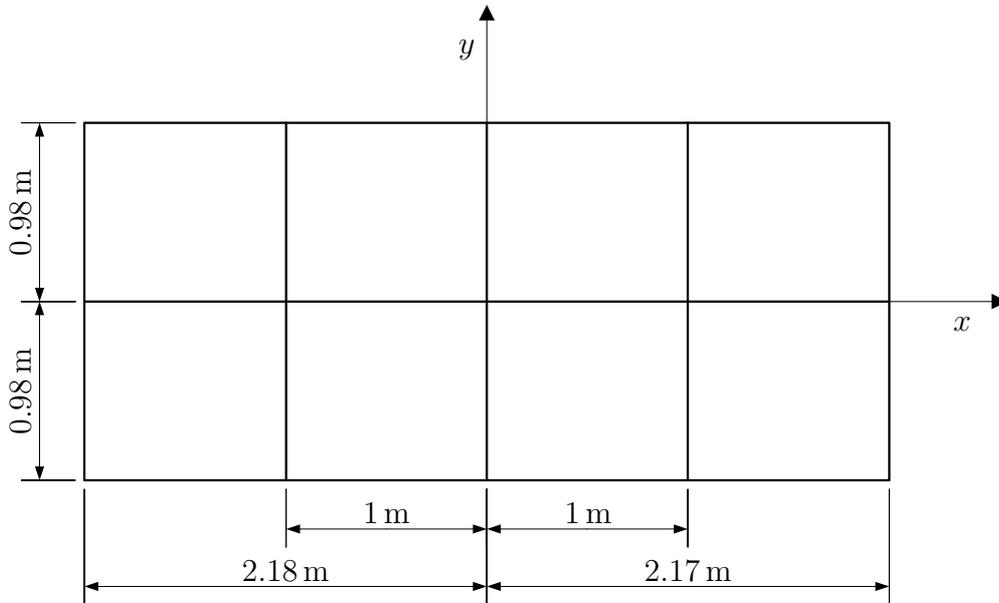


Figure 6.19: Field of play for driving tests

Figure 6.19 illustrates the field of play which was used for all robot driving tests after the cameras had been calibrated. This field was also used for the calibration validation experiments in sections 6.2.2 and 6.2.3. The field consists of a green carpet with lines of white self-adhesive marker tape attached to it. Each line has a width of 4 cm, the dimensions in figure 6.19 refer to the centerline of each marker tape because the robot localization algorithm will compute these coordinates. This is because both sides of the tape produce a color boundary from white to green, the algorithm will not see the face of a tape but its two edges. Then the least squares algorithm computes the coordinates of the best fit line through all the edge pixels which is located in the middle of the tape.

6.2.2 Robot Self Localization

For the experimental validation of the robot's localization of its own position on the field of play, the robot was put in different positions on the field. The robot was always facing in positive x -direction, its orientation was kept

constant $\alpha=0^\circ$. The robot was placed in different spots inside one of the eight rectangular areas outlined by the marker tapes. It was not necessary to place it all over the field because of the symmetry of the field and because the computed position is independent of the distance from the origin. For this test, one of the eight areas surrounded with marker tape is as good as any other. The robot's actual reference position was determined with a measurement tape relative to the marking lines. The result of this experiment is plotted in figure 6.20 and the numeric values are listed in table 6.4.

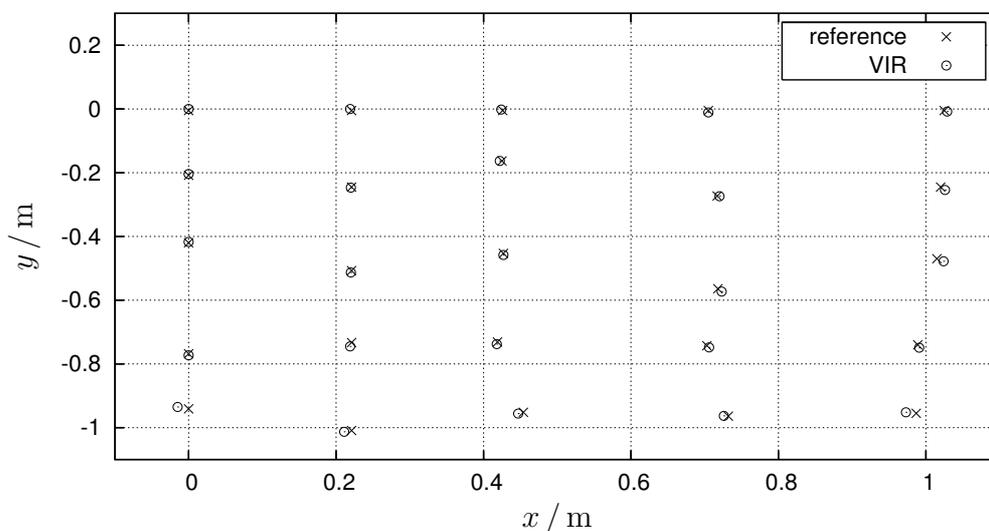


Figure 6.20: Sensor testing: Robot position

Figure 6.20 plots the robot's reference position obtained with the measuring tape as crosses, and the position computed by the VIR algorithm as circles. In total, 25 different robot locations were examined and the figure suggests very good agreement between actual and computed position.

Table 6.4 allows a more detailed analysis. For each location; the robot coordinates of the reference measurement and from the VIR algorithm are listed. The error is provided as deviation in each coordinate Δx and Δy , and as total deviation $\Delta = \sqrt{\Delta x^2 + \Delta y^2}$. The error becomes larger when the robot is closer to the line $y = -1$ m. In these cases, the robot computed a position that is too far in negative direction. This is probably caused by an unprecise position of the field marking lines. These lines are white marker tape glued on the carpet. They shifted their position a little bit after the robot performed a large number of drivecycles in previous experiments.

The maximum error is $\hat{\Delta} = 16.2$ mm and the average value of the 25 measurements is $\Delta_{\text{avg}} = 7.2$ mm. These results are considered very satisfying.

The average error easily meets the target value of 10 mm defined in the introduction. The maximum error is not much larger than the target value. Compared to other teams participating in the RoboCup competition who reported error ranges of more than 500 mm, this result is extremely good.

#	reference position		video image recognition				
	x_{ref} mm	y_{ref} mm	x_{VIR} mm	y_{VIR} mm	Δx mm	Δy mm	Δ mm
1	0	- 5	0	0	0	- 5	5.0
2	0	- 208	0	- 204	0	- 4	4.0
3	0	- 420	0	- 417	0	- 3	3.0
4	0	- 768	0	- 773	0	5	5.0
5	0	- 941	- 15	- 935	15	- 6	16.2
6	221	- 5	219	0	2	- 5	5.4
7	221	- 245	220	- 247	1	2	2.2
8	221	- 507	220	- 513	1	6	6.1
9	221	- 733	219	- 745	2	12	12.2
10	221	-1009	211	-1013	10	4	10.8
11	426	- 5	424	- 2	2	- 3	3.6
12	425	- 163	422	- 163	3	0	3.0
13	427	- 453	427	- 458	0	5	5.0
14	419	- 731	418	- 738	1	7	7.1
15	454	- 952	447	- 956	7	4	8.1
16	705	- 5	705	- 11	0	6	6.0
17	717	- 273	720	- 274	- 3	1	3.2
18	718	- 564	723	- 573	- 5	9	10.3
19	703	- 743	706	- 748	- 3	5	5.8
20	732	- 964	726	- 963	6	- 1	6.1
21	1025	- 5	1029	- 8	- 4	3	5.0
22	1020	- 245	1026	- 254	- 6	9	10.8
23	1015	- 470	1024	- 478	- 9	8	12.0
24	989	- 740	991	- 749	- 2	9	9.2
25	987	- 955	973	- 952	14	- 3	14.3

Table 6.4: Sensor testing: Robot position

The following numbers give an impression on the statistics involved in the VIR algorithm. They were taken from a ride along a trajectory created with the MCL method from section 6.6 when the ball was dribbled. The

drivecycle lasted about 18 s during which the robot position was determined over 450 times with the VIR. This is considered more representative than the 25 test locations described in this section. Typically, an average of 250 pixels were used from each camera summing up to a total of 750 pixels to compute the robot position in each time step. This is a rather large number of pixels, so a good statistical result can be expected. The minimum and maximum number of pixels that were contributed by one single camera in one time step during this test were 61 and 426, respectively. The minimum and maximum total number of pixels from all cameras were 329 and 1232, respectively.

6.2.3 Ball Localization

The experimental validation of the ball localization with the video cameras according to section 5.3 was carried out similarly to the robot self localization. This time, the robot was placed at the center of the field of play, its orientation was again kept constant $\alpha=0^\circ$. The ball in this experiment as well as in all other experiments was a red FIFA size 5 soccer ball according to the regulations of the RoboCup competition. This ball was placed in different locations all over the field. Its actual reference position was determined with a measurement tape relative to the marking lines. This position was compared with the ball position computed by the VIR.

Figure 6.21 shows the results of this experiment. In total, 29 different ball positions were investigated. The ball's reference position obtained with the measuring tape is plotted with crosses, and the position computed by the VIR algorithm with circles. It seems in the figure that the absolute error of the VIR increases with the distance between ball and robot. There are two reasons for this: First, at a greater distance the ball becomes smaller in the video images and therefore, less pixels are available to compute the center of the ball image as described in section 5.3.1. Second, the camera calibration is less accurate for larger distances for the same reason and therefore, the computation of the ball position as described in section 5.3.2 is also less accurate. It should therefore be expected that the error of the ball position grows at least quadratically with the distance between ball and robot. Note that most of the ball locations in figure 6.21 were seen by only one camera. The three lines extending from the origin give a rough impression of the vision limits of each camera, the circle indicates the position and approximate size of the robot. There is a small overlap of the field of view of two cameras, so some ball locations that are close to one of the three lines were seen with two cameras. In that case, the ball positions from the two cameras were averaged. The ball spots on the right side of figure 6.21 were seen with camera 1, the spots on the left top and left bottom were seen with cameras

2 and 3, respectively. Another observation is that the VIR ball positions seem to be shifted towards the negative x -direction for distances above 1 m. There is no explanation for this phenomenon. Maybe the light was falling more intensely from one side creating shadows or reflections on the ball that influenced the image recognition in an unbalanced way.

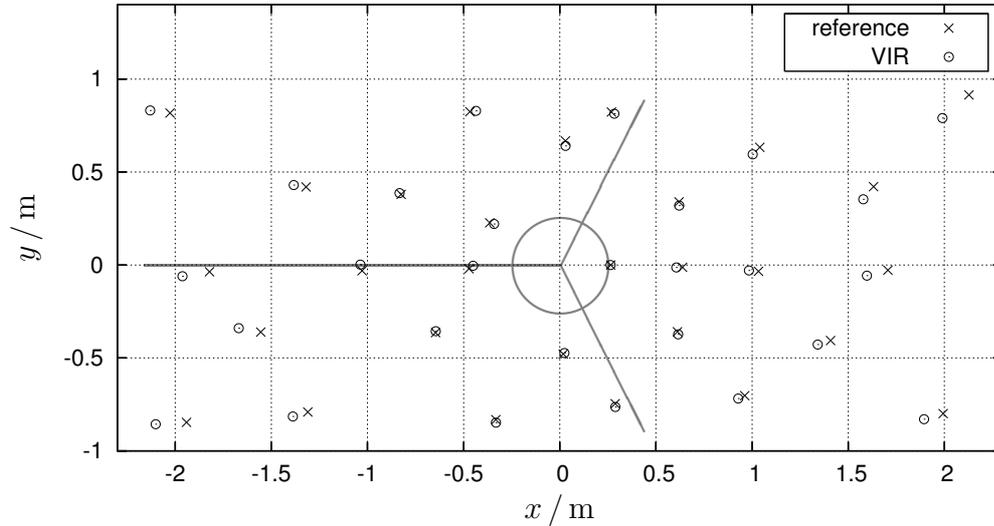


Figure 6.21: Sensor testing: Ball position

Table 6.5 lists the numeric results of each measurement, sorted by distance between ball and robot. In this experiment, the distance between ball and robot is relevant for the error, it is computed with the reference tape measurement, $d_{\text{ref}} = \sqrt{x_{\text{ref}}^2 + y_{\text{ref}}^2}$. The error is provided as deviation in each coordinate Δx and Δy and as total deviation $\Delta = \sqrt{\Delta x^2 + \Delta y^2}$. In general, it can be stated that the absolute error Δ increases with the distance between ball and robot. This is already a result from the discussion of figure 6.21 and the reasons were explained. The smallest position deviation was less than 1 mm, it was measured at the ball neutral position for dribbling. The largest deviation was 185.5 mm and occurred at the most remote spot at a ball distance of 2316 mm. It is reasonable to compute the relative error Δ/d_{ref} because of the two influences of distance on the error explained above. This value can be found in the last column of table 6.5. The smallest relative error of 0.2% occurs at the closest measurement, and the largest error of 8.0% at the most remote measurement. The average value of the relative error is 4.2% which is considered very good. It meets the target value which was defined in the introduction to be 5% of the actual distance. It should be kept in mind that this precision can easily be increased with a higher resolution of the cameras' digitized video images.

#	reference			video image recognition					
	position		dist.	position		error			
	x_{ref}	y_{ref}	d_{ref}	x_{VIR}	y_{VIR}	Δx	Δy	Δ	Δ/d_{ref}
	mm	mm	mm	mm	mm	mm	mm	mm	%
1	265	0	265	264.5	0	0.5	0	0.5	0.2
2	-364	227	429	-342	221	-22	6	22.8	5.3
3	-472	-20	472	-451	-4	-21	-16	25.7	5.4
4	18	-479	479	24	-473	-7	-6	8.5	1.8
5	638	-12	638	606	-13	32	2	31.5	4.9
6	29	668	668	30	641	-2	27	26.5	4.0
7	620	340	707	621	320	-1	20	20.0	2.8
8	612	-359	710	616	-373	-4	14	14.6	2.1
9	-646	-364	741	-645	-356	-1	-8	8.1	1.1
10	288	-746	800	289	-764	-1	18	18.0	2.3
11	269	823	866	284	814	-15	9	17.5	2.0
12	-333	-832	896	-332	-848	-1	16	16.0	1.8
13	-825	380	908	-833	387	8	-7	10.6	1.2
14	-467	825	948	-436	829	-31	-4	31.3	3.3
15	-1028	-31	1028	-1037	2	10	-33	33.9	3.3
16	1033	-34	1033	984	-29	49	-5	48.7	4.7
17	962	-703	1191	928	-718	34	15	37.2	3.1
18	1040	633	1217	1003	595	37	38	53.0	4.4
19	-1319	420	1384	-1383	430	64	-10	64.8	4.7
20	1408	-406	1465	1341	-428	67	22	70.5	4.8
21	-1310	-791	1530	-1388	-815	78	24	81.6	5.3
22	-1555	-361	1596	-1669	-340	114	-21	115.9	7.3
23	1632	422	1686	1578	354	54	68	86.8	5.2
24	1706	-28	1706	1597	-57	109	30	112.4	6.6
25	-1821	-36	1821	-1962	-60	142	25	143.6	7.9
26	-1941	-846	2117	-2101	-856	160	10	160.3	7.6
27	1994	-799	2148	1894	-829	100	30	104.4	4.9
28	-2027	818	2186	-2130	832	103	-14	103.9	4.8
29	2128	915	2316	1990	791	138	124	185.5	8.0

Table 6.5: Sensor testing: Ball position

Table 6.5 suggests that the relative error Δ/d_{ref} increases with the ball's distance d_{ref} from the robot. Figure 6.22 analyzes the dependence between distance and relative error. It plots the term Δ/d_{ref}^2 versus distance. Except for the first few points close to the robot, the data points distribute

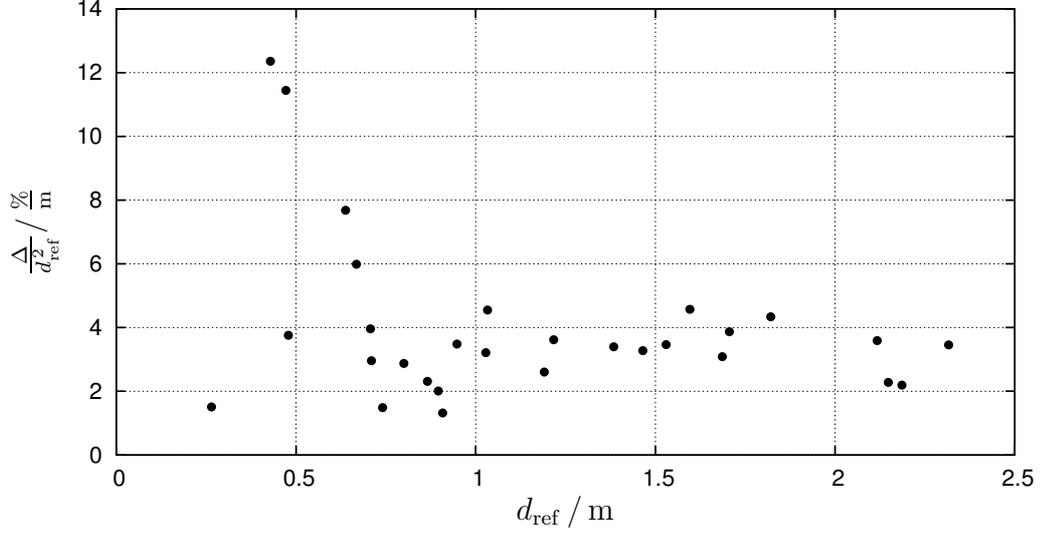


Figure 6.22: Sensor testing: Ball position error

around a constant value of about 3.4 %/m, independent of the distance. This supports the assumption that the position error Δ depends quadratically on the distance d_{ref} between ball and robot. The ball perspective parameter χ_B was determined at the beginning of this experiment by placing the ball in its neutral position. In this position, the perspective parameter was modified until the video image recognition produced the correct location. This is the reason why the error is minimal in the neutral position. The value of the perspective parameter is $\chi_B=0.87$. This means that the ball will be out of sight for the robot if it is located at a distance greater than $\chi_B \cdot d_{\text{cal}}=3.13$ m from the robot center. This is because the ball localization first determines the coordinates of point P depicted in figure 5.9, p. 95, and then scales these according to equations (5.36) and (5.37), p. 96, to obtain the ball position.

Some statistical values can also be provided for the ball. When the ball is located in its neutral dribbling position which is as close as it can get to the robot, it can only be seen by camera 1. In one of the dribbling experiments presented in section 6.6, the ball was located more or less permanently in this neutral position. During the complete drivecycle, the number of ball outline pixels used for the computation of the ball position reached maximum and minimum values of 62 and 42 while the average was 52. On the other hand, when the ball was over 2m away from the robot as in table 6.5, its video image grew so small that the outline sometimes consisted of only 10 pixels. Note that the minimum number of pixels that is required to compute the ball position was limited to 10 in the ball detection software.

6.3 Robot System

This section describes the robot setup and the physical parameters of the actual implementation. The robot with all its components remained unchanged throughout all experiments in this chapter except for the wheel type variation described below. Section 1.3 shows 3-dimensional drawings and a photograph of the robot hardware assembly and describes the major components. Refer to appendix A.3 for a description of the robot control program that is responsible for the video image recognition, control of the propulsion motors, and radio communication with the user terminal.

Table 6.6 lists all the relevant physical parameters of the robot which are required to compute the motor speeds and torques. These parameters were determined with the double roller type wheels because these wheels showed the best driving performance which is described below.

Parameter	Symbol	Value	Source
wheel radius	r_W	0.110 m	experiment
distance robot center to wheel	r_R	0.240 m	experiment
robot mass	m_R	27.180 kg	measurement
ball neutral position	ξ_0	0.265 m	measurement
ball perspective parameter	χ_B	0.87	measurement
robot height of center of mass	h_M	0.1609 m	UG
robot moment of inertia, α	$\Theta_{\alpha,R}$	0.9458 kg m ²	UG
wheels moment of inertia, ax	$\Theta_{ax,W}$	0.0234 kg m ²	UG

Table 6.6: Robot system: Physical parameters (double roller wheels)

The wheel radius r_W was determined by carefully pushing the robot manually in ξ -direction by a specified distance. The robot's odometer based on the relative rotor positions measured with the digital servoamplifiers (DES) and the digital encoders was compared with the actual distance. Special attention was paid so that the robot orientation was always constant and that no slip occurred at the contact area between the wheels and the floor. The experiment was carried out a number of times for reproducibility. The value for r_W was adjusted until the odometer reported the correct distance.

The distance r_R from the robot center to the contact point of the wheel with the floor was determined in a similar way. This time, the robot was turned about its vertical axis for a certain number of full rotations with its translational coordinates fixed. With the known wheel radius r_W , the robot's odometer was again compared with the actual number of rotations, and the value of r_R was modified until a satisfactory result was achieved.

This experiment was repeated a number of times to ensure reproducibility. Note that this distance has to be seen as an effective radius because it was determined for the double roller wheel type, and the two rows of rollers have different distances to the robot's central α -axis.

The robot mass was simply measured by putting the complete robot onto scales, including battery and wheels. In addition to that, the weight of every single component was determined separately.

The ball neutral position ξ_0 was measured with a tape for a FIFA size 5 soccer ball which is played in the RoboCup competition.

In the same way, the ball perspective parameter χ_B used in equations (5.36) and (5.37), p. 96, was determined. The ball was placed in the defined neutral position and the perspective parameter was modified until the video image recognition produced the correct location.

The next three parameters, the height of the center of mass as well as the two moments of inertia, are very difficult to determine experimentally. Therefore, they were obtained in a different way. The complete robot was designed on the 3-D computer aided design software Unigraphics (UG). This application allows to produce a so called weight report including masses and moments of inertia of a complex assembly. First, the geometry of all components was modelled accurately. Then, the density for each component was selected homogeneously and set to the value which produced the correct component mass. Note that according to UG, the robot's center of mass is not exactly located on its geometrical central α -axis, but this was ignored in all calculations.

Wheel Variation

This section presents the experiments that were conducted to determine the best wheel type for the omnidirectional propulsion system. Three different wheel types were tested. They are depicted as 3-dimensional drawings and as photographs of the actual hardware in section 1.1. The test procedure is simple. The robot was sent along a predefined trajectory with its motion controller gains k_α and k_{xy} set to zero, and the course deviation was observed. An *eight* course was created with the CAT method according to section 3.6. A similar parameter set as in section 6.1.2 was chosen, only that two full rounds were driven here and the parameters $\mathbf{\Lambda} = [2, 2, 2, 2]$ and $\mathbf{\Gamma}_{\text{CAT}} = [0, 1, 0]$ were used. The video image recognition was used to record the robot's actual path, the extrapolated angle α_{EXT} from the VIR was used to map the drivecycle velocity into robot coordinates according to equation (1.8), p. 16. The wheel variation tests were conducted with an early version of the robot motion controller according to figure 4.2, p. 75. The robot control software was

called five times during one image recognition cycle with an average sampling period of 8 ms. The set speed ramping option of the DES was not utilized at this time. Instead, the parameter \dot{n}_{ramp} was set to a very large value. The motor controller parameters that were used are listed in table 6.8, p. 139.

Single Roller Type

Figure 1.1, p. 10, illustrates the single roller wheel type which was originally intended to be used for the robot. Eight rollers are distributed around the wheel's circumference. Their shape was chosen such that their envelope forms a circle. Each gap between two rollers creates a secant to that circle, the maximum deviation between that secant and the circle is smaller than 0.5 mm. It was hoped that the vibrations caused by these gaps did not influence the robot's driving performance significantly.

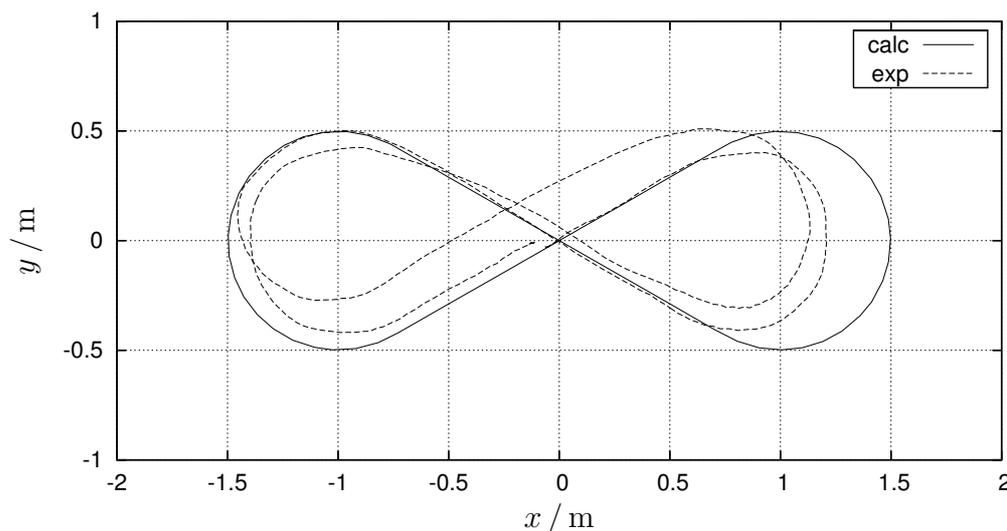


Figure 6.23: Robot system: Single roller wheel

Figure 6.23 shows the calculated drivecycle and the actually driven trajectory in the experiment. It is obvious that the result is not very satisfying. The robot cuts a large bit of the first loop. In the second loop however, at $y=0.5\text{m}$ it comes rather close to the calculated trajectory. Then, it deviates a fair bit in negative x -direction and completes the first round at $x=-0.48\text{m}$. The second round looks similar and for some unknown reason, the robot manages to complete the drivecycle only 7 cm away from its initial position. While the robot was driving, there were visible and audible vibrations shaking the whole machine. It was quite obvious that these vibrations

were induced by the gaps between the wheel rollers, and that they were responsible for the lousy driving performance. The cameras at the top end of the mast were shaking so violently that it is remarkable that the video image recognition was not impaired by blurry video images.

Double Roller Type

This situation could be improved by adding a second row of eight rollers to each wheel. Each roller of the second row covers a gap between two rollers of the first row which is illustrated in figure 1.2, p. 10. The expected disadvantage of this design was that the robot radius r_R used for the initialization of the system matrix \mathcal{Y}_0 defined in equation (1.14), p. 17, is no longer uniquely defined by the geometry. It changes by the distance of the two rows of rollers depending on which roller is in contact with the floor. Therefore, the effective radius r_R was determined by an experiment as described with table 6.6.

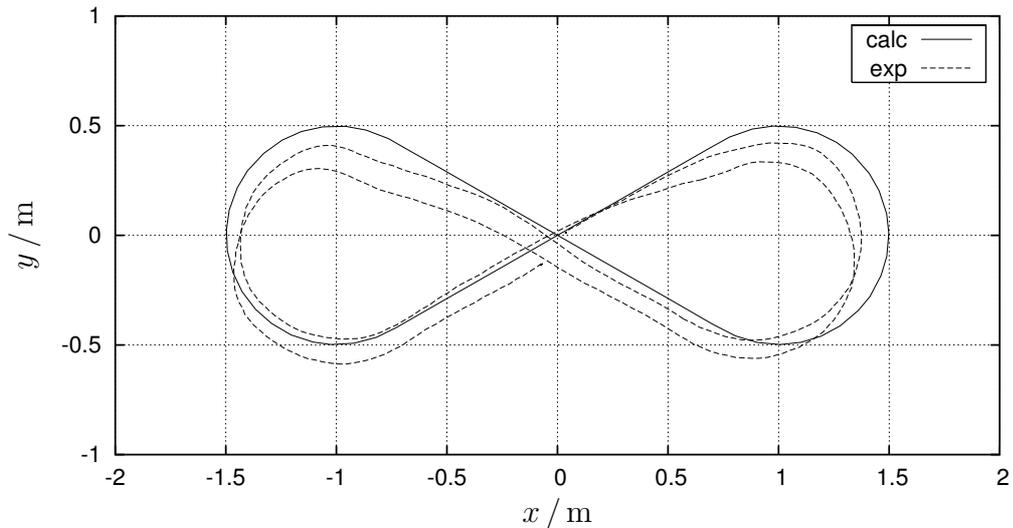


Figure 6.24: Robot system: Double roller wheel

The resulting driving performance can be seen in figure 6.24. The robot follows the calculated trajectory better now. However, it still cuts all curves and never reaches the maximum values in positive and negative x -direction. The driven path is more symmetric than the one produced with the single roller type wheels, except for the drift in negative y -direction. It could not be observed or concluded that the lack of a geometrically defined robot radius r_R imposed any disadvantages on the overall performance.

The driving performance could be further optimized by modifying the robot motion controller. For all subsequent driving experiments after the wheel type variation, the robot motion controller was only called one time during each VIR cycle instead of five times. The motor set speed ramping ability was activated, the DES parameter \dot{n}_{ramp} was computed and communicated to the DES with every new motor speed n_{ramp} . The resulting improvement can be seen in figures 6.33 and 6.34, p. 144, showing an open loop experiment that uses the same trajectory shape and the same velocity parameter Λ , but the robot orientation is constantly $\alpha_{\text{cyc}} = 90^\circ$. Figure 6.37, p. 149, also used the same trajectory shape and velocity, but a different damping ratio in the parameter Γ_{CAT} . Refer to section 4.1 for an explanation of the improved performance with the activated set speed ramping ability.

Cutting Edge Type

The third wheel type illustrated in figure 1.3, p. 11, was developed to reduce complexity, weight, and cost of the omnidirectional wheels. It was designed without rollers, instead the wheel has a large number of sharp edges around its circumference. These edges are supposed to cut into the necessary soft floor in order to create a preferred direction of passive slip along the edge. In addition to that, the edges are slanted at an angle $\gamma = 30^\circ$ to reduce wheel induced vibrations. It was the goal to allow the next cutting edge to contact the floor before the previous one has lost its contact to it. This is similar to the helical tooth system for gear wheels. Two disadvantages were expected: A significantly increased friction between wheel and floor, and the undefined geometry of the robot radius r_{R} . The effective radius r_{R} was determined with the same experiment as used for the double roller type wheels.

Figure 6.25 shows the result of the test. The robot came off its calculated course dramatically in negative y -direction during the first loop. The second loop was taken much better, almost no deviation was added to the y -coordinate. The third loop was the right bend again, and in its second half the robot had to be stopped before leaving the narrow field. The reason why the robot was better in driving left curves can be explained with the geometry of the wheels. All three wheels are identical, this creates preferred directions with respect to the friction between wheels and floor. In a left curve, the left front wheel is inside the curve, and the cutting edges point approximately in tangential direction of the circle. That means this wheel does not need to rotate much. The right wheel is on the outside of the circle, its cutting edges are pointing approximately in radial direction allowing maximum grip with a small velocity along the edges. That means, the wheel that is propelling the robot against the friction is the outside wheel, pushing

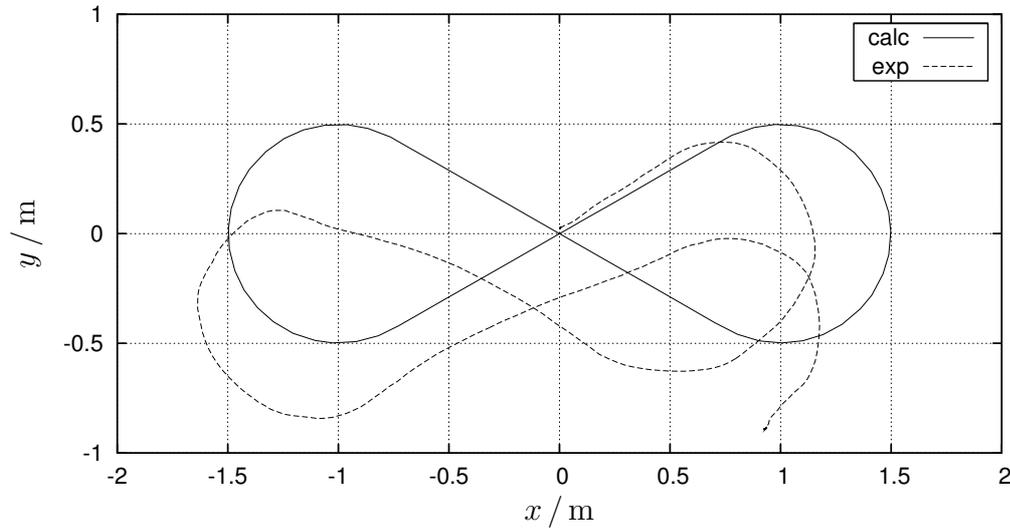


Figure 6.25: Robot system: Cutting edge wheel

the robot towards the inside of the circle. In a right curve, it is the other way round. The propelling wheel is the inner right front wheel, and it pushes the robot towards the outside of the circle while the left front wheel is outside and is rather passive.

It was decided that this wheel type is not an option because of the great influence of friction on the driving behavior.

Power Consumption

In a second test, the power consumption of the three wheel types was tested. This time, the same trajectory was driven with reduced velocity $\mathbf{\Lambda} = [1, 1, 1, 1]$ and the same parameter $\mathbf{\Gamma}_{\text{CAT}} = [0, 1, 0]$. The robot motion controller was active with the proportional gains $k_{xy} = k_{\alpha} = 1 \text{ s}^{-1}$. The test procedure foresaw that the battery was fully charged at the beginning. Then, the robot repeated the *eight* slalom until the battery management system stopped at a defined battery voltage to protect the robot. This test was conducted one time with each wheel type, the resulting times are listed in table 6.7:

single roller	3150 s = 52'30"
double roller	3390 s = 56'30"
cutting edge	1170 s = 19'30"

Table 6.7: Robot system: Power consumption

The driving duration with the double roller wheel type is almost three times as long as with the cutting edge wheel type. This is an expected result due to the immensely increased friction caused by the edges sliding across the carpet. It is interesting that the driving durations between the single and the double roller wheel types differ by about 7%. Perhaps, the additional electric power consumed by the single roller wheels is used to excite all the vibrations in the robot. Note that the durations in table 6.7 include the electric power consumed by the electronic components as PC, cameras, and the DES. If these were eliminated and only the power consumed by the propulsion motors was compared for the three wheel types, their difference would be even more pronounced. All driving durations are long enough to last for one half time in a RoboCup competition which currently is 10 min playing time with no timeouts.

6.4 Actuators

This section investigates the behavior of the three electric motors that make up the robot's propulsion system discussed and dimensioned in section 2.2. The goal is to validate the mathematical model of the robot and the motor dimensioning by comparing the calculated wheel velocities and torques with the measured values from the experiment.

The testing trajectory was created with the CAT method using the parameters $\mathbf{\Lambda}=[1.5, 2.5, 1.5, 0.5]$ and $\mathbf{\Gamma}_{\text{CAT}}=[0.8, 5, 0.2]$. The data plotted in the following figures was recorded during a fast dribbling experiment discussed in section 6.6, the associated robot and ball trajectories are plotted in figure 6.47, p. 159. These parameters were also used for the discussion of the CAT trajectory planning method in section 6.1.2, the only difference is that here two full rounds were driven whereas section 6.1.2 only used one. Refer to section 6.6 for more details on the experimental setup.

Figure 6.26 shows the angular speeds $\dot{\varphi}$ of all wheels with respect to time for a complete run, comparing the calculated drivecycle wheel speeds to their actual values. The drivecycle wheel speeds were calculated from the drivecycle velocity $\dot{\mathbf{x}}_{\text{cyc}}$ in x - y coordinates using equations (1.8) and (1.9), p. 16. The actual robot orientation α_{EXT} was used in equation (1.8) to compute the velocity in robot coordinates. This was discussed in section 4.2. The actual wheel speeds were calculated from the motor speeds \mathbf{n}_{DES} using equation (1.7), p. 16, and dividing by the gear ratio i_{gear} . In figure 6.26, the actual wheel speeds show some clearly visible oscillations around their calculated value at some times. This holds specifically for wheels 1 and 2. At other times, there are no oscillations at all and the actual speed follows the

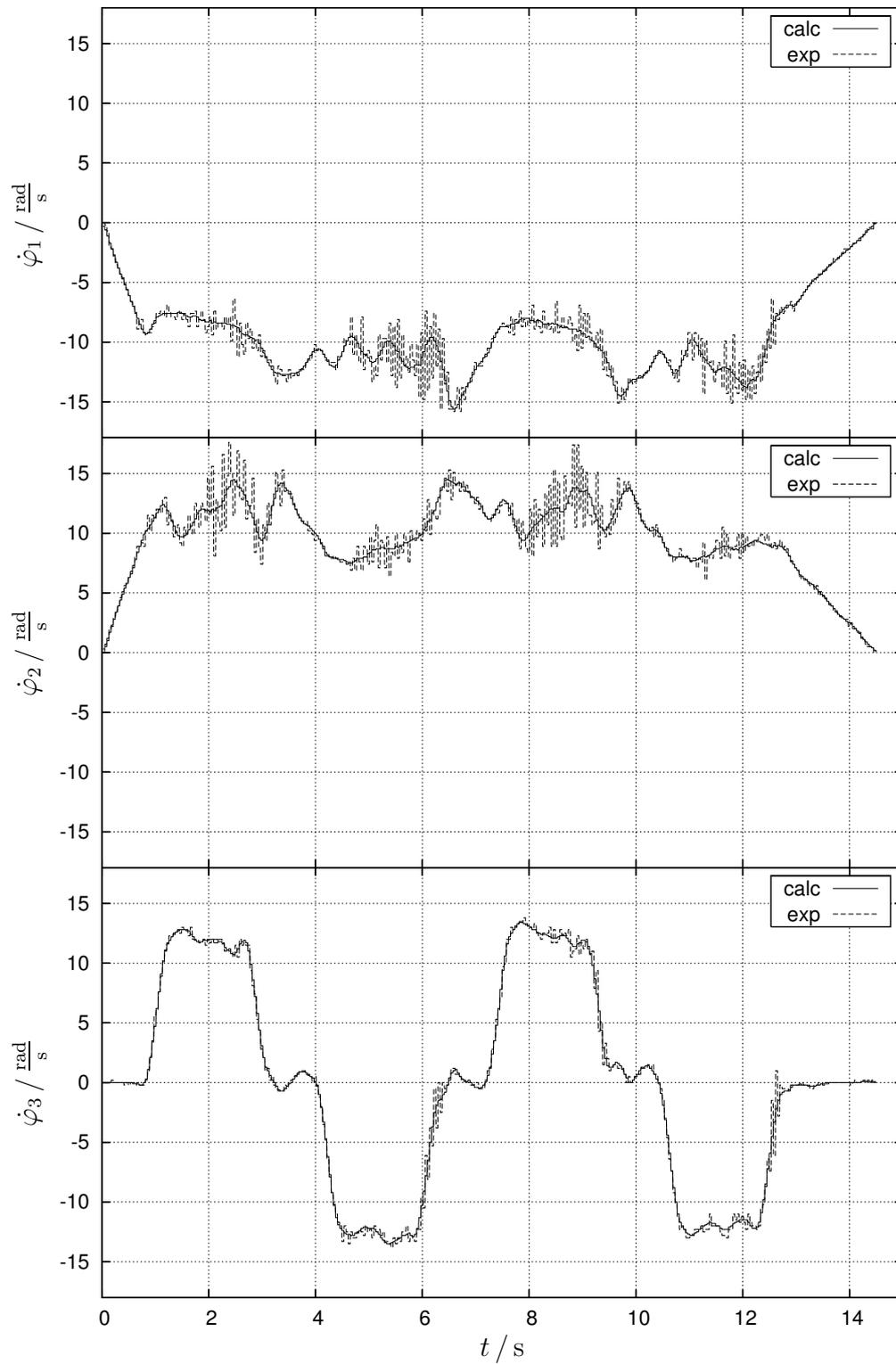


Figure 6.26: Actuator testing: Wheel speeds

calculated speed very precisely. This is the case for wheels 1 and 2 during the ramps at the beginning and in the end, and for wheel 3 more or less permanently. The reasons for the oscillations are probably the high gains for the DES speed controller that were obtained as a result of section 6.5.1. The oscillations do not seem to reduce the robot's ability to follow its predefined course. This is also discussed in section 6.5.1. Comparing the calculated wheel speeds in this experiment with those in figure 6.13, p. 114, it can be seen that they do not match exactly. The speeds in figure 6.13 are often constant over long periods of time or they are ramping at a constant rate. The calculated speeds in figure 6.26 that represent the drivecycle velocity are not as straight and constant, they permanently show some curvature. The reason for this is that different robot orientations were used to calculate the wheel speeds. In figure 6.13, the calculated robot orientation α_{cyc} from the drivecycle was used while in figure 6.26 the actual robot orientation α_{EXT} was used. The calculated robot orientation is plotted in figure 6.10, p. 112. It is constant or changing at a constant rate most of the times. The actual robot orientation is not plotted here but its deviation from the calculated orientation grew as high as $\pm 15^\circ$ during this test.

Figure 6.27 plots the three wheel torques \mathbf{T} with respect to time for the same test run, comparing the calculated drivecycle torques to their actual values. The calculated torque is computed with equation (1.51), p. 22, using the actual robot orientation α_{EXT} and the robot physical parameters listed in table 6.6. Note that the calculated wheel torques were computed with the total mass matrix \mathcal{M} including both gliding and rolling contributions because they were compared with the motor torques, and the motors have to accelerate both the wheels and the complete robot system. The actual wheel torque was computed from the measured effective motor current I_{DES} using equation (2.14), p. 36, with the constant maximum gearbox efficiency mentioned in section 2.2 and in table A.2, p. 171. The wheel torques also show strong oscillations, sometimes their amplitude is larger than the calculated value. Again, wheel 3 shows less oscillations than the other two wheels. The only explanation for the oscillations can be seen in the DES current controller parameters. Refer to section 6.5.1 for the description and the results of the tuning procedure. In addition to the oscillations, the actual torque does not follow the calculated torque as well as the wheel speeds did, not even during the periods with little oscillations. There are many transients when the torque is expected to stay constant for a while. These are created by the DES speed controller discussed in the next section. It must also be considered that the current detection system inside the DES has an unknown error, and the gearbox efficiency in equation (2.14), p. 36, is the maximum efficiency supplied by the manufacturer. The real efficiency is probably smaller and

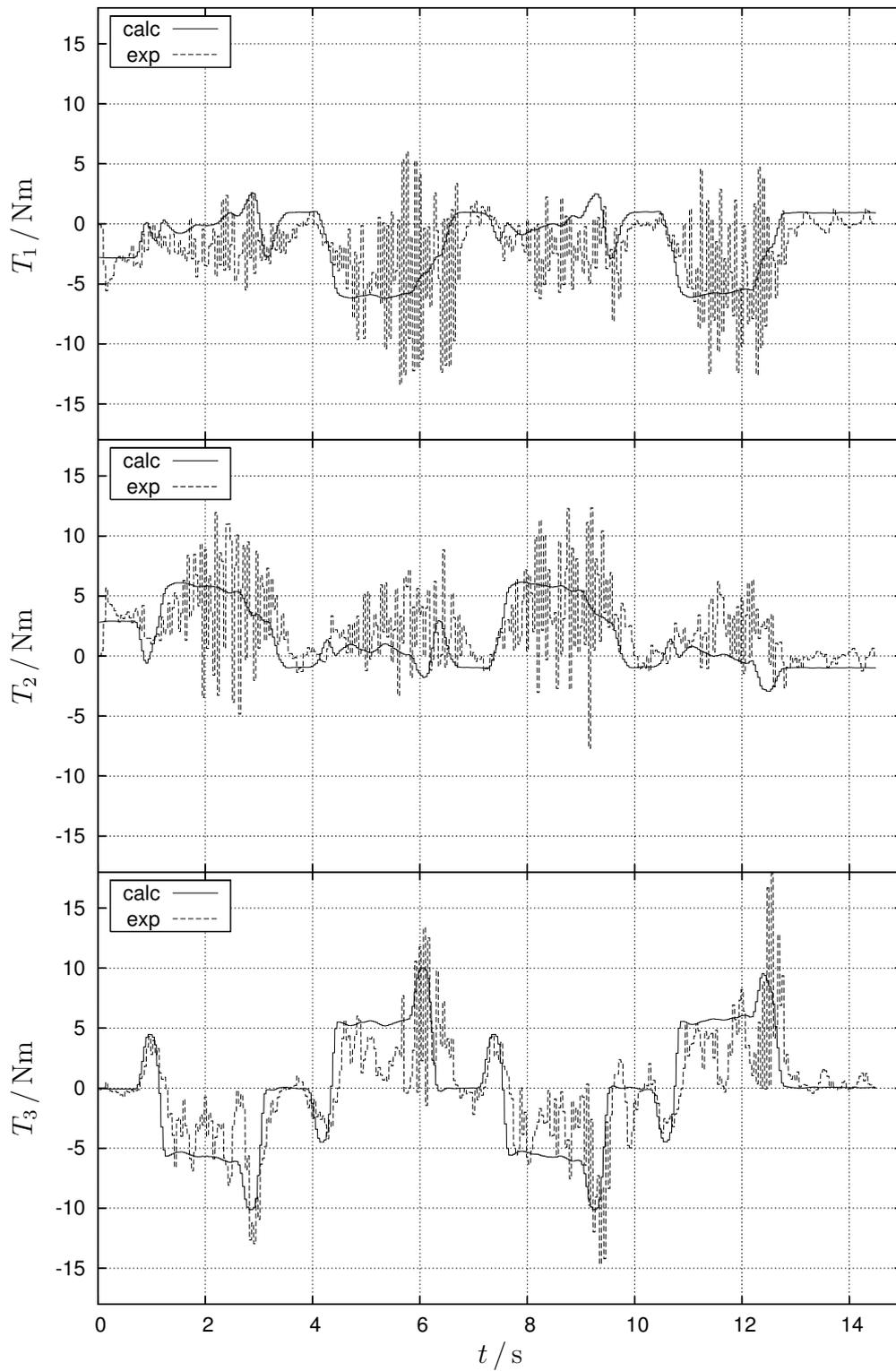


Figure 6.27: Actuator testing: Wheel torques

depends on the actual operating conditions.

Considering these uncertainties in the torque measurement, it can be concluded that figure 6.27 successfully validates the kinetic robot model derived in section 1.2.2 while figure 6.26 verifies the kinematic description in section 1.2.1.

Figure 6.28 allows to evaluate the success of the motor dimensioning described in section 2.2. It plots the calculated and measured wheel speeds $\dot{\varphi}$ versus their respective torques \mathbf{T} for all three wheels. The experimental data was taken from the same fast dribbling experiment as in the previous two figures. Each data point represents a measurement in the robot motion controller subroutine. They are therefore spaced at a constant time increment of 40 ms. The limits derived from the motor characteristic are also added to the figure. The top right quadrant of each plot can be compared to figure 2.2, p. 35. This quadrant represents the propelling case in the wheels' forward direction. The straight line with bullets delimits the maximum possible operating conditions. It is identical with the line for the actual batteries in figure 2.2. The bottom left quadrant of each plot represents the propelling case for the wheels' reverse direction. The remaining two quadrants represent the generating cases with the motor recharging the batteries. In these cases, the maximum wheel torque is increased by the factor η_{gear}^{-2} compared to the propelling cases according to equation (2.14), p. 36. Figure 6.28 shows that wheel 1 sometimes comes close to its limits in the reverse direction. Wheel 2 also comes close to its limits, but in forward direction. Both wheels 1 and 2 operate at low speeds and torques most of the times. This can be estimated by the density of the data point clouds. It could also be concluded from the calculated curves but they are completely hidden by the experimental data points. Wheel 3 shows a different behavior. The data points condense around the calculated conditions which create the shape of two ellipses. The wheel operates at different conditions compared to the other two wheels because it is the rear wheel. It is predominantly responsible for maneuvering the robot around the circular sections. Therefore, the trajectory shape is reflected in the operating conditions of this wheel. Note that some of the data points reach beyond 16 Nm which is possible since the motors are generating in these cases. The cases in which the measured motor loads come close to the limits were caused by the oscillations in the motor currents. These cases should not be used to evaluate a successful motor dimensioning. The calculated operating conditions are at a fair distance from the limits imposed by the propulsion system. This means that the motors are dimensioned properly.

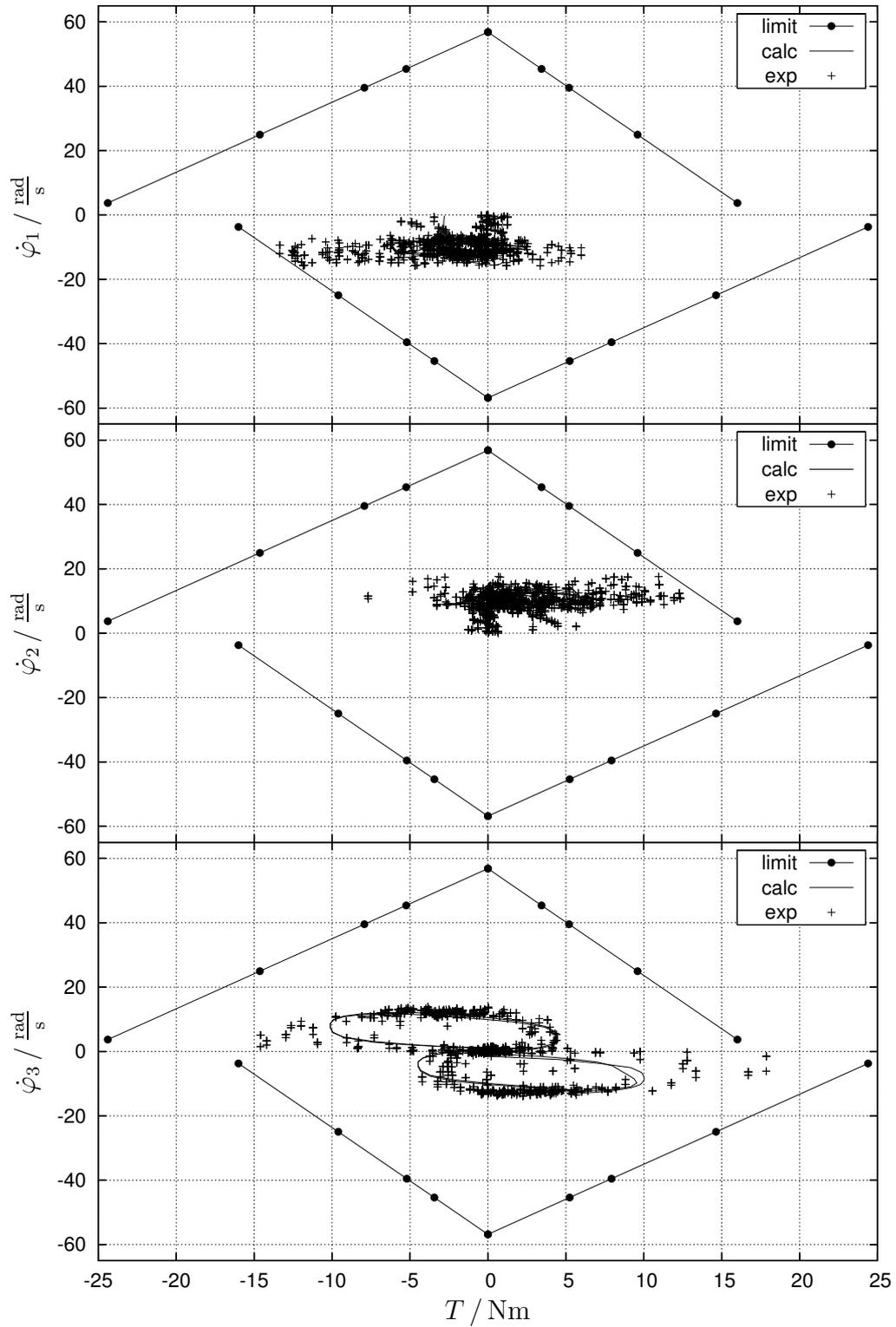


Figure 6.28: Actuator testing: Wheel speed–torque diagram

6.5 Controllers

This section describes the process and the results of tuning the parameters of the DES motor controllers as well as the robot motion controller according to chapter 4.

6.5.1 Motor Controller Tuning

The motor controller contained in each DES consists of a cascade of a speed and a current controller, as described in section 4.1. According to the manufacturer, there is no preferred procedure to obtain an optimal set of control parameters $k_{n,P}$, $k_{n,I}$, $k_{I,P}$, and $k_{I,I}$. The manufacturer does not even provide the physical units of measurement of the parameters because the best method to tune the controllers is by trial and error. This was done in the following experiments. The controllers were optimized with respect to their response to set point variation. This is because the set points for both speed and current controller vary permanently due to the robot's dynamic driving. Their response behavior to disturbances was not investigated. Disturbances can be neglected because the robot moves on a flat floor without bumps or slopes. A standard procedure for determining the controller could not be applied because of the missing physical units. Starting point was the parameter set listed in table 6.8. It was obtained with some early driving tests that were not conducted very thoroughly and that were just intended to allow the robot to drive somewhat reasonably. The wheel variation tests in section 6.3 were conducted with this preliminary parameter set.

$$\begin{array}{ll} k_{n,P} = 1000 & k_{n,I} = 800 \\ k_{I,P} = 100 & k_{I,I} = 50 \end{array}$$

Table 6.8: Motor controller tuning: Initial parameter set

Current Controller Tuning

It is reasonable to tune the DES current controller first, because it can be done independently of the speed controller. The robot was lifted off the ground to allow the wheels to spin freely. A DES was switched to current regulation mode and a step change of 500 mA was applied to the set current I_{set} starting with an idle wheel and zero current. The actual effective motor current I_{mot} was measured inside the DES and was communicated to the data acquisition system as the variable I_{DES} via CAN bus. This limited the sampling period to 1 ms. The value for the new set current I_{set} was cho-

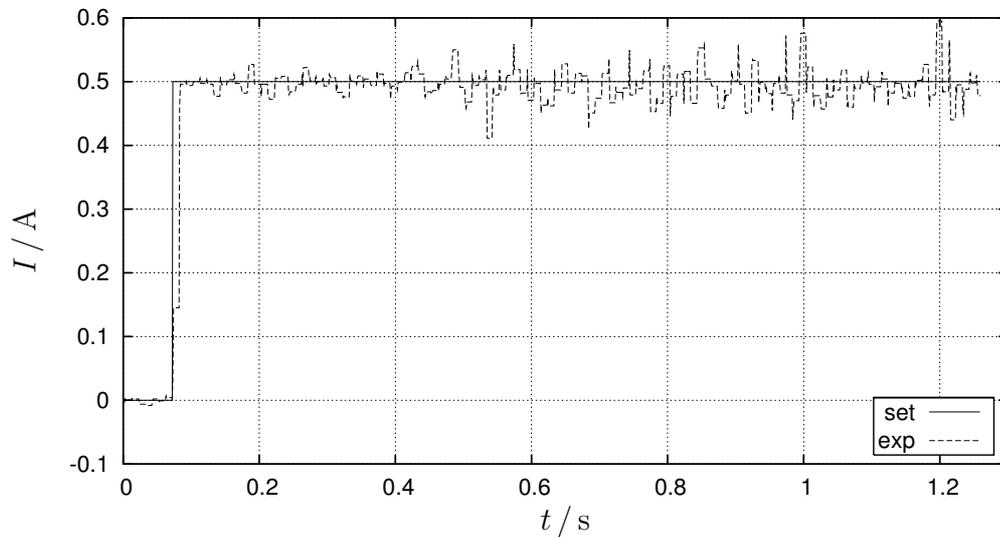


Figure 6.29: Motor current controller: Initial step response

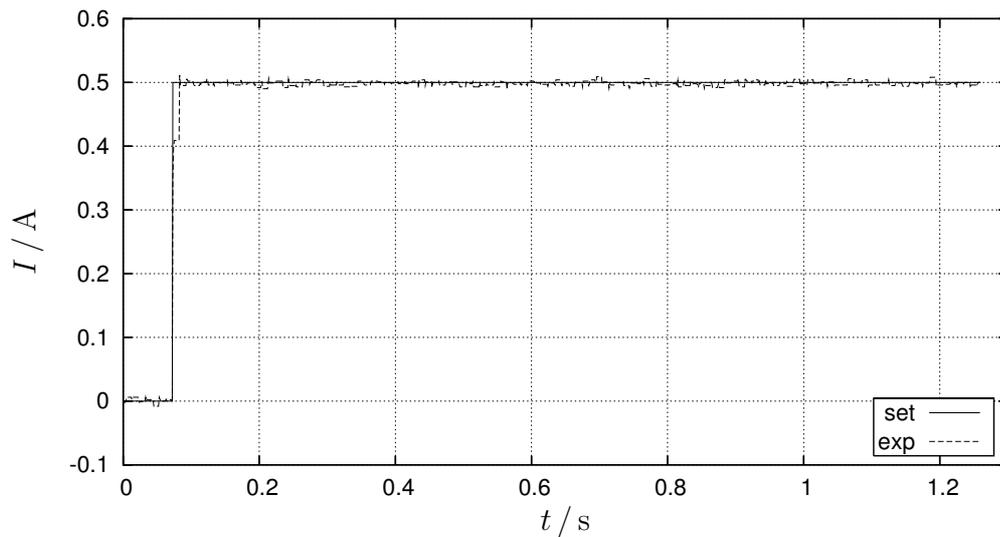


Figure 6.30: Motor current controller: Final step response

sen as high as possible. The value 500 mA seemed reasonable for this test. For higher currents, the wheel speed increased too rapidly.

Figure 6.29 plots the results of this test for the first 1.3 s while the wheel was constantly accelerating. The rise time after which the actual measured current reached 95 % of the set value is approximately 12 ms. After that, the measured current oscillates at a very high frequency around its set value. The amplitude of this oscillation is about 50 mA which is 10 % of the set value.

The amplitude seems to be increasing with respect to time, this effect may be related to the increasing rotor speed. Recall that the measured value I_{DES} is not filtered, and the high frequency oscillations are physically possible. A complete series of tests with different parameter settings was conducted, the two controller gains were varied systematically in order to find an optimal set. In the beginning, the integrator was disabled by setting $k_{I,I}=0$ and a good value for $k_{I,P}$ was determined. For values of $k_{I,P}$ below 100, the rise time increased significantly. For larger values, the rise time never dropped below 11 ms. Activating the integrator did not reduce the rise time either, nor did it produce any typical overshoot behavior. Therefore, the amplitude of the oscillations around the steady state set value following the rising transient was used as a second criterion for assessing the controller parameter set. This is the reason for the large time scale in figures 6.29 and 6.30 that does not emphasize the rising transient at the beginning. Many improved results were obtained and finally the parameters $k_{I,P}=2000$ and $k_{I,I}=500$ were chosen for all subsequent experiments. The resulting step response is plotted in figure 6.30. The 95% rise time of the actual measured current could be slightly reduced to 11 ms. A major improvement was obtained with respect to the oscillations, their amplitude reduced to less than 10 mA which is 2% of the set value.

Speed Controller Tuning

The speed controller parameters were determined with the robot driving the *eight* slalom. They cannot be tuned with the robot lifted up because the robot's inertia must be accounted for. Two criteria were monitored to evaluate the controller performance. The first criterion was the course of its input and output variables with respect to time. Input variables are the actual rotor speed n_{DES} determined with the encoder, and the set speed n_{set} created by the DES speed ramping function. Output variable is the set current I_{set} . It was converted to a wheel torque and compared to the calculated torque from the trajectory generation. The second criterion was the actual course deviation of the robot that can be viewed as a trajectory in the x - y coordinate system. The following tests produced data at a sampling rate of 40 ms because the robot's internal state variables are updated at the period of the video image recognition. This includes communication with the DES.

The testing trajectory was created with the CAT method using the parameters $\mathbf{\Lambda} = [2, 2, 2, 2]$ and $\mathbf{\Gamma}_{\text{CAT}} = [0, 1, 0]$. The robot orientation was fixed at the constant value $\alpha = 90^\circ$ because this way, the robot stays most accurately on its predefined course, allowing to observe the effects of the controller tuning with minimum external disturbances. This was found out with a dif-

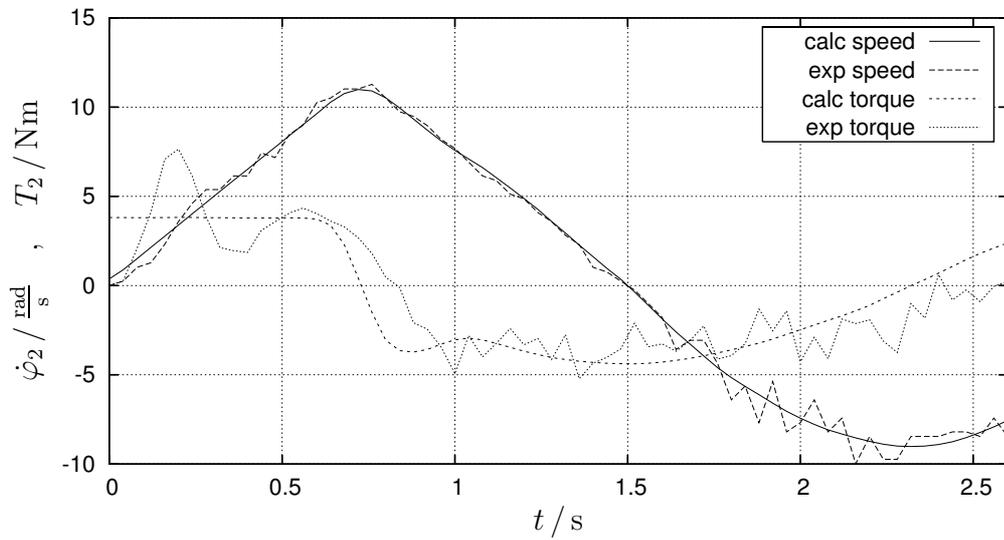


Figure 6.31: Motor speed controller: Initial wheel velocity and torque

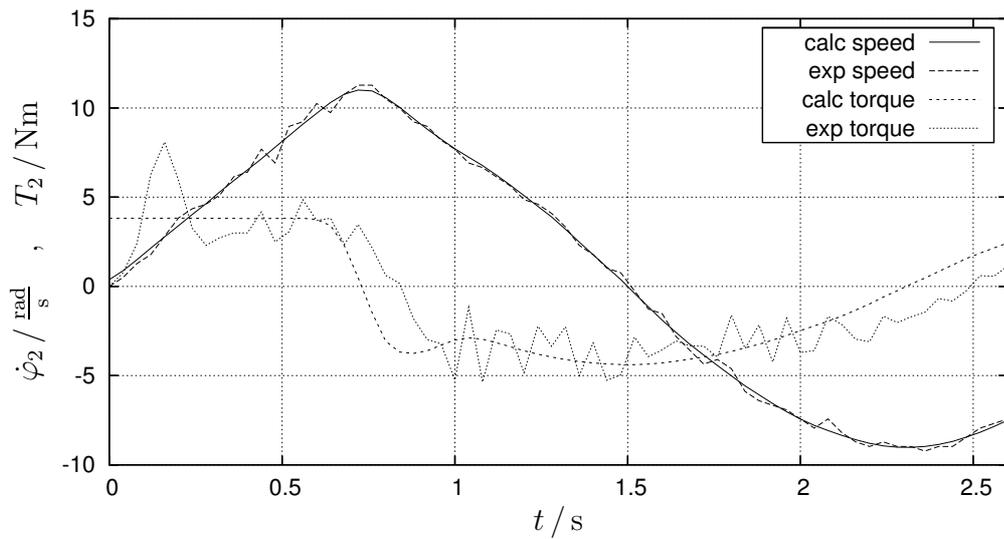


Figure 6.32: Motor speed controller: Final wheel velocity and torque

ferent test series that is not described here in more detail. The reason why the orientation $\alpha = 90^\circ$ is so much better than any other orientation lies in the nature of the *eight* cycle. The two crossing straight diagonal sections during which the longitudinal acceleration and deceleration take place are oriented at angles of $\pm 30^\circ$ with respect to the x -axis. If the robot orientation is $\alpha = 90^\circ$, wheel 1 is facing in direction of the trajectory at the start. Wheels 2 and 3 are loaded symmetrically when the robot is taking off from

the start which causes minimal disturbances. When the robot is driving on the straight line from the fourth to the second quadrant in x - y coordinates, there is a similar situation. Now wheel 2 is facing in driving direction while wheels 1 and 3 are symmetrical at the rear end relative to the driving direction. The robot motion controller gains were $k_{xy} = k_\alpha = 0$, the feed forward compensation used the angle α_{EXT} to compute the wheel speeds.

Figures 6.31 and 6.32 show the results of the first criterion for the speed controller before and after tuning, respectively. The angular speed and torque are plotted for wheel 2 representing motor speed and motor current of motor 2. The actual wheel speed $\dot{\varphi}_2$ was calculated from the motor speed $n_{2,\text{DES}}$, the actual wheel torque was calculated from the speed controller output set current I_{set} using equation (2.14), p. 36. Recall that in section 6.4, the measured effective motor current I_{DES} was used to compute the wheel torque, but this value is delayed by the response time of the current controller. Now it is possible to compare the actual wheel torque produced by the speed controller with the calculated torque from the drivecycle according to equation (1.51), p. 22. During each experiment, all three DES were operated with the same controller parameters. The test results in both figures 6.31 and 6.32 look very similar, they can hardly be distinguished. Therefore, the following discussion holds for both figures. At the beginning, there is a constant acceleration phase that lasts about 0.6 s, the calculated speed $\dot{\varphi}_2$ ramps up with a constant rate. During that time, a constant torque $T_2 = 3.8 \text{ Nm}$ is required at wheel 2. For the first 0.2 s the actual speed falls behind the calculated speed, this causes a control deviation of the motor speed Δn according to equation (4.2), p. 73. This deviation feeds the integrator and the proportional gain of the speed controller which in turn increases the set current I_{set} . This is represented by the increase of the actual torque in both figures. Note that the integrator was initialized to zero by the DES at the beginning of the test. The integrator value can not be initialized externally. The DES set current has a significant overshoot at about 0.2 s. Its maximum occurs when the actual speed crosses the calculated speed because from that time on, a negative speed deviation enters the integrator. When the speed controller output is about to settle near the calculated value at 0.7 s, the acceleration period is already over and a negative torque is required to ramp down the wheel speed. The actual wheel speed follows the calculated speed very precisely in both figures 6.31 and 6.32.

It was tried to optimize two aspects in the time domain: The response of the controller output current I_{set} , specifically its overshoot and undershoot, and the high frequency oscillations in both speed and current. These oscillations can only be vaguely interpreted from the two figures. There are some periods of time when a number of subsequent sampling points is above and

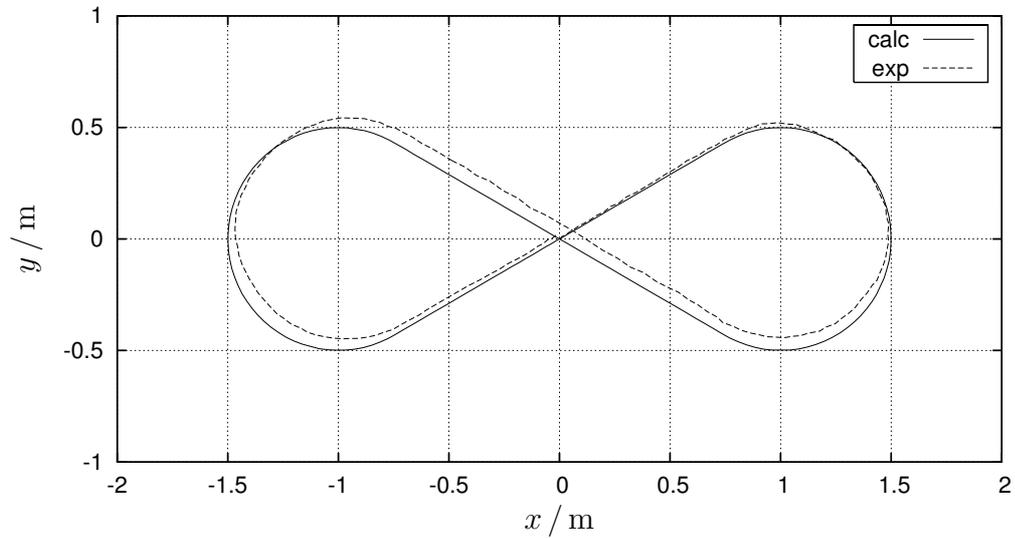


Figure 6.33: Motor speed controller: Initial trajectory

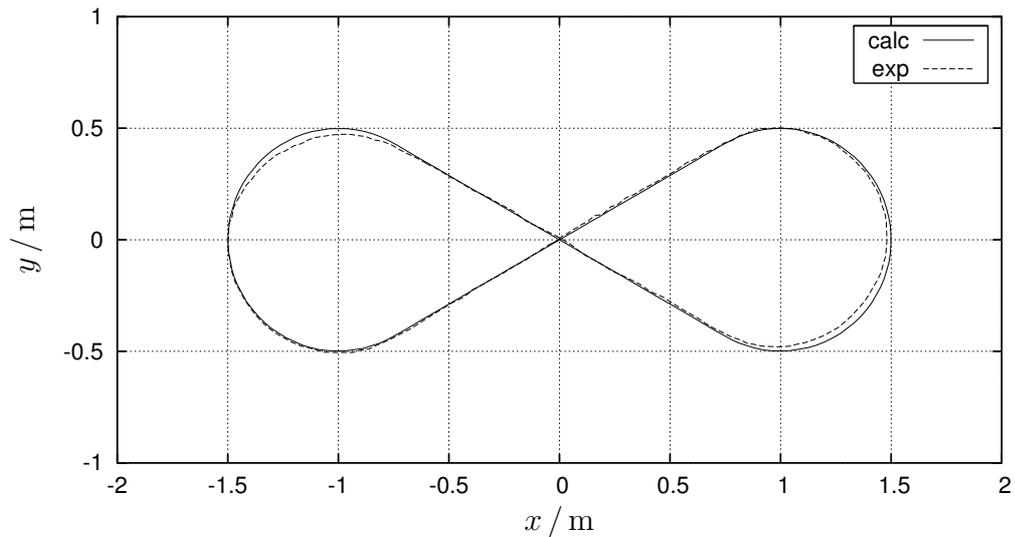


Figure 6.34: Motor speed controller: Final trajectory

below the calculated value, for example around 2.0s in figure 6.31. Unfortunately, the sampling frequency of the robot was too low to capture what was going on in more detail. Again, a whole series of tests was conducted and the parameters $k_{n,P}$ and $k_{n,I}$ were altered systematically. It was not possible to find parameters that were able to reduce the response time of the current at the start significantly. A too large value of the integrator gain $k_{n,I}$ created a large overshoot in the set current, on the other hand the controller was

not able to follow the speed ramp if the value of $k_{n,I}$ was too small, causing the robot to leave its course. The overshoot and the undershoot behavior of the set current could be slightly improved. It seems that the actual torque settles a few milliseconds earlier after the tuning, but this observation is not very well founded due to the low sampling rate. The high frequency oscillations are not very pronounced in either of the two figures, they were present with much higher amplitudes in tests with different parameters. They can be seen more clearly in figures 6.26 and 6.27 with a different trajectory. No correlation could be observed between the oscillations and the overall robot performance. They do not seem to impair the robot's accuracy in staying on its course. Overall, it can be said that figures 6.31 and 6.32 do not offer much information on the improvement of the speed controller's performance. Therefore, the second criterion is now discussed by evaluating the actual position of the robot trajectory in the x - y coordinate system.

Figures 6.33 and 6.34 show the robot's trajectory before and after tuning of the DES speed controller parameters $k_{n,P}$ and $k_{n,I}$. The improvement in the accuracy to follow the calculated trajectory is obvious. The test was repeated a number of times to ensure reproducibility. Both figures can also be compared to the result presented in figure 6.24 which used the same trajectory shape and speed parameters, and the same double roller wheel type. The significant improvement in accuracy compared to figure 6.24 was obtained with two modifications: First, the robot orientation is constantly 90° in the experiments described in this section. The advantages were explained before. Second, the robot motion controller utilizes the DES ramping ability in all experiments in this section and it uses the new optimized DES controller parameters. Table 6.9 summarizes the DES controller parameters that were produced as results of the experiments in this section.

$$\begin{array}{ll} k_{n,P} = 3000 & k_{n,I} = 2000 \\ k_{I,P} = 2000 & k_{I,I} = 500 \end{array}$$

Table 6.9: Motor controller tuning: Final parameter set

6.5.2 Robot Position Estimation

This section investigates the quality of the robot position prediction that is described in section 4.3. Recall that the robot's actual position is determined with the video image recognition. The evaluation of this sensor signal has a delay of approximately 60 ms. Therefore, the recent robot positions are extrapolated to obtain the actual robot position required for the motion controller. The MCL method according to section 3.5 was used to create

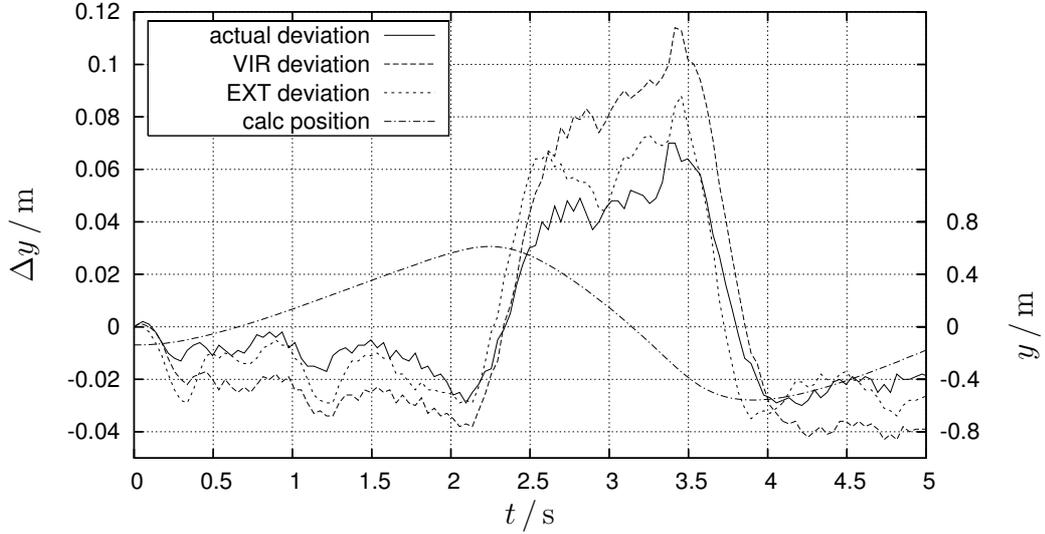


Figure 6.35: Robot position deviation estimation

the trajectory for this experiment because one concern with the utilized extrapolation method was that in narrow curves the extrapolation would not predict the curvature precisely and would therefore produce extrapolated robot positions outside the actual trajectory. The MCL method produces smaller local curve radii that are not constant, and therefore, its smallest local radius is smaller than the constant curve radius produced by the CAT method for an equivalent trajectory. The control points in table 6.2 were used to generate the trajectory shape, the parameters $\mathbf{\Lambda} = [1.5, 2.5, 1.5, 0.5]$ and $\mathbf{\Gamma}_{\text{MCL}} = [0.8, 5]$ determined the robot's position. The data is taken from the fast dribbling experiment discussed in section 6.6. The motion controller was active during this experiment, its gains were $k_{xy} = k_{\alpha} = 1 \text{ s}^{-1}$. The DES controller parameters were chosen according to table 6.9.

Figure 6.35 shows the robot position and deviations in y -direction for the first loop of this drivecycle which lasted about five seconds. The three ragged curves are all robot deviations Δy , they map to the left ordinate. The smooth solid curve with the sine shape is the calculated robot position from the drivecycle, its values are shown on the right ordinate. This curve has been added to the graph for orientation. The ragged solid curve represents the actual deviation of the robot. This is the most accurate deviation that could be obtained with the video image recognition and it serves as a reference here. It was obtained by shifting the robot positions \mathbf{x}_{VIR} backwards in time to the time when the respective video images were exposed in the cameras. This can not be done in real time, so this deviation will never be available to the robot when it is driving, instead the data was processed after the experiment was

completed. The actual course deviation Δy acquires a negative value while the robot is moving in positive y -direction, it changes to a positive sign when the robot starts moving in negative y -direction at 2.2 s, and changes back to a negative sign at 3.9 s when the robot starts moving in positive y -direction again. One reason for this behavior could be slip between wheels and the floor: During acceleration the robot falls behind while it catches up during deceleration. The maximum value of the actual deviation in figure 6.35 is about 0.07 m after 3.4 s of driving.

The curve labelled 'VIR deviation' is computed in real time by the robot. It ignores the delay of approximately 60 ms in the video signal chain and assumes that the last robot position \mathbf{x}_{VIR} is up to date. This will add an error proportional to the robot velocity because it does not account for the distance that has been driven in the respective time interval. This behavior can be read from figure 6.35: The error of the VIR deviation relative to the actual deviation is zero at the start, then it grows and reaches a maximum value simultaneously with the maximum velocity in y -direction at about 1.4 s. When the position y has reached its maximum at 2.2 s, the error of the VIR deviation compared to the actual deviation is zero because the robot velocity in y -direction is zero. The absolute value of the VIR deviation is always greater than that of the actual deviation in figure 6.35, and they have their zero crossings at approximately the same time with the velocity. The largest error was produced by the VIR method at about 3.45 s. The actual deviation in y -direction was 0.063 m at that time, while the VIR suggested a deviation of 0.113 m, resulting in an error of 0.050 m.

The EXT deviation computed with the weighted polynomial approximation extrapolation described in section 4.3 produces better results. Most of the times it predicts a robot course deviation that is still larger in its absolute value than the actual deviation, but it is closer to the actual deviation than the VIR assumption. At 2.5 s, the VIR extrapolation computes a deviation that is not as good as the EXT value. This could be the product of the trajectory curvature effect mentioned before. The negative curvature of the graph of $y(t)$ causes the extrapolation to produce a position estimation that is too large. Therefore, the VIR deviation is greater than the EXT deviation. At 3.9 s, the same phenomenon can be observed with opposite signs. The largest error was produced by the EXT method at about 2.5 s. The actual deviation in y -direction was 0.031 m at that time, while the EXT deviation predicted 0.064 m, resulting in an error of 0.033 m.

Summarizing the above results it can be concluded that the robot position estimation produces reasonable results, but there is still potential to improve its accuracy. One approach is to take the robot velocity into account which can be obtained from the drivecycle or from the wheel speeds measured

with the DES. In the approach that has been implemented at this point, it was tried to base the position and deviation computation on the results of the video image recognition exclusively, ignoring any information from other sensors or from the drivecycle. An alternative approach that incorporates information from new sources will be reconsidered for future improvements.

6.5.3 Robot Motion Controller Tuning

This section investigates the performance of the robot motion controller that was discussed in section 4.2. Again, the *eight* slalom was used as benchmark trajectory. This time, both trajectory generation methods, the MCL and the CAT method, were used to create the drivecycles. It was the purpose to find out if one of the two methods creates trajectories that can easier be followed by the robot. The same control points as in section 6.1 were used to determine the trajectory shape, the parameters $\mathbf{\Gamma}_{\text{MCL}} = [0, 10]$ and $\mathbf{\Gamma}_{\text{CAT}} = [0, 10, 0]$ were used to determine the robot orientation and position for the MCL and the CAT method, respectively. Note that the weight factor ψ is zero in both methods, which means that the robot trajectory coincides with the generic trajectory. Consequently, the velocity and acceleration constraints represented in the parameter vector $\mathbf{\Lambda}$ directly apply to the robot's motion. Both methods are using a damping ratio of $\delta = 10 \text{ s}^{-1}$. This causes the viscous damping to dominate the robot orientation and therefore, the robot will be oriented almost in tangential direction. The reason why this high damping ratio was chosen is similar to the motivation to use a constant orientation $\alpha = 90^\circ$ for the speed controller tuning described in section 6.5.1. When the robot is accelerating and decelerating in longitudinal direction, it stays more accurately on its predefined course if there is a symmetry with respect to the tangential direction. At an orientation of $\alpha = 90^\circ$, the two respective rear wheels in driving direction were symmetric with respect to the tangential direction on the two crossing straight sections of the *eight* slalom. With the high damping ratio that was used here, the two front wheels were almost symmetric. Note that the tests in this section and in section 6.5.1 were carried out with both orientations, but only some representative results were selected for discussion.

Open Loop Slow Driving

Before the controller was tested, the *eight* course was driven with the controller gains k_α and k_{xy} set to zero. This open loop test served as a reference for the subsequent test with the position controller enabled. The velocity for the open loop test was chosen not too fast with the parameter $\mathbf{\Lambda} = [2, 2, 2, 2]$

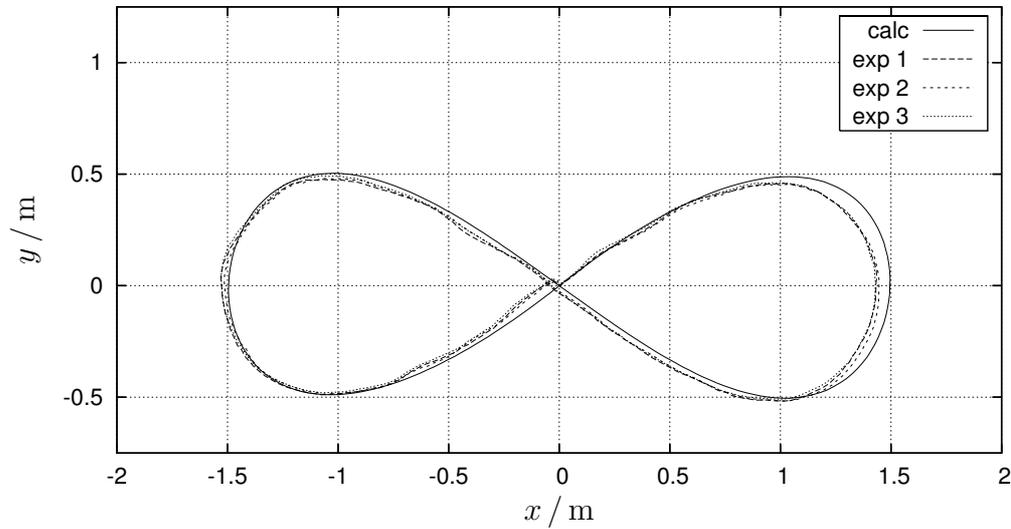


Figure 6.36: Robot motion controller: MCL trajectory, slow, open loop

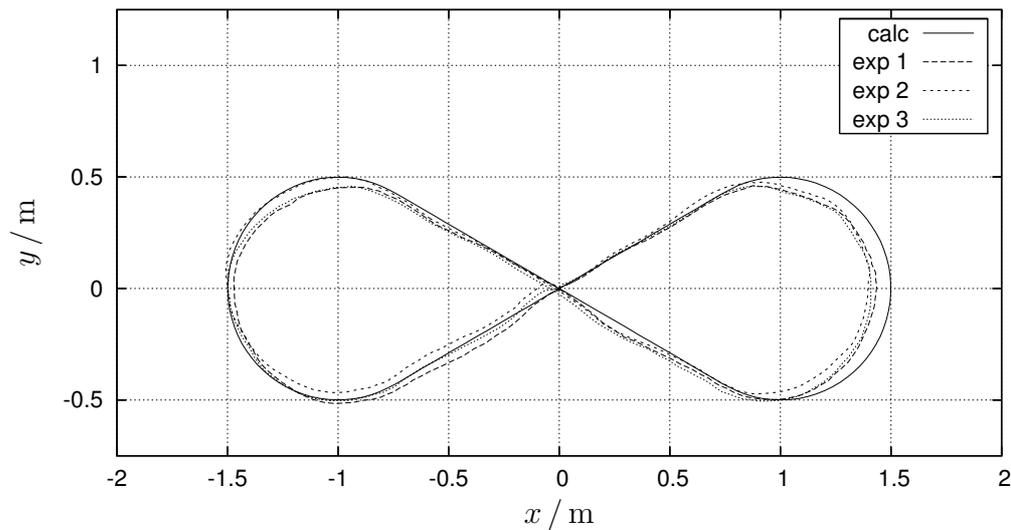


Figure 6.37: Robot motion controller: CAT trajectory, slow, open loop

for both methods. This produced a maximum translational velocity 0.88 m/s for the MCL method and 2.00 m/s for the CAT method. These two velocities can not be compared because the maximum velocity produced by the MCL method is limited by the minimum curve radius $\hat{\rho}$ and the allowed lateral acceleration constraint a_{lat} . The maximum velocity of the CAT method occurs at the origin after the first loop between the acceleration and the deceleration phase. This phase does not exist in the MCL method. The CAT trajectory

has a velocity of 1.00 m/s in its circular sections. This velocity can be compared to the MCL method. It is larger because the constant CAT curve radius is larger than the smallest local MCL curve radius $\hat{\rho}$. The maximum coefficient of friction $|\hat{\mu}|$ at the wheels is another good measure to evaluate how demanding a drivecycle is, because it reflects the ability of the wheels to apply their circumferential forces to the floor. The MCL trajectory demands $|\hat{\mu}_{\text{MCL}}|=0.43$, the CAT trajectory requires $|\hat{\mu}_{\text{CAT}}|=0.56$. Each test consisted of one full round starting and stopping at the origin with zero velocity, the DES were initialized with the controller parameters listed in table 6.9.

Figures 6.36 and 6.37 show the results of the open loop tests for the two methods, each test was repeated three times. The test results in both figures are quite good considering the fact that the robot does not react to its course deviations. Comparing the trajectories obtained with the three tests in each figure, it can be stated that the reproducibility is excellent. The maximum deviation in lateral direction occurs at $x=1.5$ m in both cases, it is about $\Delta x=0.07$ m for the MCL method and $\Delta x=0.09$ m for the CAT method. Apart from that, it can not be observed that one of the methods seems better suited for the robot to stay on course. As mentioned before, the results in figure 6.37 can be compared to those from the wheel variation in figure 6.24, p. 130, and to those from the DES speed controller tuning in figures 6.33 and 6.34, p. 144. Note that the test results here are not as good as those with the same DES parameter set in figure 6.34. The only reason for this is the different robot orientation.

Closed Loop Slow Driving

The next step was closing the robot's position deviation control loop by applying controller gains k_α and k_{xy} . A series of test was conducted to find the best values, and finally the gains $k_{xy}=k_\alpha=1 \text{ s}^{-1}$ were identified to create very good performance. The drivecycle used in the preceding open loop experiment was repeated three times again, the resulting robot trajectories are plotted in figures 6.38 and 6.39. The robot exhibits an impressive accuracy to stay on its calculated course now for both methods. In both figures, the robot trajectory shows a minor oscillation around the calculated trajectory, they are probably produced by the robot position controller. It is interesting to mention that the shape and position of these oscillations is the same for all three tests, strongly underlining the reproducibility. It seems that the oscillations are more pronounced in figure 6.39 but there is no obvious explanation for that.

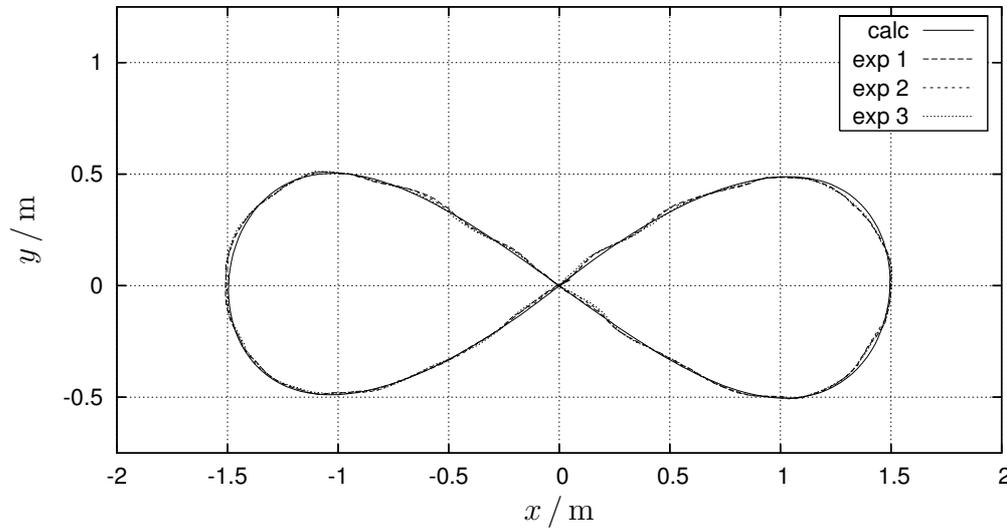


Figure 6.38: Robot motion controller: MCL trajectory, slow, closed loop

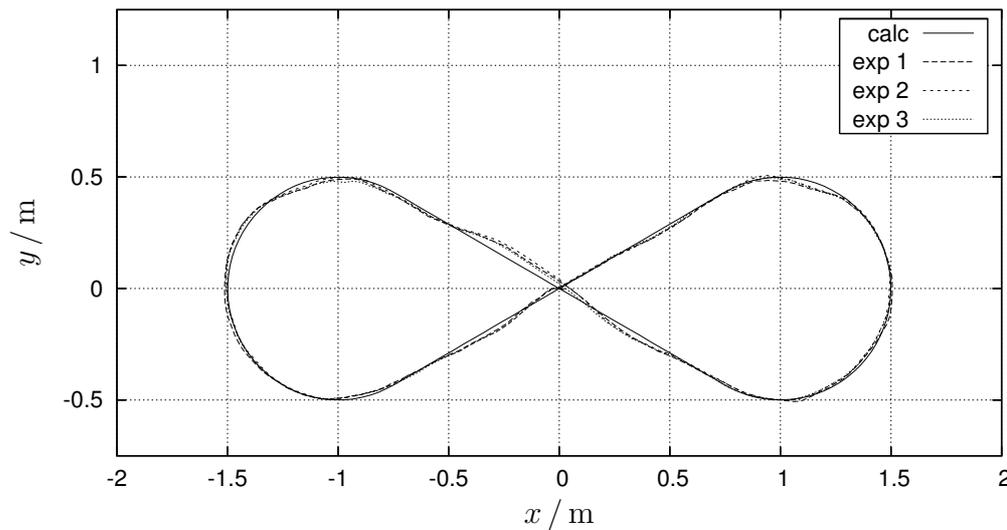


Figure 6.39: Robot motion controller: CAT trajectory, slow, closed loop

Closed Loop Fast Driving

The convincing results with the closed loop slow driving encouraged to increase the velocity to determine the limits of the robot and its motion controller. The constraints on maximum velocity v_{\max} and lateral acceleration a_{lat} which limit the velocity in curved sections were doubled, the new velocity parameter $\Lambda = [4, 4, 2, 2]$ was used for both methods. The result-

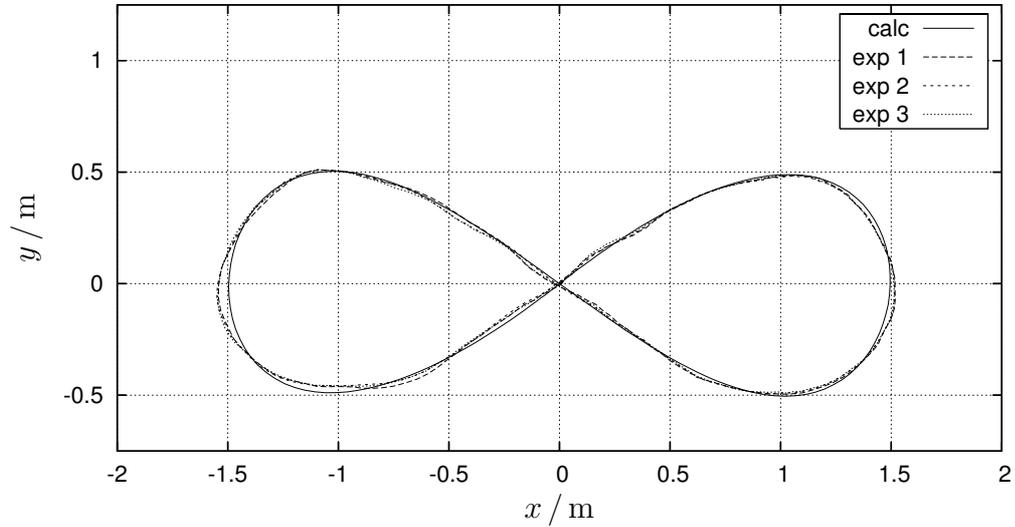


Figure 6.40: Robot motion controller: MCL trajectory, fast, closed loop

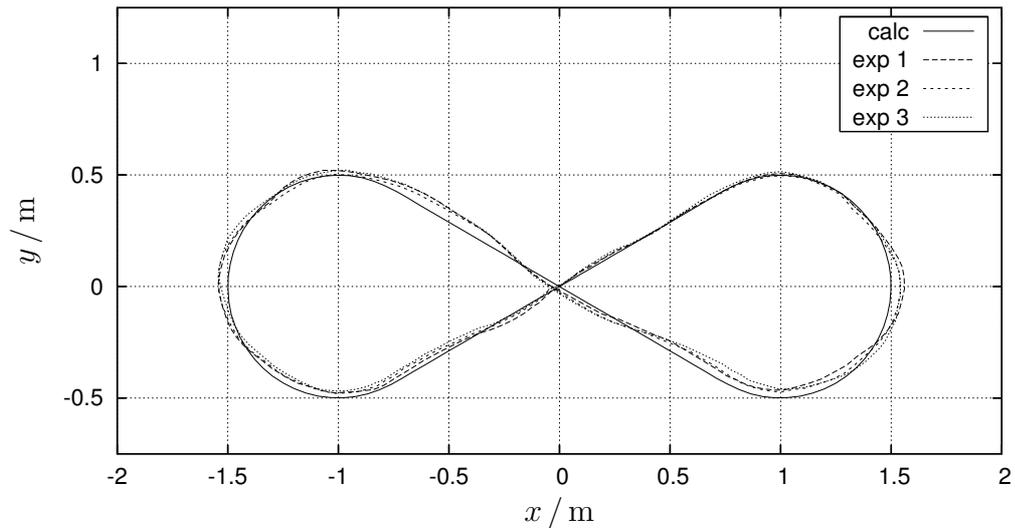


Figure 6.41: Robot motion controller: CAT trajectory, fast, closed loop

ing maximum translational velocity with the MCL method increased by a factor $\sqrt{2}$ to the value 1.23 m/s because it is limited by the lateral acceleration a_{lat} . For the same reason, the CAT velocity increased to 1.41 m/s in its circular sections. The new maximum velocity with the CAT method increased by a much smaller factor to 2.34 m/s because it is influenced only indirectly by the parameter Λ . The maximum coefficients of friction increased significantly, they are now $|\hat{\mu}_{\text{MCL}}| = 0.72$ for the MCL trajectory and

$|\hat{\mu}_{\text{CAT}}|=1.03$ for the CAT trajectory.

Figures 6.40 and 6.41 show the results of the fast closed loop test for both methods. Again, each test was repeated three times. The controller gains were kept $k_{xy}=k_{\alpha}=1\text{ s}^{-1}$. Again, the results are remarkably good, and again, the robot stays more accurately on the MCL course with less deviations in lateral direction. Both drivecycles show oscillations of the robot position in lateral direction around the calculated trajectory as they did before with the smaller velocity. This time, the amplitudes of the oscillations are more pronounced, and again, the oscillations of all three respective test runs seem to be in phase. The fast closed loop driving tests were also repeated with different robot orientations and it must be mentioned that the results plotted in figures 6.40 and 6.41 were the best ones obtained. However, the robot never came off the test field with any of the different orientations.

Figures 6.42 and 6.43 plot the calculated drivecycle velocity and the controller output velocities with respect to time. The maximum drivecycle velocities can be read from the graphs for both methods, as well as the constant velocity plateaus of the CAT methods in the two circular sections. The CAT method is almost two seconds faster because of its higher velocity. Both trajectories have almost the same length of about 7.7 m. The controller velocity $v_{\text{ctr}}=(\dot{x}_{\text{ctr}}^2+\dot{y}_{\text{ctr}}^2)^{1/2}$ is plotted for each of the three respective tests in both figures. The controller velocity was smaller than 0.1 m/s and 0.25 m/s at all times for the MCL and the CAT trajectory, respectively. With a translational controller gain $k_{xy}=1\text{ s}^{-1}$, the respective maximum course deviations can be calculated as 0.1 m and 0.25 m, respectively. Note that these course deviations are only partially in perpendicular direction to the trajectory, most of the deviation occurs in longitudinal direction when the robot is falling behind the scheduled position. This longitudinal contribution can not be seen in the previous x - y plots. The maximum deviation in the CAT course occurs in the middle during the extra acceleration and deceleration phase. It is not appropriate to compare this deviation to the MCL method where this phase does not exist. The maximum CAT course deviation outside the acceleration and deceleration phase is 0.15 m. The maximum controller velocity relative to the maximum calculated drivecycle velocity is 8 % and 11 % for the MCL and the CAT test, respectively.

The robot controller experiments discussed in this section produced very satisfying results. Not only could be demonstrated that the robot shows an excellent open loop behavior, but also, the controller managed the robot to stay on course even at high velocities. The maximum friction coefficient $|\hat{\mu}_{\text{CAT}}|=1.03$ is considered a real challenge that was overcome successfully. It was not possible to increase the lateral acceleration constraint to $a_{\text{lat}}=5\text{ m/s}^2$ according to the dimensioning specifications listed in table 2.3 because the

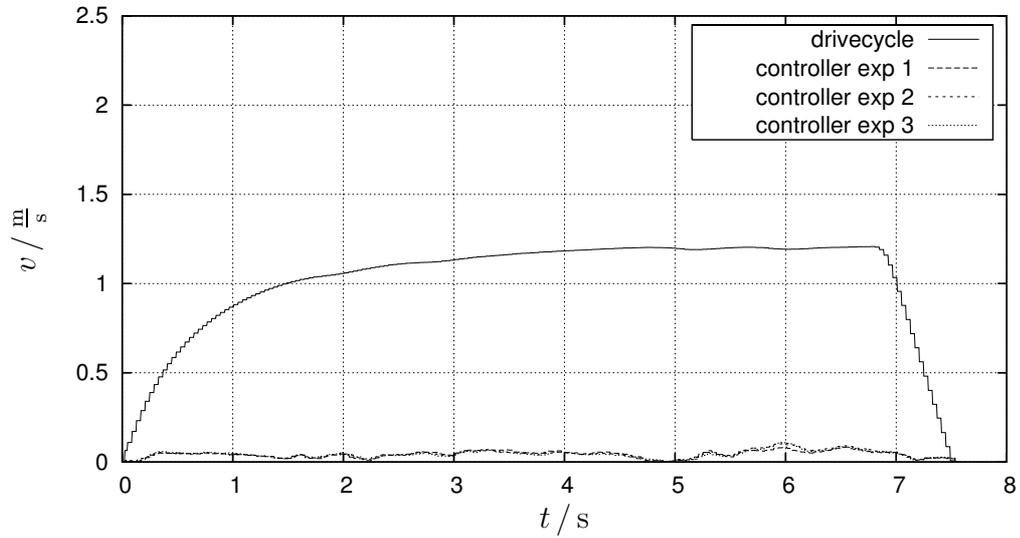


Figure 6.42: Robot motion controller: MCL controller velocity, fast

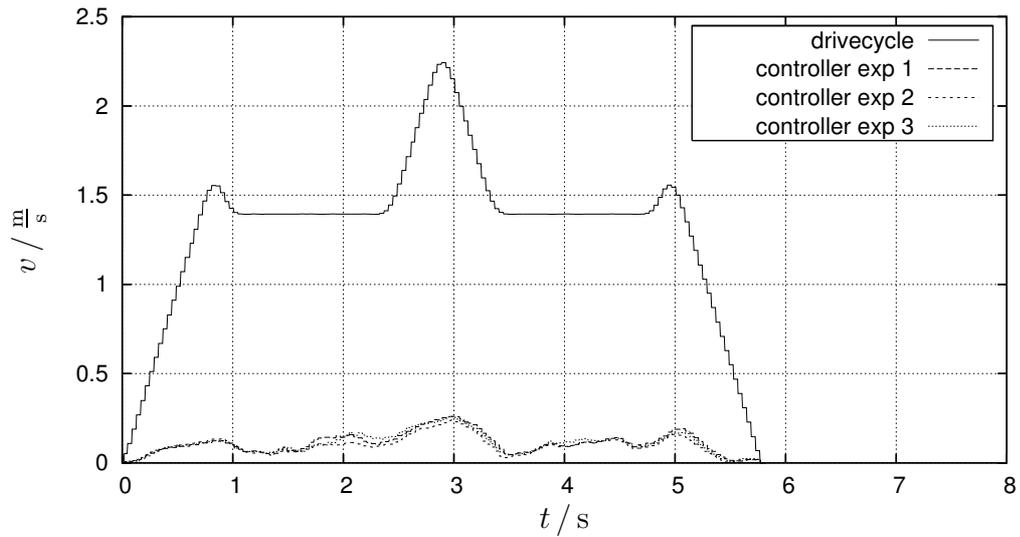


Figure 6.43: Robot motion controller: CAT controller velocity, fast

robot became unstable and left the field. The reason for this is that one wheel lifted off the ground and created an uncontrolled degree of freedom, allowing the robot to travel in an undesired direction. Before this test can be repeated, the center of mass needs to be lowered to reduce shifting of the dynamic wheel loads. At this point, there is no need identified to improve the robot position controller, and the approach with a simple proportional gain seems fully justified.

6.6 Ball Handling

This experimental section investigates the robot's capability of fulfilling its primary objective: To solve the Two Body Problem defined in section 3.1. After all parameters related to the robot's hardware components had been determined in the previous experiments, the ball dribbling tests presented here were the last experiments in chronological order. The robot was equipped with the double roller wheels depicted in figure 1.2, p. 10. The ball dribbling mechanism is shown in figure 2.3, p. 38, it can also be seen in the photograph of the complete robot system in figure 1.9, p. 26. The motor controller parameters were set to the values listed in table 6.9, p. 145, the robot position controller gains were set to $k_{xy} = k_{\alpha} = 1 \text{ s}^{-1}$. Both trajectory planning methods were tested for their dribbling capabilities, the minimal curve length (MCL) method and the circle and tangent (CAT) method. The trajectories were designed to pass through the common points listed in table 6.1, p. 98.

The experiments started with a slow maximum velocity defined by the parameter vector $\mathbf{\Lambda}$ according to equation (6.8), p. 101. The parameter vector $\mathbf{\Gamma}$ which defines the robot's position and orientation relative to the trajectory was modified for each set of $\mathbf{\Lambda}$ to obtain the best ball handling during dribbling the *eight* slalom. Then the parameters v_{\max} and a_{lat} represented in the vector $\mathbf{\Lambda}$ were increased gradually for each test series to find out how the robot handles the ball at higher velocities. This was the first time the deceleration parameter a_{dec} was used for a meaningful reason, because the robot drove two rounds with the ball and then slowed down to stop at the end, trying not to lose the ball. If the deceleration value was too high, the robot lost the ball because the friction imposed by the dribbling mechanism was not large enough to slow down the ball's inertia. The acceleration parameter a_{acc} was chosen not too high to make sure that there was not too much slip at the start causing the robot to fall behind its scheduled course.

Slow Dribbling

Two representative test results with different velocities are discussed in the following. They are extracted from a series of test runs and they are referred to as slow and fast dribbling tests. The slow dribbling experiment used the velocity parameter $\mathbf{\Lambda} = [1.5, 1.8, 1.5, 0.2]$. The best dribbling performance was obtained with the parameters $\mathbf{\Gamma}_{\text{MCL}} = [0.8, 1]$ for the MCL method and $\mathbf{\Gamma}_{\text{CAT}} = [0.8, 1, 0.2]$ for the CAT method. The robot reached maximum translational velocities of 1.31 m/s and 1.28 m/s during the trajectories created with the MCL and CAT method, respectively. In both cases, the lateral acceleration a_{lat} limited the top velocity. The two maximum velocities can be

compared for the two methods in this case because the CAT method reaches its maximum robot velocity in the circular sections, not in the straight sections. This is because the weight factor that determines the robot's distance relative to the generic trajectory was chosen as $\psi=0.8$. Therefore, the robot's trajectory around the curved sections is longer than the generic trajectory which causes a higher robot velocity in the curved sections compared to the generic velocity. The maximum coefficients of friction occurring in the MCL

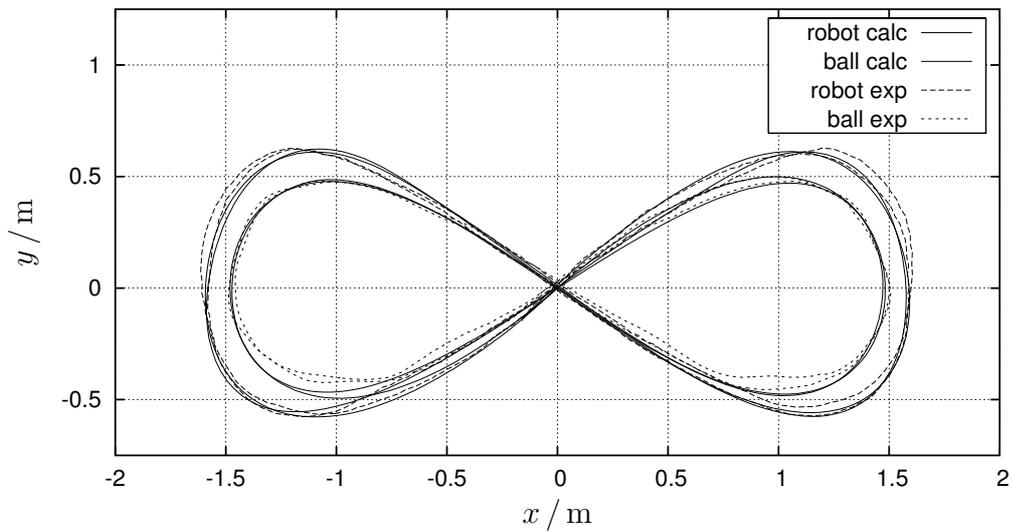


Figure 6.44: Ball dribbling: MCL trajectory, slow

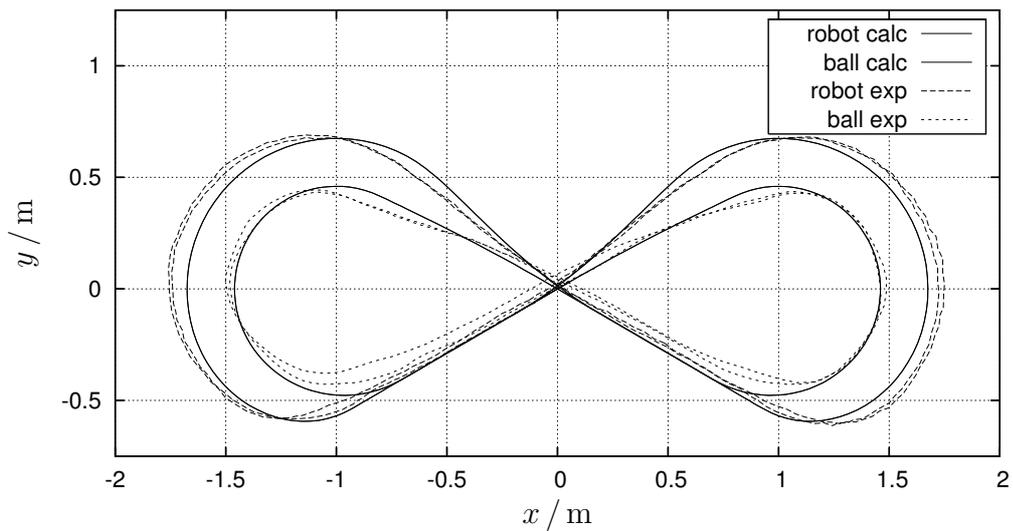


Figure 6.45: Ball dribbling: CAT trajectory, slow

and CAT method are $|\hat{\boldsymbol{\mu}}_{\text{MCL}}|=0.64$ and $|\hat{\boldsymbol{\mu}}_{\text{CAT}}|=0.50$, respectively. The CAT method used a deceleration free interval of 0.2 m before entering a circular section. This produced a smoother transition from the straight into the circular sections.

Figures 6.44 and 6.45 plot the trajectories for robot and ball in x - y coordinates for both methods. In both cases, the robot stayed relatively precisely on its calculated trajectory, sometimes it was outside and sometimes inside the loops. The ball also stayed on its predetermined track quite well, it had a tendency to cut the curves. The most critical point for losing the ball was the transition from a straight to a circular section. With the MCL method, such transitions do not exist due to the nature of the approach using a single continuous polynomial for the complete trajectory. The trajectories created with the CAT method however have four transitions between straight and curved sections of the generic trajectory in each round of the *eight* cycle. These discontinuities are smoothed out a bit by using a non-zero weight factor ψ , because the secant method creates a continuous smooth transition for the robot trajectory. Even with a non-zero weight factor it could still be observed that the ball gets lost in most of the cases at the transition from a straight to a curved section. This behavior could not be observed at the transition points from a curved to a straight section. If the maximum velocity parameter v_{max} was reduced such that the robot did not accelerate and decelerate in the two diagonal straight sections, the ball got lost less often. It therefore was concluded that it is an additional challenge for the robot to enter a curve while it is still decelerating. This conclusion makes sense because during deceleration the ball is pulled out of the robot's dribbling mechanism by its inertia. This finding is the motivation for the deceleration free rolling zone d_0 that is the third parameter in $\boldsymbol{\Gamma}_{\text{CAT}}$ and that was added to the CAT method in the course of the dribbling experiments. It turned out that a deceleration free rolling zone $d_0=0.2$ m is sufficient to reduce the ball losses significantly. The experiment illustrated in figure 6.45 with the CAT trajectory could be extended to 100 rounds without stopping, and the robot did not loose the ball. This was impressive to watch. It is also worth mentioning that both methods worked well with the same damping ratio $\delta=1\text{ s}^{-1}$, underlining the physical background of the interaction between robot and ball. In addition to that, a common weight factor $\psi=0.8$ could also be used in both methods.

Fast Dribbling

The fast dribbling experiment increased the value of a_{lat} , it used the velocity parameter $\boldsymbol{\Lambda}=[1.5, 2.5, 1.5, 0.5]$. In this case, the best dribbling performance

was obtained with the parameters $\Gamma_{\text{MCL}} = [0.8, 5]$ for the MCL method and $\Gamma_{\text{CAT}} = [0.8, 5, 0.2]$ for the CAT method. Now, the robot reached maximum translational velocities of 1.34 m/s and 1.50 m/s during the trajectories created with the MCL and CAT method, respectively. This time, the maximum velocity produced with the CAT method was slightly larger in the straight section than in the circular sections. Note that for the MCL trajectory the value of v_∞ increased by a factor $\sqrt{\frac{2.5}{1.8}} = 1.18$ compared to the slow dribbling trajectory, but the maximum robot velocity only increased by a factor 1.02. This is because the larger value of the damping ratio δ produced a robot trajectory closer to the generic trajectory, resulting in smaller translational robot velocities in curves. The maximum coefficients of friction occurring in the MCL and CAT method are $|\hat{\mu}_{\text{MCL}}| = 0.73$ and $|\hat{\mu}_{\text{CAT}}| = 0.92$, respectively. Both trajectories were discussed in many details in sections 6.1.1 and 6.1.2. The experimental data from the CAT experiment was also used in section 6.4 to plot the wheel speeds and torques, the data from the MCL experiment was used in section 6.5.2 to examine the robot position extrapolation.

Figures 6.46 and 6.47 show the trajectories for robot and ball for both methods. Again, the robot completed two full rounds before it stopped at the origin without losing the ball. The graphs show that the course deviations of both robot and ball did not increase dramatically. They also suggest that the MCL method produces better results because both ball and robot trajectory are smoother and more accurate than those obtained with the CAT method. At this neck-breaking velocity, the ball was lost more often than with the slower velocity mentioned before. The large value of the coefficient of friction $|\hat{\mu}_{\text{CAT}}|$ close to unity indicates that the limit is reached for this method, while there are still some reserves in the MCL method. Subsequent experiments with even higher velocities showed that the physical limits were indeed reached, it was not possible to find a faster CAT trajectory that did not lose the ball, because the robot started skidding. Again both methods used the same damping ratio $\delta = 5 \text{ s}^{-1}$, but it is different from the one found for the slow dribbling tests. For the fast dribbling tests, the robot needed to face over proportionally more in driving direction in order not to lose the ball which could only be achieved with an increased damping ratio. Another less important result was that the deceleration value could be increased to $a_{\text{dec}} = 0.5 \text{ m/s}^2$ and the ball was still not lost at the end.

The conclusion of the dribbling tests presented in this section is simple: The Two Body Problem has successfully been solved. It is very satisfying that the robot can be pushed to its physical limits by increasing the maximum velocity, still being able to dribble a rolling ball along the calculated course.

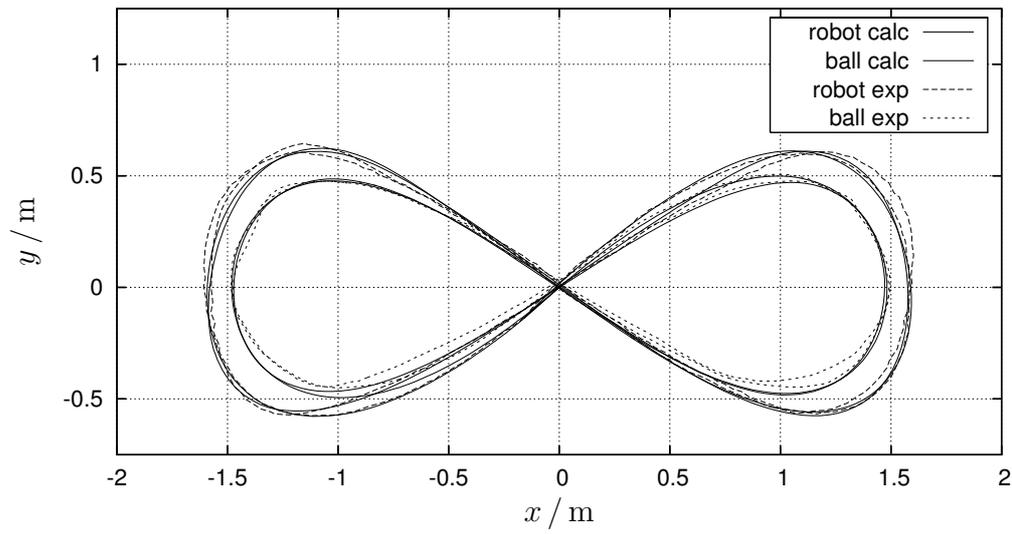


Figure 6.46: Ball dribbling: MCL trajectory, fast

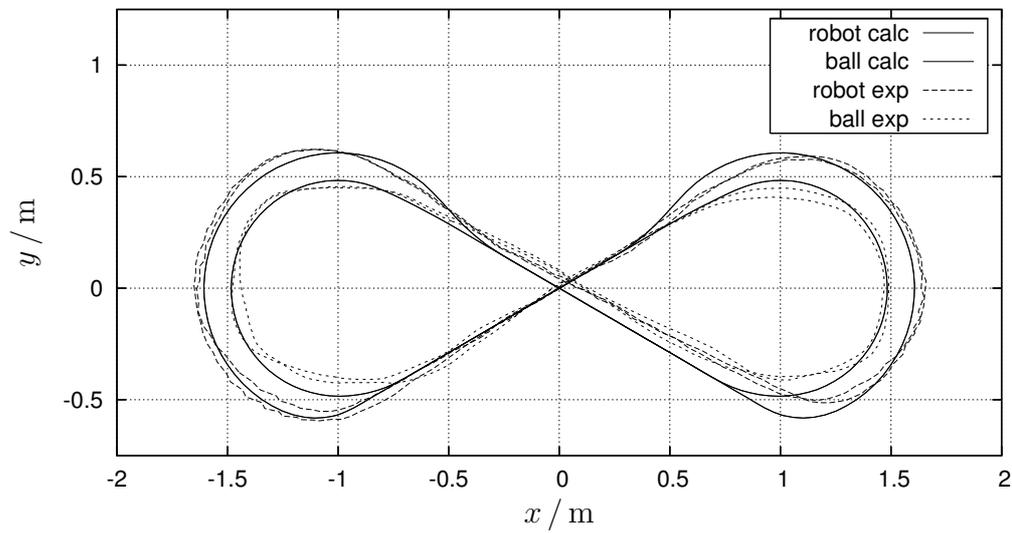


Figure 6.47: Ball dribbling: CAT trajectory, fast

Conclusions

It was the objective of the research project documented in this work to develop a superior robot platform that is equipped with the basic skills required for a successful participation in the RoboCup competition. The two primary skills were identified as orientation and agility. The Two Body Problem was defined as a benchmark problem used to validate the success of the development. It simply postulates that a ball must be dribbled along an arbitrary predefined planar trajectory, one of the most basic requirements that a robot is confronted with during a RoboCup competition. All aspects of this comprehensive challenge were analyzed, and adequate solutions were developed, implemented, and validated. Figure 1, p. 6, shows a schematic drawing of the robot as a mechatronic system, its five major elements are *trajectory planning*, *actuators*, and *robot system* in the forward branch, and *sensors* and *controllers* in the feedback loop.

Trajectory Planning

Two approaches for trajectory generation with different numerical parameterizations were developed and tested, the minimal curve length (MCL) method and the circle and tangent (CAT) method. The primary focus of the MCL method is to create a smooth and bumpless curve that prevents the robot from rapid movements which could cause the ball to get lost during dribbling. The trajectory is created in three steps: The first step determines the geometrical shape of a generic trajectory used as a guideline, the second applies a velocity profile, and the third computes the robot position relative to the generic trajectory. A polynomial parameterization was chosen to determine the geometric shape, because it provides continuous derivatives of the trajectory. Its objective is to minimize the trajectory curve length requiring an iterative process to solve for the optimum solution. The Newton method was applied and it showed good convergence and a robust numerical solution. An extension to the objective function that included an optimization of the integral curvature managed to improve the final result in most cases, but it significantly reduced the convergence behavior. The velocity

profile was also parameterized fully analytically and its maximum velocity was computed such that a lateral acceleration constraint was reached at the point with the smallest local curve radius. The profile incorporates a significant drawback because it always starts at zero velocity and continuously accelerates towards an asymptotic final velocity. This does not provide the freedom required for participating in the RoboCup competition, but it does solve the Two Body Problem. The position of the robot and the ball relative to the generic trajectory are calculated with a physical model of the ball being a passive object that is pushed around. During the dribbling experiments it could be shown that the smoothness of the trajectories created with the MCL method is a great advantage as expected. On the other hand, there is also a number of disadvantages associated with this method, predominantly related to the numerical treatment. First, the method consumes a lot of computational time because large matrices need to be inverted to solve for coefficients, several numerical integrations must be performed to obtain the curve length and its derivatives, and finally, the solution is obtained iteratively which repeats the aforementioned steps a number of times. Second, the method is very sensitive with respect to the choice of control points. Even for the benchmark *eight* slalom course used for the experimental validation which has two axes of symmetry, the choice of control points was not easy. The control points determine the trajectory shape, they are located on the generic trajectory. It is hard to imagine how the choice of control points can be automated in a continuously changing environment like a RoboCup soccer match. Third, the velocity profile along the generic trajectory is very difficult to tackle, especially with the requirement that the derivatives must be continuous. No appropriate solution was found that allows acceleration and deceleration in longitudinal direction, is simple to implement, and fulfills the continuity requirement. Therefore, in a temporary approach, the velocity was increased asymptotically and artificially ramped down at the end of the respective drivecycle. This was a measure to allow the robot to stop at the end of each *eight* slalom during the experiments. However, the discontinuity created by the ramp violates the continuity requirements and can not be accepted in the long term.

The CAT method was developed subsequently. It is an extremely straightforward approach and was designed to overcome the disadvantages of the MCL method. It uses the same three step approach, but it does not have the continuity restriction on the derivatives. The shape of the generic trajectory is assembled piecewise with two geometric elements, circular arcs and straight tangent lines. The center points and radii of the circles are provided as control points, two subsequent circles are connected with their tangent lines. This is a very pragmatic approach because the positions of opposing

robots can be used as control points, and the trajectory will automatically avoid them. The velocity profile is also assembled with simple construction rules, it either accelerates or decelerates with a constant value, or it maintains constant velocity due to active constraints on maximum velocity or lateral acceleration. This is again very pragmatic as it reflects the physical limits of the robot hardware, and it is simple to implement. The robot and the ball position relative to the generic trajectory are calculated with the same physical model of the ball previously used by the MCL method, allowing a direct comparison of the two methods. Simplicity and robustness are the striking advantages of this method. Both trajectory shape and velocity profile can be determined with simple rules, no iteration is required. The choice of control points is also obvious and can easily be automated by a strategy module to participate in a RoboCup competition. Tests with the single round *eight* slalom have shown that the CAT method consumes at least 10 times less computational time than the MCL method in creating a comparable trajectory. The only disadvantage of the CAT method that was expected and could be verified experimentally is the jerky transition from straight to circular sections and vice versa. It is caused by the discontinuity of the generic trajectory's curvature on the one hand and by the piecewise constant accelerations on the other hand. The secant method used to compute the robot's orientation smooths out the bumpy curve entries for the robot trajectory, but at these points, the ball was lost predominantly in the dribbling experiments. This behavior could be remedied with the introduction of a zero deceleration zone before curve entries, because deceleration contributed to an instable ball handling. It can now be said that the CAT method produces very satisfying dribbling results which can be illustrated with some numbers obtained from experimental results: In the slow dribbling experiment, the robot reached a maximum velocity of 1.28 m/s and could dribble a large number of rounds before the ball got accidentally lost. This emphasizes the good reproducibility. In the fast dribbling experiment, the physical limits of the robot were reached at the maximum velocity of 1.50 m/s. This velocity was limited by the high location of the robot's center of mass, because the maximum coefficients of friction reached a peak value of 0.92 and one wheel came close to lifting off the floor. It can be concluded that the CAT approach is the preferred method for creating trajectories.

Actuators

The robot propulsion motors were dimensioned according to the specifications listed in table 2.3, p. 31, and they perform very well. They were over dimensioned with respect to maximum speed and torque for performing the

sample *eight* slalom. This is because the slalom is dominated by narrow curves with short straight sections which does not allow large maximum velocities. In addition to that, the large height of the robot mass center reduces the possible accelerations to prevent the robot from falling over. The experimental results suggest that the motors are not over dimensioned with respect to torque because some peak values caused by oscillations in the motor current reached the theoretical limits. It is therefore concluded that the motors were dimensioned correctly. In combination with the digital servoamplifiers (DES), the motors offer the ability to implement a highly precise odometric system by evaluating the rotor angle increment with digital encoders. The robot's position computed with the odometric system almost perfectly reflected the position according to the calculated drivecycle trajectory. Therefore, the odometric system does not offer additional information while the robot is driving along a calculated drivecycle trajectory. The odometric position is only useful in the experiment for determining the robot physical parameters.

The ball handling mechanism was deliberately kept as simple as possible because the primary focus was to develop a good trajectory planning algorithm first. The sole purpose of the mechanism was to reduce excitations of the ball during dribbling by imposing some friction. It allowed the ball to roll freely during all dribbling experiments which is an important postulation of the RoboCup rule set. There is much room for improvement of the dribbling mechanism. One way to go is to develop a mechanism that allows to dribble the ball by driving sideways or even backwards which is allowed in RoboCup for a limited distance. This would require an active interface between robot and ball, like two rotating wheels that can pull the ball towards the robot.

Both target maximum velocity and acceleration defined in the introduction were reached. Due to battery limitations, they can not be reached at the same time, but this is considered a minor drawback. The robot can accelerate and travel significantly faster than most of the robots participating in the RoboCup tournament.

Robot System

The omnidirectional approach proved a full success. It provides the robot with the agility that is required to solve the Two Body Problem, and the trajectory generation takes full advantage of that fact. With the omnidirectional propulsion system everything seems possible, the only limitations are the laws of physics and the degrees of freedom offered by the trajectory generation. Three different wheel types were investigated, illustrated in section 1.3. The single roller wheel type consists of eight small passive rollers that are dis-

tributed around the circumference of the wheel. Their envelope is a circle. The gaps between two adjacent rollers constitute the major drawback of this design, because they induce vibrations into the robot system when the robot is driving. These vibrations can lead to significant course deviations because they increase the slip between wheels and floor. This could be remedied with a double roller wheel design by adding a second row of rollers filling the gaps of the first row. The robot drives much calmer with these wheels compared to the single roller design, and it stays on its course much more precisely. The only disadvantage of the double roller wheel type is the increased weight and size. The robot equipped with double roller wheels violates the RoboCup design restrictions, because it exceeds the maximum permissible length and width. A redesign of the wheels is required to get approval for participation in an official competition. The third wheel type was designed without rollers to reduce the complexity of the wheel. It was named cutting edge type because it is equipped with a multitude of sharp edges that can cut into a soft surface of the floor which creates a preferred direction along the edges. The drawbacks are the fact that a soft floor is required, the increased power consumption, and the greatly reduced performance due to the increased friction. It was not even possible to complete one full round of the *eight* cycle at a given velocity because the robot completely left its course and came off the field. The best performance was obtained with the double roller wheel type which therefore was used during all experiments.

Sensors

The only sensors that were used are color video cameras because they are the only type of sensor that allows to determine an absolute position on the field without external references. In addition to that, colors play a vivid role in RoboCup which renders cameras absolutely necessary anyway. Three synchronized cameras equipped with wide angle objectives were placed at the top of the robot. Their video images provide an omnidirectional view and are processed simultaneously. An algorithm was developed that uses the white field marking lines for orientation and that computes both the positions of the robot and the ball in a global coordinate system. The algorithm is a multi-step approach that first classifies the colors in the video image in the YUV space, then detects color gradients defining object outlines, then maps the coordinates of all outline pixels to a robot coordinate system, and finally calculates the position of the field markings in robot coordinates. The robot coordinates are in physical units of meters and the mapping is performed with a large lookup-table obtained with an initial calibration procedure. The coordinates of the pixels from all three cameras are mapped into the same

coordinate system. This way they complement each other to yield a complete surround view. The positions of the field marking lines must be known to the robot. It determines its own position by finding the location of the set of marking lines in robot coordinates such that all marking outline pixels match according to an optimality criterion. This criterion postulates that the average squared distance of all marking outline pixels to their respective line is minimal. There is one weakness in this method: It is an iterative process and requires an initial guess for the robot position in order to create the correct pixel–marking line associativity. If outline pixels are associated with the wrong field marking lines, the method will converge to the wrong solution for the robot position. Apart from that, the implemented video image processing algorithm produced very precise calculations of the robot position on the benchmark field. The maximum error was 16.2 mm in an experiment, and the average value of 25 measurements was 7.2 mm. These results are considered very satisfying and superior when compared to the results of the 24 teams that participated in the 2004 RoboCup championship.

The position of the ball is computed in a similar approach. After its outlines have been detected, its center and radius in the video image is computed with an approach similar to the robot self localization. The actual ball position in robot coordinates is obtained by mapping and scaling the coordinates of the ball center in the video images using the calibration lookup table. The ball position was computed only for creating the diagrams resulting from the dribbling experiments, the robot does not react to the ball yet. The error in the ball position grows with increasing distance to the robot. Therefore, the position error is provided relative to the distance between ball and robot. In an experiment with 29 different ball positions around the robot, the maximum distance was about 2.3 m. The smallest relative error of 0.2 % occurred at the closest measurement and the largest error of 8.0 % at the most remote measurement. The average value of all measurements was 4.2 % which is considered very good.

The following advantages are seen with the video image processing algorithms that have been developed: They use standard hardware components and only one type of sensor. The three synchronized cameras provide omnidirectional vision which is very useful because the number of marking lines on a soccer field is limited and sometimes a robot is oriented towards a direction that offers insufficient information for self localization if only one camera was looking in forward direction. It is relatively simple to implement, and it allows for a real time processing of 25 half frames per second. Its flexibility allows to easily vary the geometry of the field marking lines. Due to the statistical approach, the computed solution is very stable and subsequent calculated positions do not deviate much. The detection of the precise center

of a partially hidden ball is possible due to the color gradient and outline method. The major disadvantage of the robot self localization is the fact that it is based on an iterative search and therefore requires a good initial value. In addition to that, there is no measure for the quality of the obtained solution. The average distance between marking outline pixels to the respective marking line does not help because it provides no information on the correct associativity. The method is only useful for detecting floor level markings in two dimensions. If the environment is not perfectly flat, the mapping method fails. For the RoboCup environment, the method is fully appropriate and satisfying. The video image recognition algorithm fulfills all requirements that were defined in the introduction. It runs at the specified frame rate, and it meets the target values of the resulting precisions for the robot and the ball.

Controller

The motor controllers were developed by the supplier, it was only possible to optimize their performance by tuning the internal controller gains. The internal ramping function of the DES for the set speed turned out extremely useful. Compared to a previous implementation that used a constant and very high slope of the ramp, it could be shown that the robot stayed much more accurately on its predetermined course when the speed ramping function was active and the slope of the ramp was updated dynamically with the set speed.

The robot motion controller was implemented with a feed forward compensation that sends the calculated robot velocity from the drivecycle directly to the motors, and a position feedback controller that adds a controller velocity. The position feedback controller was deliberately designed as simple as possible, because it was a primary focus to optimize the open loop behavior of the robot to render a sophisticated control strategy unnecessary. The robot position controller was implemented as a mere proportional controller, its output velocity is proportional to the robot's course deviation in each coordinate. A series of experiments was conducted to investigate the behavior of the robot motion controller, and it could be shown that the maximum controller velocity relative to the maximum calculated drivecycle velocity was about 10%. That means that the proportional controller does not need to correct very much course deviation. The robot stayed on its trajectory very accurately and therefore, the approach with a simple proportional gain is fully justified.

However, the robot position estimation by extrapolating the previous results of the video image recognition can be improved because it does not

predict the curvature of the trajectory very accurately. This is because it was designed to solely rely on the information produced with the video sensors. One approach to improve the accuracy of the estimation is to take the robot velocity into account which can be obtained from the drivecycle or from the wheel speeds measured with the DES.

Perspective

The overall performance of the developed algorithms and hardware is very satisfying. It can be stated that the robot developed in the course of this research project is equipped with the two basic skills identified in the beginning: The capability of acquiring a detailed view of the surrounding world and the agility to handle a rolling soccer ball and maneuver it to a desired position. The following steps are identified to approach the ultimate goal of participating in a RoboCup competition: First, the robot chassis needs to be redesigned to meet the geometrical restrictions imposed by the RoboCup rule set. The development of an adequate ball handling mechanism is a challenging issue that also relates to modifications of the robot hardware. Second, the trajectory planning algorithm must be implemented on board the robot, such that the robot can compute its course by itself. Third, a few basic supplemental algorithms need to be developed, e.g. to determine the fastest way to reach a certain point, or to search the soccer field systematically for the ball if it can not be seen anywhere. It is believed by the author that even in the 2004 RoboCup competition, it would have been possible to win most matches with a single robot that can master the two basic skills but has no artificial intelligence except for the aforementioned basic algorithms. But finally, it is necessary to build a number of robots to meet the minimum player requirement.

Appendix A

Robot Implementation

A.1 Technical Data Components

Component	Supplier	Type
battery cells	Sanyo	Ni-Cd 2400 mAh
motor controllers	maxon	DES 70/10 (228597)
motors	maxon	EC 45 (136208)
choke coils	maxon	Drosselmodul (232359)
gearbox	maxon	GP 42 C (203115)
DC/DC converter	PEAK	custom
video cameras	Phytec	VCAM-003 (AK035)
frame grabbers	Phytec	eGrabber-2 (EPC-901)
PC/104 main board	E.E.P.D.	Profive CPU-T5V
CAN board	Phytec	eNET-CAN (EPC-011-2)
PCMCIA board	Phytec	eCard-1 (EPC-010)
multi I/O board	PEAK	custom
wireless CAN	Kvaser	WaveCAN

Table A.1: Robot components and suppliers

Battery

type	Ni-Cd, rechargeable
number of cells	36
charge	2.4 Ah
cell voltage	1.2 – 1.4 V
voltage	43.2 – 50.4 V
internal resistance	0.618 Ω
number per robot	1

DES motor controllers

supply voltage	24 – 70 V
maximum continuous current	10 A
maximum peak current	30 A
frequency for final drive	50 kHz
frequency for current controller	8 kHz
frequency for speed controller	1 kHz
maximum motor speed	25000 rpm
minimum required inductivity	0.400 mH
communication interfaces	analog, CAN, RS232 serial
maximum efficiency	92 %
maximum effective output voltage	90 % of supply voltage
number per robot	3

Motors & choke coils

type	DC brushless
commutation	electronic
rotor magnet	neodymium permanent
winding connection	3 phase Y
ohmic resistance phase to phase	0.64 Ω
inductivity phase to phase	0.260 mH
inductivity extra choke coils	0.150 mH
max. continuous torque at 5000 rpm	0.3 Nm
max. continuous current at 5000 rpm	6 A
maximum speed at 36 V	6000 rpm
no-load current	0.370 A
torque constant k_M	0.054 $\frac{\text{Nm}}{\text{A}}$
speed constant k_N	175 $\frac{\text{rpm}}{\text{V}}$
slope of the motor characteristic	2100 $\frac{\text{rpm}}{\text{Nm}}$
maximum efficiency	85 %
number per robot	3

Gearbox	
number of stages	2
gear ratio i_{gear}	$\frac{49}{4} = 12.25$
maximum radial force output shaft	150 N
maximum efficiency	81 %
number per robot	3
DC/DC converter	
supply voltage	21 – 56 V
5 V output power	50 W
12 V output power	72 W
number per robot	1
Video cameras	
lens focus	2.3 mm fixed
supply voltage	12 V
video sensor	1/3 inch CCD
video standard	PAL
resolution	752 * 582 pixels
frame rate	25 fps
number per robot	3
Frame grabbers	
resolution	720 * 576 pixels
processor	BT 848
interface to mainboard	PCI 2.1 bus
number per robot	3
PC/104 mainboard	
CPU	Pentium MMX 266 MHz
RAM	64 MB
operating system	MS-DOS
programming language	C and C ⁺⁺
number per robot	1
Peripheral electronic (per robot)	
number of CAN boards	1
number of PCMCIA boards	1
number of multi I/O boards	1
number of wireless CAN devices	1

Table A.2: Robot components technical data

A.2 Camera Calibration Program

This section describes the functionalities of the software that was developed to perform the calibration of the video position sensors described in section 6.2.1. The software runs on the robot platform because it is extremely hardware oriented. It is programmed in C and C⁺⁺ and it shares some sub-routines with the video image processing routine. The `djgpp` development system [53] was used for programming both the camera calibration software and the robot control program. The calibration software offers a graphical user interface with three different screens serving the various functionalities.

Screen 1: Video Image

The first step in the calibration procedure is capturing a video image from the current robot position. Figure A.1 shows screen 1 with the video image of a camera. Note that it is not necessary to reduce the resolution of the video images for the calibration process because only one camera at a time is calibrated and this is not a real time application, so time is not crucial. Both half images are utilized at a resolution of 720*576 pixels which is possible because the robot is standing still and the delay of 20 ms in the exposure of the two half images does not matter. The video image shows the calibration pattern lying on top of the field of play in front of the robot. The field of play with its marker tape can be seen sticking out on both sides of the calibration carpet, it is not needed for the calibration process. In the far left corner of the room there is a glass door through which sunlight falls into the laboratory and reflects on the calibration carpet. Note that all experiments described in chapter 6 were conducted in a different lab with daylight only entering from the ceiling. The toilet paper packages on the left and on the far side were used as bumpers to protect the robot during driving tests in case it started driving in an unexpected direction, they are irrelevant for calibration.

The calibration carpet is painted with black and white squares, the outlines of these squares are detected as outline pixels by the software. In figure A.1, these outline pixels are plotted in red color, not visible in a grayscale printout. The outline pixels are determined according to section 5.1.3: The criterion is a minimum difference in the luminance value (Y) between two neighboring pixels. This difference can be adjusted manually with the contrast selector in the top right corner of screen 1, the new outline pixels are computed instantaneously to give the user a feedback. Another important feature in this screen is the base grid constituted by a number of horizontal and vertical lines. The intersect points of two lines in this grid are the

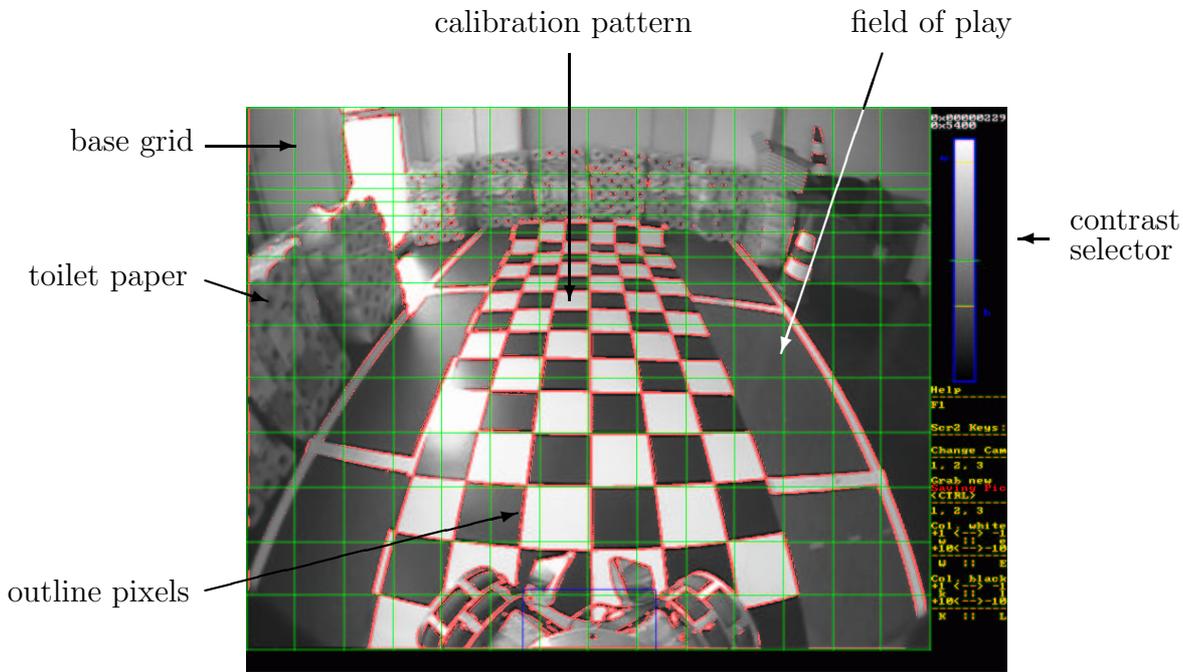


Figure A.1: Camera calibration program: Screen 1, video image

nodes that store the ξ - and η -coordinates for the respective pixel in the video bitmap. The relative location of these nodes is stored as the first information in the camera calibration file. That means, their location is saved in percent of the total width and height of the video image to make the camera calibration independent of the actual video image resolution of the robot control software. Note that the spacing of the horizontal lines is not equidistant. The spacing in the video image becomes smaller with increasing distance of the objects from the robot. This accounts for the perspective effect and allows an increased accuracy in that region. The topmost spacing is very large because all information shown in that band is far beyond the calibration horizon and therefore irrelevant. Note also that this camera can see almost two complete wheels. Due to the robot symmetry, all cameras can see their respective two wheels which gives an impression of the size of the overlapping area of two cameras.

Screen 2: ξ - η Coordinates

This screen displays the real life ξ - η coordinates of all outline pixels detected with the settings from screen 1. These outline pixels are plotted white on

a black background in figure A.2. There are pixels outlining the squares of the calibration carpet ("outline pixels"). They are required for the further calibration process. There are pixels outlining both sides of the marker tape glued on the field of play that is sticking out on both sides. They are irrelevant. And there are outline pixels reflecting what the robot can see of itself. They are located in the bottom center of this screen. The pixels in the top left and top right corners of this view belong to the surroundings in the laboratory and can be ignored. The ξ -axis indicating the robot forward direction is oriented from bottom to top in screen 2, the η -axis is located at the bottom and oriented left. This screen can be interpreted as a top view of the robot.

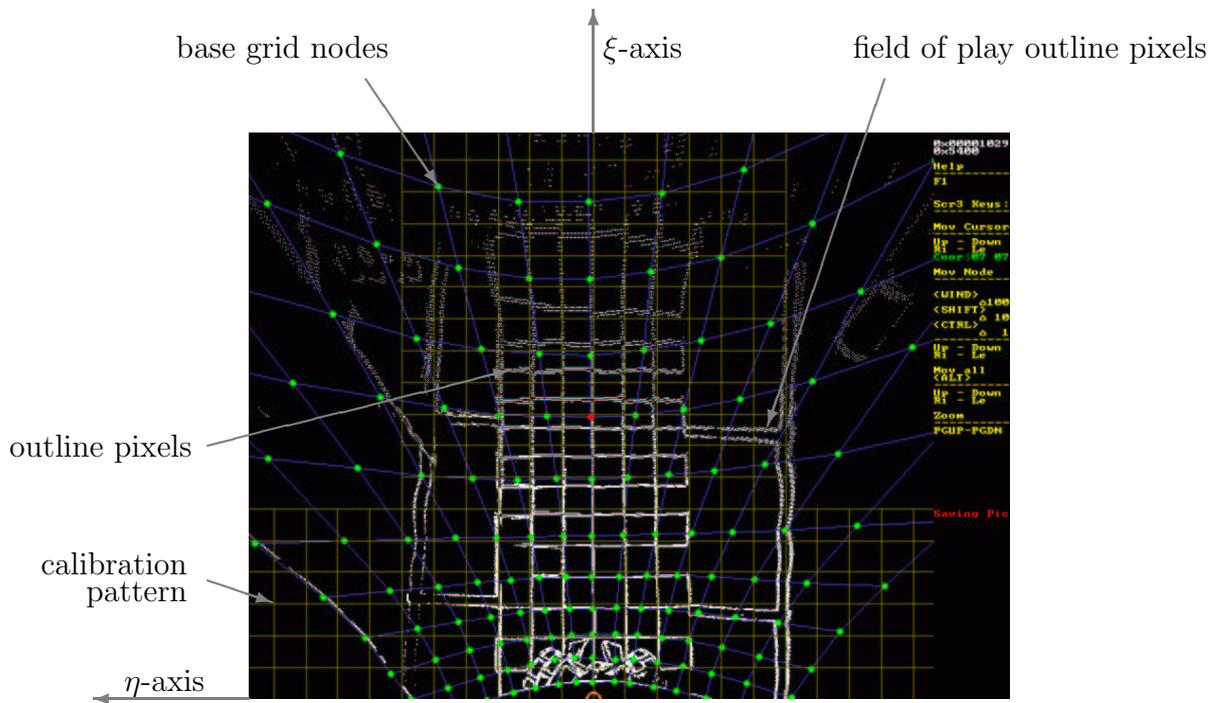


Figure A.2: Camera calibration program: Screen 2, ξ - η coordinates

There is a number of horizontal and vertical lines visible in screen 2, they represent the calibration pattern. Their coordinates are stored in a configuration file that is read by the calibration program at startup. Their spacing equals the size of the squares of the calibration pattern. In figure A.2, they cover an area of an upside down letter 'T' because they only need to cover all possible configurations of the robot standing on the calibration pattern. For a wider calibration carpet, new lines must be added.

The last objects in screen 2 are the base grid nodes and the straight lines connecting two adjacent nodes. Remember that each grid node from the video image is associated with a real life coordinate (ξ, η) . The actual task of calibrating a camera is to determine the best coordinate (ξ, η) for each grid node. This is achieved by the person performing the calibration by picking a node and moving it in ξ - and η -directions. The coordinates of all outline pixels located in the four quadrants around a respective base grid node are effected by the node's coordinates and therefore, the outline pixels will move with the grid node. It is the goal of the calibration procedure to match all relevant outline pixels with their respective calibration pattern line by adjusting the coordinates (ξ, η) of all base grid nodes.

The coordinates (ξ, η) of every pixel of the video image displayed in screen 1 are calculated with a bilinear transformation of the coordinates (ξ, η) of the four surrounding base grid nodes. Every time a base grid node is moved, the coordinates (ξ, η) of all its dependent outline pixels are recomputed so the operator gets an immediate feedback. The complete calibration process has to be carried out manually. A semi-automated calibration is conceivable for a fine tuning of the node coordinates but has not been developed. The most difficult problem to solve for a fully automated calibration procedure is the pixel-calibration line associativity. The computer needs to know which line a pixel belongs to. This problem was solved for the real time position calculation described in section 5.2 by providing a sufficiently accurate initial guess of the robot position and by the boundaries around each line. The calibration software allows zooming and panning of the objects in screen 2 for optimal accuracy. The resulting base grid node coordinates are saved as the second piece of information in the camera calibration file.

Each camera is calibrated separately by the process described above. There is no absolute reference for the grid nodes. If the robot is not aligned perfectly with the calibration pattern, this can be adjusted with shifted base node coordinates. One general guideline for the position of the base nodes is therefore employed. It should always be tried to locate the central column of nodes with the ξ -axis. The initial robot setup position is chosen such that this condition is fulfilled without moving the central nodes in η -direction. Figure A.2 shows an advanced calibration process that is not complete yet. The outline pixels in the robot's vicinity match very well with the calibration pattern but the pixels at a greater distance are too close to the robot, so the respective base grid nodes need to be moved in positive ξ -direction. Note the U-shape of the lines connecting the base grid nodes in horizontal direction. In the upper half of figure A.2, they are U-shaped upwards while in the lower half they are U-shaped downwards. The point of inflection marks the optical axis of the camera lens.

Screen 3: Camera Conspectus

The final step in calibrating the cameras is aligning the angles of the three cameras with respect to each other. Screen 3 in figure A.3 shows the outline pixels of all three cameras in ξ - η coordinates at the same time. The outline pixels of camera 1 are plotted in red color, the outline pixels of cameras 2 and 3 are plotted in white color so they can not be distinguished in this figure. The calibration pattern and the field of play sticking out can be recognized clearly. Outside of these carpets there is a large number of pixels that must be considered as noise produced by laboratory walls and other objects standing around with changing brightness. The large circle that fills almost the complete screen indicates the calibration horizon. It can be adjusted by the person conducting the calibration. Video image pixels whose coordinates are located inside this circle will be considered as calibrated by the robot control software. Pixels which map outside will be ignored. With the help of this screen, the forward direction of camera 1 is defined. If any camera is not mounted at the precise horizontal angle of a multiple of 120° , it can be adjusted here. Note that camera 1 is still not calibrated properly and there are some more things that look odd in this figure.

The three camera angles together with the calibration horizon radius are

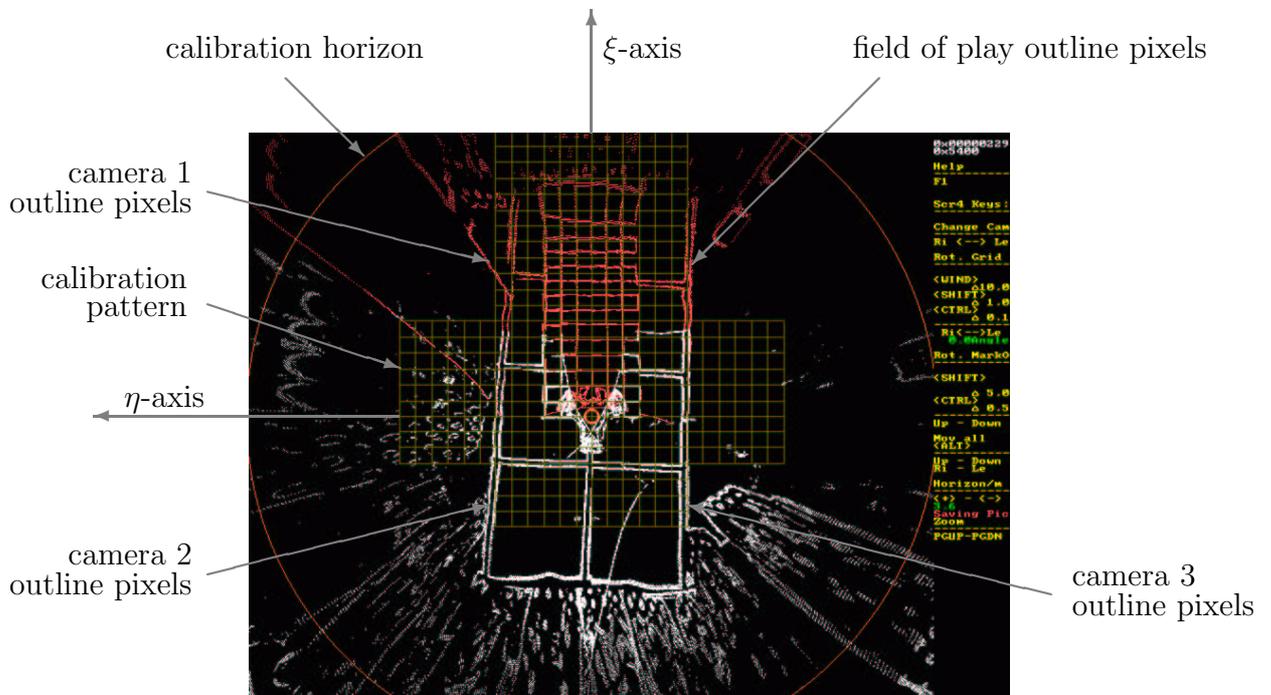


Figure A.3: Camera calibration program: Screen 3, camera conspectus

the third and last piece of information that is saved in the camera calibration file. This file is loaded by the robot control software at startup. A subroutine creates a huge lookup table that stores the coordinates (ξ, η) for each individual pixel in the video image of each of the three cameras. This totals to $2 * 3 * 92 * 144 = 79488$ table entries for the coordinates of all pixels (2 dimensions, 3 cameras, 92 columns, 144 rows). In addition to that, more tables are initialized containing the powers of the coordinates needed to compute the coefficients listed in table B.2, p. 208.

A.3 Robot Control Program

This section describes the graphical output of the robot control program that is responsible for the video image recognition, the control of the propulsion motors, and the radio communication with the user terminal in real time. The robot control program provides two screens that were mainly used for debugging during the software development phase, and for adjusting operation parameters. They are also very useful to explain how the video image processing algorithm works. At this time, these screens are the only possibility to display detailed inside information of the video image recognition process because the bandwidth of the wireless CAN communication is by far not sufficient to handle the associated large bundles of information. A monitor needs to be connected to the robots's mainboard to display the graphical output of these screens. When the robot is driving it is awkward and dangerous to maintain a wired connection between robot and monitor. Therefore, this was never done. On the other hand, when the robot is driving, it produces huge amounts of video information in real time that a human being can not process at that rate anyway, so there is no actual need to monitor the screen output during driving. It is necessary to review the screens every time the robot enters a new environment in order to adjust the relevant parameters like the coordinates of the markings on the floor or the limits used for color detection to account for different lighting conditions.

Screen 1: Video Image Processing

Screen 1 reports the results of all steps related to the video image recognition as presented in section 5.1. Figure A.4 shows a screenshot of this screen while the robot is standing on the field of play. The screen is divided into a number of fields:

In the left half of the screen, an array of nine video images can be seen.

Each column represents one of the three cameras, camera 1 is in the middle because it is facing forward, cameras 2 and 3 are on the sides because they are facing rear left and rear right, respectively. The first row shows the even half frames of the original video images as captured by the three frame grabbers in RGB format with 24 bit color depth. The resolution was reduced to 92×144 pixels such that the CPU is able to process the amount of video data in real time. Recall that even half frames are used for visualization in RGB format while odd half frames are used for color detection in YUV color format. The ball can be seen in camera 1 lying in front of the robot on the intersect of two marking lines. The second row of video images shows the result of the color recognition as described in section 5.1.2. Every pixel in this row has one of the six colors defined by RoboCup or is purple for an undefined color. The last row shows the original video images in dimmed grayscale and adds the result of the outline detection according to section 5.1.3. Pixels outlining the white marker tape on the green field are plotted green in this row, pixels outlining the red ball in front of white tape, green floor, or blue or yellow goals are plotted in red color. Note that only these four neighboring colors are accepted because they unite to form a circular ball outline which is required to compute the ball position according to section 5.3. Red pixels with black neighbors are not accepted as outline pixels because they might indicate a second robot standing in front of the ball and covering it partially. These pixels would not contribute to a circular appearance of the ball outline and are therefore ignored. In the outline image of camera 1, a box can be seen around the ball. This box is used to constrain the ball outline pixels to a small area in the video image, because the method to calculate the bitmap coordinates of the ball center presented in section 5.3.1 is sensitive to pixels that do not actually belong to the red ball. For example, if there are two balls in the robot's view, they are distinguished by an algorithm that is not explained here, and each ball will be isolated within a separate box. Then the positions of both balls will be computed and the robot checks which of the balls is closer to the expected ball position.

The second field in screen 1 in the top right corner shows the U-V color space that is used to classify pixels by their color, as described in section 5.1.2. This field is the bridge from row 1 to row 2 in the video image array on the left side of screen 1. All pixels from the original video images of all three cameras are plotted in this field. Their position is determined with their U and V chrominance values that are provided by the grabbers for the odd half images. Their color is determined with their RGB value which is provided by the grabbers for the even half images. The position and color information of each pixel in this field therefore does not match exactly because it is derived from both half images. That means, the position information is shifted by

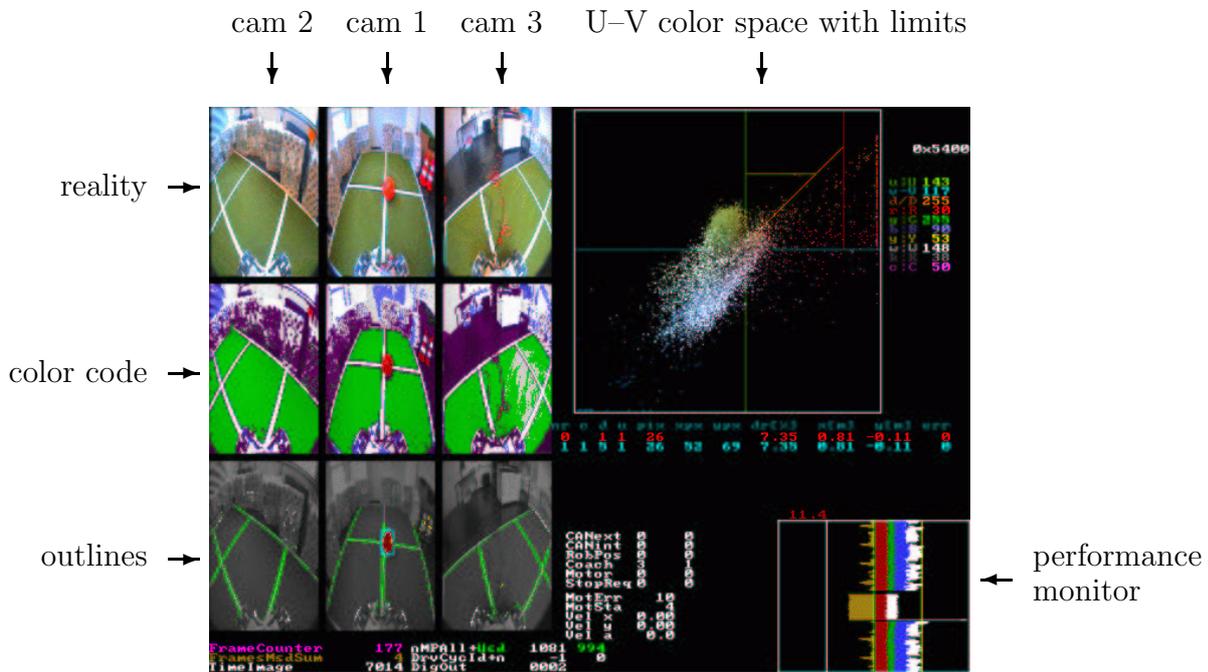


Figure A.4: Robot control program: Screen 1, video images

one line of the video camera compared to the color information, and their exposure times are shifted by 20 ms. Remember that the RGB color space representation is only used for visualization while the robot is standing still, so these two imperfections do not hurt. For the actual computation of the robot and the ball position, only the odd half images are used which is consistent. There are also two horizontal, a vertical, and a diagonal line in this field dividing the U-V color space in separate regions, each region represents one of the RoboCup colors. Note that the RoboCup colors black and white are identified by their brightness value Y which is not represented in this color space. That is why there are dark and bright pixels located next to each other in this diagram, some pixels are even covered with other pixels having the same U- and V-value but different brightness. The location of the dividing lines can be adjusted with a number of parameters, their values are printed on the right hand side of the U-V color space. Note that the U-V color space in screen 1 is mirrored about the U-axis compared to figure 5.5, p. 85. In screen 1, blue is in the bottom left quadrant while green is in the top left quadrant.

In the bottom right corner of screen 1, the performance monitor is located. It is a visual representation of the CPU time consumed for each of the video image recognition steps. The processing of each odd video half image

is represented by a horizontal line in the performance monitor, the length of these lines is proportional to the consumed CPU time. Each horizontal line is divided in five sections with different colors representing the different steps. These are from left to right: Brown color for waiting, red color for color detection, green color for outline detection, blue color for computation of robot and ball position, and white color for updating the graphical output in this screen. The first section in brown color that is ragged on its left side represents the time the CPU waits in idle mode for a new video image to arrive from the grabbers. Then the actual video image recognition process starts, the red sections of all lines are therefore left aligned. The monitor views 128 lines, it updates them from top to bottom, then it starts at the top again overwriting the previous lines. It also erases the 5 upcoming lines to make it easier to detect the current position. All lines of the monitor can therefore display the last 5.12s. In this screenshot, the robot control software started about 7s ago, the current line in the performance monitor is in the middle and the next lines that are about to be overwritten were produced during the initialization phase. At startup, the robot control program waits for 80 frames to allow the frame grabbers to adjust their internal parameters for the automatic gain control. During that time, only the following actions are performed: Wait for next image, check colors, update screen. The performance monitor shows a relict of that initialization phase. The vertical lines in the monitor are spaced at the equivalent of 40 ms. One can easily see that every time a VIR cycle was finished faster than that, the CPU had to wait in the next cycle for the unused portion of time. Note that the screen output consumes a significant portion of the total CPU time in each cycle.

Screen 2: ξ - η Coordinates Processing

The second screen visualizes the ξ - η coordinates of the recognized field marking outline pixels from all cameras. Figure A.5 shows this for a different robot setup. This screen is much like screen 2 in the camera calibration program, only that here are much less false outline pixels. This is because the robot control program identifies outline pixels by color while the calibration program identifies them by brightness which is less strict. The calibration horizon is plotted as a large circle and the area in which the ball is expected is plotted as a small circle. The field marking objects defined in section 5.2 are plotted together with their boundaries that determine whether an outline pixel is associated with this line or not. The coordinates of these marking object lines are mapped from global to robot coordinates based on the solution of equation (5.19), p. 91, which yields the position of the field origin in

robot coordinates.

In the top right corner, the numeric values of the robot and ball position are printed together with the number of outline pixels that were used. The bottom right corner shows the performance monitor again. This time the CPU is very busy: Every image requires about 40 ms to be processed, there is almost no waiting time. If a cycle takes longer than 40 ms, it is still possible to process the next one without waiting time. An intelligent management system decides when it is too late to start detecting the colors of a video image, because it must be avoided that an image is overwritten with the next one during color detection. It is therefore possible that one complete image has to be dropped if the preceding cycles accumulated sufficient delay. In that case, the wait time for the next image is about 20 ms. This happened two times, at the top and in the middle of the performance monitor in figure A.5.

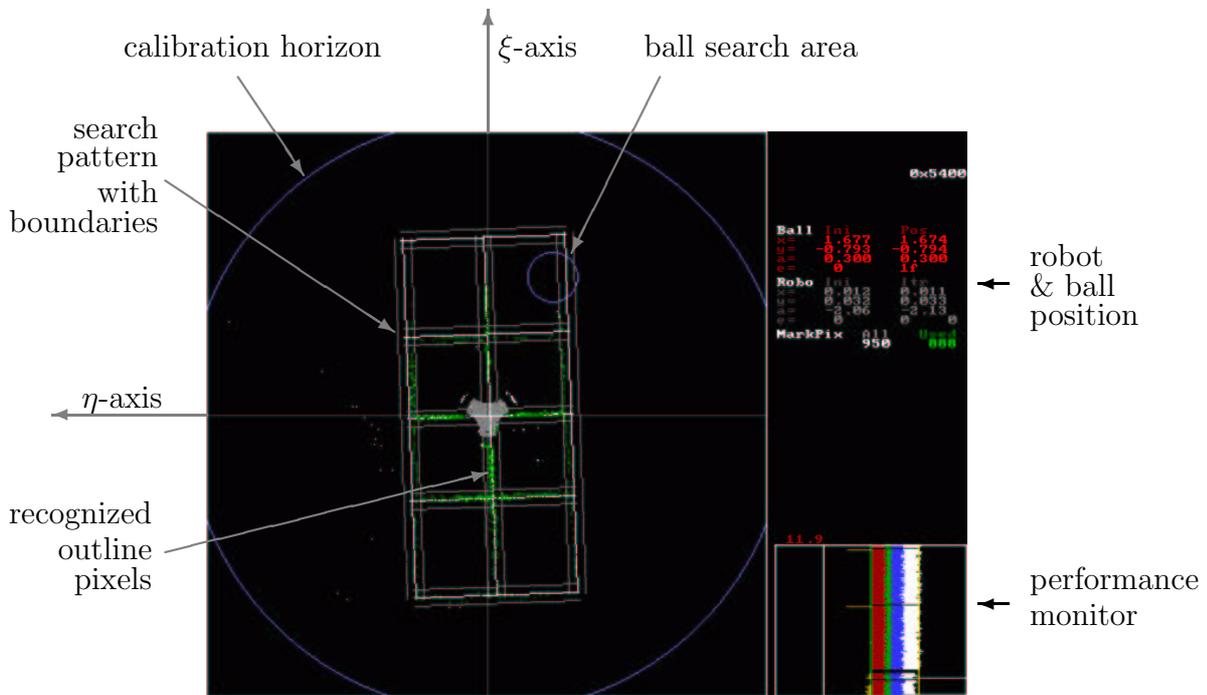


Figure A.5: Robot control program: Screen 2, ξ - η coordinates

Appendix B

Numerical Solutions

B.1 MCL Method: Trajectory Shape

This section describes how to compute the derivatives of the coefficient vector c_k and the contributions s^* and κ^* to the objective function z^* with respect to r_p . The parameterization of the trajectory coordinates x and y according to equations (3.21) and (3.22), p. 47, yields the following derivatives with respect to r :

$$x(r) = \sum_{k=0}^{n_c-1} c_k \cdot r^k \quad (\text{B.1})$$

$$y(r) = \sum_{k=0}^{n_c-1} c_{n_c+k} \cdot r^k \quad (\text{B.2})$$

$$x'(r) = \sum_{k=1}^{n_c-1} k c_k \cdot r^{k-1} \quad (\text{B.3})$$

$$y'(r) = \sum_{k=1}^{n_c-1} k c_{n_c+k} \cdot r^{k-1} \quad (\text{B.4})$$

$$x''(r) = \sum_{k=2}^{n_c-1} (k-1) k c_k \cdot r^{k-2} \quad (\text{B.5})$$

$$y''(r) = \sum_{k=2}^{n_c-1} (k-1) k c_{n_c+k} \cdot r^{k-2} \quad (\text{B.6})$$

Let z_i be a column vector containing the coordinates of the control points x_i , y_i , or both. This generalization is useful when start or end conditions on the trajectory's direction or curvature are required. Let R_{ik} be the matrix containing the powers of the entries of the solution vector r_p for the trajectory

shape. It is square and has as many rows as z_i :

$$R_{ik} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & r_1 & r_1^2 & r_1^3 & \dots & r_1^{n_c-1} \\ 1 & r_2 & r_2^2 & r_2^3 & \dots & r_2^{n_c-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & r_{n_p-2} & r_{n_p-2}^2 & r_{n_p-2}^3 & \dots & r_{n_p-2}^{n_c-1} \\ 1 & 1 & 1 & 1 & \dots & 1 \end{bmatrix} \quad (\text{B.7})$$

The first and second derivatives of c_k with respect to r_p are obtained with the following equations:

$$c_k = (R_{ik})^{-1} \cdot z_i = R_{ki}^{-1} \cdot z_i \quad (\text{B.8})$$

$$\frac{dc_k}{dr_p} = \frac{dR_{ki}^{-1}}{dr_p} \cdot z_i \quad (\text{B.9})$$

$$\frac{d^2c_k}{dr_p dr_q} = \frac{d^2R_{ki}^{-1}}{dr_p dr_q} \cdot z_i \quad (\text{B.10})$$

The derivatives of R_{ki}^{-1} with respect to r_p are obtained with the following equation in which δ_{ji} is the Kronecker delta. Its entries are unity for $j=i$ and zero otherwise.

$$R_{jk} \cdot R_{ki}^{-1} = \delta_{ji} \quad (\text{B.11})$$

$$\frac{dR_{jl}}{dr_p} \cdot R_{li}^{-1} + R_{jk} \cdot \frac{dR_{ki}^{-1}}{dr_p} = 0 \quad (\text{B.12})$$

$$\frac{d^2R_{jn}}{dr_p dr_q} \cdot R_{ni}^{-1} + \frac{dR_{jm}}{dr_p} \cdot \frac{dR_{mi}^{-1}}{dr_q} + \frac{dR_{jl}}{dr_q} \cdot \frac{dR_{li}^{-1}}{dr_p} + R_{jk} \cdot \frac{d^2R_{ki}^{-1}}{dr_p dr_q} = 0 \quad (\text{B.13})$$

These equations can be solved for the derivatives of R_{ki}^{-1} with respect to r_p :

$$\frac{dR_{ki}^{-1}}{dr_p} = -R_{kj}^{-1} \cdot \frac{dR_{jl}}{dr_p} \cdot R_{li}^{-1} \quad (\text{B.14})$$

$$\frac{d^2R_{ki}^{-1}}{dr_p dr_q} = -R_{kj}^{-1} \cdot \left[\frac{dR_{jm}}{dr_p} \cdot \frac{dR_{mi}^{-1}}{dr_q} + \frac{dR_{jl}}{dr_q} \cdot \frac{dR_{li}^{-1}}{dr_p} + \frac{d^2R_{jn}}{dr_p dr_q} \cdot R_{ni}^{-1} \right] \quad (\text{B.15})$$

Note that $\frac{d^2R_{jn}}{dr_p dr_q} = 0$ for $p \neq q$ because R_{ik} does not contain mixed entries of

r_p and r_q . Note also that $\frac{d^2R_{ki}^{-1}}{dr_p dr_q}$ is symmetrical with respect to p and q .

The curve length that is required for the objective function z^* and its derivatives are computed by

$$s^* = \int_0^1 s'(r) dr \quad (\text{B.16})$$

$$\frac{ds^*}{dr_p} = \int_0^1 \frac{ds'}{dr_p} dr \quad (\text{B.17})$$

$$\frac{d^2s^*}{dr_p dr_q} = \int_0^1 \frac{d^2s'}{dr_p dr_q} dr \quad (\text{B.18})$$

with the integrands

$$s' = \sqrt{(x')^2 + (y')^2} \quad (\text{B.19})$$

$$\frac{ds'}{dr_p} = \frac{1}{s'} \left[x' \frac{dx'}{dr_p} + y' \frac{dy'}{dr_p} \right] \quad (\text{B.20})$$

$$\frac{d^2s'}{dr_p dr_q} = \frac{1}{s'} \left[\frac{dx'}{dr_p} \frac{dx'}{dr_q} + \frac{dy'}{dr_p} \frac{dy'}{dr_q} - \frac{ds'}{dr_p} \frac{ds'}{dr_q} + x' \frac{d^2x'}{dr_p dr_q} + y' \frac{d^2y'}{dr_p dr_q} \right] \quad (\text{B.21})$$

Note that $\frac{d^2s'}{dr_p dr_q}$ is symmetrical with respect to p and q .

The derivatives of x' and y' with respect to r_p are obtained via the coefficients c_k :

$$\begin{aligned} \frac{dx'}{dr_p} &= \frac{dx'}{dc_k} \frac{dc_k}{dr_p} & \frac{dy'}{dr_p} &= \frac{dy'}{dc_k} \frac{dc_k}{dr_p} \\ \frac{d^2x'}{dr_p dr_q} &= \frac{dx'}{dc_k} \frac{d^2c_k}{dr_p dr_q} & \frac{d^2y'}{dr_p dr_q} &= \frac{dy'}{dc_k} \frac{d^2c_k}{dr_p dr_q} \end{aligned} \quad (\text{B.22})$$

The functions x' and y' are linear functions of the coefficients c_k . Therefore, only the first derivatives exist:

$$\frac{dx'}{dc_k} = \begin{cases} k \cdot r^{k-1} & \text{for } 1 \leq k \leq n_c - 1 \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.23})$$

$$\frac{dy'}{dc_k} = \begin{cases} (k - n_c) \cdot r^{k-n_c-1} & \text{for } n_c + 1 \leq k \leq 2n_c - 1 \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.24})$$

The curvature penalty function κ^* is treated similarly to s^* :

$$\kappa^* = \frac{1}{2} \int_0^1 \kappa^2(r) dr \quad (\text{B.25})$$

$$\frac{d\kappa^*}{dr_p} = \int_0^1 \kappa \frac{d\kappa}{dr_p} dr \quad (\text{B.26})$$

$$\frac{d^2\kappa^*}{dr_p dr_q} = \int_0^1 \left(\kappa \cdot \frac{d^2\kappa}{dr_p dr_q} + \frac{d\kappa}{dr_p} \cdot \frac{d\kappa}{dr_q} \right) dr \quad (\text{B.27})$$

The derivatives of the integrands with respect to r_p are longer expressions than those for s' :

$$\kappa = \frac{x'y'' - x''y'}{((x')^2 + (y')^2)^{\frac{3}{2}}} \quad (\text{B.28})$$

$$\frac{d\kappa}{dr_p} = \frac{d\kappa}{dx'} \frac{dx'}{dr_p} + \frac{d\kappa}{dy'} \frac{dy'}{dr_p} + \frac{d\kappa}{dx''} \frac{dx''}{dr_p} + \frac{d\kappa}{dy''} \frac{dy''}{dr_p} \quad (\text{B.29})$$

$$\begin{aligned} \frac{d^2\kappa}{dr_p dr_q} &= \frac{d\kappa}{dx'} \frac{d^2x'}{dr_p dr_q} + \frac{d\kappa}{dy'} \frac{d^2y'}{dr_p dr_q} + \frac{d\kappa}{dx''} \frac{d^2x''}{dr_p dr_q} + \frac{d\kappa}{dy''} \frac{d^2y''}{dr_p dr_q} \\ &+ \frac{dx'}{dr_p} \left[\frac{d^2\kappa}{dx' dx'} \frac{dx'}{dr_q} + \frac{d^2\kappa}{dx' dy'} \frac{dy'}{dr_q} + \frac{d^2\kappa}{dx' dx''} \frac{dx''}{dr_q} + \frac{d^2\kappa}{dx' dy''} \frac{dy''}{dr_q} \right] \\ &+ \frac{dy'}{dr_p} \left[\frac{d^2\kappa}{dy' dx'} \frac{dx'}{dr_q} + \frac{d^2\kappa}{dy' dy'} \frac{dy'}{dr_q} + \frac{d^2\kappa}{dy' dx''} \frac{dx''}{dr_q} + \frac{d^2\kappa}{dy' dy''} \frac{dy''}{dr_q} \right] \\ &+ \frac{dx''}{dr_p} \left[\frac{d^2\kappa}{dx'' dx'} \frac{dx'}{dr_q} + \frac{d^2\kappa}{dx'' dy'} \frac{dy'}{dr_q} + \frac{d^2\kappa}{dx'' dx''} \frac{dx''}{dr_q} + \frac{d^2\kappa}{dx'' dy''} \frac{dy''}{dr_q} \right] \\ &+ \frac{dy''}{dr_p} \left[\frac{d^2\kappa}{dy'' dx'} \frac{dx'}{dr_q} + \frac{d^2\kappa}{dy'' dy'} \frac{dy'}{dr_q} + \frac{d^2\kappa}{dy'' dx''} \frac{dx''}{dr_q} + \frac{d^2\kappa}{dy'' dy''} \frac{dy''}{dr_q} \right] \end{aligned} \quad (\text{B.30})$$

The derivatives of x' and y' with respect to r_p have already been provided above, new are the derivatives of x'' and y'' :

$$\frac{dx''}{dr_p} = \frac{dx''}{dc_k} \frac{dc_k}{dr_p} \quad \frac{dy''}{dr_p} = \frac{dy''}{dc_k} \frac{dc_k}{dr_p} \quad (\text{B.31})$$

$$\frac{d^2x''}{dr_p dr_q} = \frac{dx''}{dc_k} \frac{d^2c_k}{dr_p dr_q} \quad \frac{d^2y''}{dr_p dr_q} = \frac{dy''}{dc_k} \frac{d^2c_k}{dr_p dr_q}$$

The functions x'' and y'' are also linear functions of the coefficients c_k . Therefore, only the first derivative exists:

$$\frac{dx''}{dc_k} = \begin{cases} k(k-1) \cdot r^{k-2} & \text{for } 2 \leq k \leq n_c - 1 \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.32})$$

$$\frac{dy''}{dc_k} = \begin{cases} (k-n_c)(k-n_c-1) \cdot r^{k-n_c-2} & \text{for } n_c + 2 \leq k \leq 2n_c - 1 \\ 0 & \text{otherwise} \end{cases} \quad (\text{B.33})$$

For the derivatives of κ with respect to x' , y' , x'' , and y'' , let κ be defined as:

$$\kappa = \frac{B}{A^{\frac{3}{2}}} = A^{-\frac{3}{2}} \cdot B \quad (\text{B.34})$$

with

$$A = x'x' + y'y' \quad (\text{B.35})$$

$$B = x'y'' - x''y'. \quad (\text{B.36})$$

The first and second derivatives of κ with respect to x' , y' , x'' , and y'' , can be expressed as

$$\frac{d\kappa}{dx'} = \kappa \left(-3\frac{x'}{A} + \frac{y''}{B} \right) \quad (\text{B.37})$$

$$\frac{d\kappa}{dy'} = \kappa \left(-3\frac{y'}{A} - \frac{x''}{B} \right) \quad (\text{B.38})$$

$$\frac{d\kappa}{dx''} = \kappa \left(-\frac{y'}{B} \right) \quad (\text{B.39})$$

$$\frac{d\kappa}{dy''} = \kappa \left(\frac{x'}{B} \right) \quad (\text{B.40})$$

$$\frac{d^2\kappa}{dx'dx'} = \kappa \left(15\frac{(x')^2}{A^2} - 6\frac{x'y''}{AB} - \frac{3}{A} \right) \quad (\text{B.41})$$

$$\frac{d^2\kappa}{dx'dy'} = \kappa \left(15\frac{x'y'}{A^2} + 3\frac{x'x'' - y'y''}{AB} \right) \quad (\text{B.42})$$

$$\frac{d^2\kappa}{dx'dx''} = \kappa \left(3\frac{x'y'}{AB} \right) \quad (\text{B.43})$$

$$\frac{d^2\kappa}{dx'dy''} = \kappa \left(-3\frac{(x')^2}{AB} + \frac{1}{B} \right) \quad (\text{B.44})$$

$$\frac{d^2\kappa}{dy'dx'} = \kappa \left(15\frac{x'y'}{A^2} + 3\frac{x'x'' - y'y''}{AB} \right) \quad (\text{B.45})$$

$$\frac{d^2\kappa}{dy'dy'} = \kappa \left(15\frac{(y')^2}{A^2} + 6\frac{x''y'}{AB} - \frac{3}{A} \right) \quad (\text{B.46})$$

$$\frac{d^2\kappa}{dy'dx''} = \kappa \left(3\frac{(y')^2}{AB} - \frac{1}{B} \right) \quad (\text{B.47})$$

$$\frac{d^2\kappa}{dy'dy''} = \kappa \left(-3\frac{x'y'}{AB} \right) \quad (\text{B.48})$$

$$\frac{d^2\kappa}{dx''dx'} = \kappa \left(3 \frac{x'y'}{AB} \right) \quad (\text{B.49})$$

$$\frac{d^2\kappa}{dx''dy'} = \kappa \left(3 \frac{(y')^2}{AB} - \frac{1}{B} \right) \quad (\text{B.50})$$

$$\frac{d^2\kappa}{dx''dx''} = 0 \quad (\text{B.51})$$

$$\frac{d^2\kappa}{dx''dy''} = 0 \quad (\text{B.52})$$

$$\frac{d^2\kappa}{dy''dx'} = \kappa \left(-3 \frac{(x')^2}{AB} + \frac{1}{B} \right) \quad (\text{B.53})$$

$$\frac{d^2\kappa}{dy''dy'} = \kappa \left(-3 \frac{x'y'}{AB} \right) \quad (\text{B.54})$$

$$\frac{d^2\kappa}{dy''dx''} = 0 \quad (\text{B.55})$$

$$\frac{d^2\kappa}{dy''dy''} = 0 \quad (\text{B.56})$$

B.2 MCL Method: Trajectory Speed

The derivatives of x and y with respect to time can be computed as follows:

$$\frac{dx}{dt} = \frac{dx}{dr} \frac{dr}{dt} \quad (\text{B.57})$$

$$\frac{d^2x}{dt^2} = \frac{d^2x}{dr^2} \left(\frac{dr}{dt} \right)^2 + \frac{dx}{dr} \frac{d^2r}{dt^2} \quad (\text{B.58})$$

$$\frac{d^3x}{dt^3} = \frac{d^3x}{dr^3} \left(\frac{dr}{dt} \right)^3 + 3 \frac{d^2x}{dr^2} \frac{dr}{dt} \frac{d^2r}{dt^2} + \frac{dx}{dr} \frac{d^3r}{dt^3} \quad (\text{B.59})$$

$$\begin{aligned} \frac{d^4x}{dt^4} &= \frac{d^4x}{dr^4} \left(\frac{dr}{dt} \right)^4 + 6 \frac{d^3x}{dr^3} \left(\frac{dr}{dt} \right)^2 \frac{d^2r}{dt^2} + 3 \frac{d^2x}{dr^2} \left(\frac{d^2r}{dt^2} \right)^2 \\ &\quad + 4 \frac{d^2x}{dr^2} \frac{dr}{dt} \frac{d^3r}{dt^3} + \frac{dx}{dr} \frac{d^4r}{dt^4} \end{aligned} \quad (\text{B.60})$$

The derivatives of r with respect to t are obtained in the same way:

$$\frac{dr}{dt} = \frac{dr}{ds} \frac{ds}{dt} \quad (\text{B.61})$$

$$\frac{d^2r}{dt^2} = \frac{d^2r}{ds^2} \left(\frac{ds}{dt} \right)^2 + \frac{dr}{ds} \frac{d^2s}{dt^2} \quad (\text{B.62})$$

$$\frac{d^3r}{dt^3} = \frac{d^3r}{ds^3} \left(\frac{ds}{dt} \right)^3 + 3 \frac{d^2r}{ds^2} \frac{ds}{dt} \frac{d^2s}{dt^2} + \frac{dr}{ds} \frac{d^3s}{dt^3} \quad (\text{B.63})$$

$$\begin{aligned} \frac{d^4r}{dt^4} &= \frac{d^4r}{ds^4} \left(\frac{ds}{dt} \right)^4 + 6 \frac{d^3r}{ds^3} \left(\frac{ds}{dt} \right)^2 \frac{d^2s}{dt^2} + 3 \frac{d^2r}{ds^2} \left(\frac{d^2s}{dt^2} \right)^2 \\ &\quad + 4 \frac{d^2r}{ds^2} \frac{ds}{dt} \frac{d^3s}{dt^3} + \frac{dr}{ds} \frac{d^4s}{dt^4} \end{aligned} \quad (\text{B.64})$$

The derivatives of r with respect to s are expressed in terms of the inverse function

$$\frac{dr}{ds} \frac{ds}{dr} = 1 \quad (\text{B.65})$$

which can be solved for

$$\frac{dr}{ds} = \left(\frac{ds}{dr} \right)^{-1}. \quad (\text{B.66})$$

Taking the derivative of equation (B.66) with respect to s and utilizing that

$$\frac{d}{ds} \left(\frac{ds}{dr} \right) = \frac{d^2s}{dr^2} \frac{dr}{ds} \quad (\text{B.67})$$

yields the second derivative of r with respect to s

$$\begin{aligned} \frac{d^2r}{ds^2} &= - \left(\frac{ds}{dr} \right)^{-2} \frac{d}{ds} \left(\frac{ds}{dr} \right) \\ &= - \frac{d^2s}{dr^2} \left(\frac{dr}{ds} \right)^3 \end{aligned} \quad (\text{B.68})$$

The third and fourth derivatives are obtained the same way:

$$\frac{d^3r}{ds^3} = - \left[\frac{d^3s}{dr^3} \left(\frac{dr}{ds} \right)^4 + 3 \frac{d^2s}{dr^2} \left(\frac{dr}{ds} \right)^2 \frac{d^2r}{ds^2} \right] \quad (\text{B.69})$$

$$\begin{aligned} \frac{d^4r}{ds^4} &= - \left[\frac{d^4s}{dr^4} \left(\frac{dr}{ds} \right)^5 + 7 \frac{d^3s}{dr^3} \left(\frac{dr}{ds} \right)^3 \frac{d^2r}{ds^2} \right. \\ &\quad \left. + 6 \frac{d^2s}{dr^2} \frac{dr}{ds} \left(\frac{d^2r}{ds^2} \right)^2 + 3 \frac{d^2s}{dr^2} \left(\frac{dr}{ds} \right)^2 \frac{d^3r}{ds^3} \right] \end{aligned} \quad (\text{B.70})$$

The last step is expressing the derivatives of s with respect to r in terms of the parameterized coordinate functions x and y and their derivatives with

respect to r . Starting with equation (3.33), p. 50,

$$\frac{ds}{dr} = \left[\left(\frac{dx}{dr} \right)^2 + \left(\frac{dy}{dr} \right)^2 \right]^{\frac{1}{2}} \quad (\text{B.71})$$

and taking the derivatives with respect to r , yields the following result after some simplifications:

$$\frac{d^2s}{dr^2} = \left(\frac{ds}{dr} \right)^{-1} \left[\frac{dx}{dr} \frac{d^2x}{dr^2} + \frac{dy}{dr} \frac{d^2y}{dr^2} \right] \quad (\text{B.72})$$

$$\frac{d^3s}{dr^3} = \left(\frac{ds}{dr} \right)^{-1} \left[\left(\frac{d^2x}{dr^2} \right)^2 + \frac{dx}{dr} \frac{d^3x}{dr^3} + \left(\frac{d^2y}{dr^2} \right)^2 + \frac{dy}{dr} \frac{d^3y}{dr^3} - \left(\frac{d^2s}{dr^2} \right)^2 \right] \quad (\text{B.73})$$

$$\begin{aligned} \frac{d^4s}{dr^4} = \left(\frac{ds}{dr} \right)^{-1} & \left[3 \left(\frac{d^2x}{dr^2} \frac{d^3x}{dr^3} + \frac{d^2y}{dr^2} \frac{d^3y}{dr^3} - \frac{d^2s}{dr^2} \frac{d^3s}{dr^3} \right) \right. \\ & \left. + \frac{dx}{dr} \frac{d^4x}{dr^4} + \frac{dy}{dr} \frac{d^4y}{dr^4} \right] \quad (\text{B.74}) \end{aligned}$$

For the numerical integration of equations (3.32) and (3.34), p. 50, the interval for the independent variable r is divided into n_i equally sized integration intervals. With an increasing number of control points n_p , the trajectory might gain complexity which leads to larger numerical values of the integrands and a greater error of the numerical integration scheme. Therefore, the number of integration intervals should increase with the number of control points:

$$n_i = 50(n_p - 1) \quad (\text{B.75})$$

The distance between two integration points is

$$\Delta r = \frac{1}{n_i}. \quad (\text{B.76})$$

The value of r for integration point number i is therefore

$$r_i = i \cdot \Delta r \quad , \quad i = 0, \dots, n_i \quad (\text{B.77})$$

with $r_0=0$ and $r_{n_i}=1$.

Starting with $s_0=0$, the distance along the curve in equation (3.32) can be approximated at $r=r_i$ with the following second order approximation scheme:

$$\begin{aligned} s(r_i) \approx s_i & = s_{i-1} + \frac{\Delta r}{6} \left[s'(r_{i-1}) + 4s' \left(\frac{r_{i-1} + r_i}{2} \right) + s'(r_i) \right] \\ & \text{for } i = 1, \dots, n_i \quad (\text{B.78}) \end{aligned}$$

The curvature penalty equation (3.34) and the integrals over the derivatives with respect to r_p and r_q are treated the same way. The value s_i for $i = n_i$ is an approximation for the trajectory curve length s^* :

$$s^* \approx s_{n_i} \quad (\text{B.79})$$

The values of r_i and s_i are stored in a lookup table for later reference. They are needed to compute the values of r at the sample times t_s when the distance along the curve s_s is given according to equation (3.53), p. 56. For a given value s_s , it is required to find the two embracing values s_i and s_{i+1} such that

$$s_i \leq s_s \leq s_{i+1}. \quad (\text{B.80})$$

The corresponding value r_s can then be linearly interpolated with the entries r_i and r_{i+1} in the table that correspond to s_i and s_{i+1} , respectively:

$$r_s = r_i + \frac{s_s - s_i}{s_{i+1} - s_i} (r_{i+1} - r_i) \quad (\text{B.81})$$

B.3 MCL Method: Robot Trajectory

The robot orientation is not determined with equation (3.16), p. 43, because this equation is singular for angles $\alpha = \frac{1}{2}\pi$ and $\alpha = \frac{3}{2}\pi$. Instead, two cases are considered for different sectors of the orientation.

The following forces are introduced:

$$f_x = F_{x,B} \quad (\text{B.82})$$

$$f_y = F_{y,B} \quad (\text{B.83})$$

$$f^2 = f_x^2 + f_y^2 \quad (\text{B.84})$$

with f being the resulting force on the ball acting in direction of α :

$$f_y = f \sin(\alpha) \quad (\text{B.85})$$

$$f_x = f \cos(\alpha) \quad (\text{B.86})$$

Equation (B.85) is used for all cases when $\boxed{|f_y| \leq |f_x|}$. Taking twice the derivative of this equation with respect to time yields

$$\dot{f}_y = \dot{f} \sin(\alpha) + \dot{\alpha} f \cos(\alpha) \quad (\text{B.87})$$

$$\ddot{f}_y = \ddot{f} \sin(\alpha) + 2\dot{\alpha} \dot{f} \cos(\alpha) + \ddot{\alpha} f \cos(\alpha) + \dot{\alpha}^2 f \sin(\alpha) \quad (\text{B.88})$$

which can be solved for $\dot{\alpha}$ and $\ddot{\alpha}$:

$$\alpha = \sin^{-1} \left(\frac{f_y}{f} \right) \quad (\text{B.89})$$

$$\dot{\alpha} = \frac{1}{f_x} [\dot{f}_y - \dot{f} \sin(\alpha)] \quad (\text{B.90})$$

$$\ddot{\alpha} = \frac{1}{f_x} [(\dot{\alpha}^2 f - \ddot{f}) \sin(\alpha) + \ddot{f}_y] - \frac{2}{f} \dot{\alpha} \dot{f} \quad (\text{B.91})$$

Equation (B.86) is used for all cases when $|f_y| > |f_x|$. Taking twice the derivative of this equation with respect to time yields

$$\dot{f}_x = \dot{f} \cos(\alpha) - \dot{\alpha} f \sin(\alpha) \quad (\text{B.92})$$

$$\ddot{f}_x = \ddot{f} \cos(\alpha) - 2 \dot{\alpha} \dot{f} \sin(\alpha) - \ddot{\alpha} f \sin(\alpha) - \dot{\alpha}^2 f \cos(\alpha) \quad (\text{B.93})$$

which can be solved for $\dot{\alpha}$ and $\ddot{\alpha}$:

$$\alpha = \cos^{-1} \left(\frac{f_x}{f} \right) \quad (\text{B.94})$$

$$\dot{\alpha} = \frac{1}{f_y} [-\dot{f}_x + \dot{f} \cos(\alpha)] \quad (\text{B.95})$$

$$\ddot{\alpha} = \frac{1}{f_y} [(\ddot{f} - \dot{\alpha}^2 f) \cos(\alpha) - \ddot{f}_x] - \frac{2}{f} \dot{\alpha} \dot{f} \quad (\text{B.96})$$

The values of \dot{f} and \ddot{f} are obtained by taking the derivative of equation (B.84) with respect to time twice:

$$f \dot{f} = f_x \dot{f}_x + f_y \dot{f}_y \quad (\text{B.97})$$

$$f \ddot{f} + \dot{f}^2 = f_x \ddot{f}_x + \dot{f}_x^2 + f_y \ddot{f}_y + \dot{f}_y^2 \quad (\text{B.98})$$

These equations can be solved for \dot{f} and \ddot{f} :

$$\dot{f} = \frac{1}{f} [f_x \dot{f}_x + f_y \dot{f}_y] \quad (\text{B.99})$$

$$\ddot{f} = \frac{1}{f} [f_x \ddot{f}_x + \dot{f}_x^2 + f_y \ddot{f}_y + \dot{f}_y^2 - \dot{f}^2] \quad (\text{B.100})$$

The derivatives of the vector \vec{x}_0 that indicates the ball neutral position relative to the robot center are:

$$\vec{x}_0 = \xi_0 \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} \quad (\text{B.101})$$

$$\dot{\vec{x}}_0 = \xi_0 \begin{bmatrix} -\dot{\alpha} \sin(\alpha) \\ \dot{\alpha} \cos(\alpha) \end{bmatrix} \quad (\text{B.102})$$

$$\ddot{\vec{x}}_0 = \xi_0 \begin{bmatrix} -\ddot{\alpha} \sin(\alpha) - \dot{\alpha}^2 \cos(\alpha) \\ \ddot{\alpha} \cos(\alpha) - \dot{\alpha}^2 \sin(\alpha) \end{bmatrix} \quad (\text{B.103})$$

The numerical derivations presented in appendices B.1 and B.2 as well as in this section allow the computation of the x - and y -coordinates as well as the orientation α of the robot as explicit functions of time. In the current implementation, the robot trajectory is supplied to the robot as a list of closely spaced (t, x, y, α) entries with a fixed time increment, e.g. 40 ms.

B.4 CAT Method: Trajectory Shape

This section describes the computation of the tangent points according to the circle and tangent trajectory shape generation approach presented in section 3.6.1. The trajectory is defined by a list of consecutive control points containing the coordinates x_p and y_p of the centers of circles that need to be surrounded. In addition to that, the curve radius r_p of each circle must be provided. Refer to section 6.1.2 for the discussion of a sample trajectory based on the control points listed in table 6.3, p. 109. The list of control points is processed from top to bottom by taking two adjacent control points and computing the coordinates of the connecting tangent line. Let $|r_1|$ and $|r_2|$ be the radii of two consecutive circles in the list of control points. Either radius may be positive, $r > 0$, if the circle is to be surrounded anticlockwise, or negative, $r < 0$, if the circle is to be surrounded clockwise. Then four distinct topological cases can be distinguished that determine the calculation of the tangent points:

Case 1: $r_1 \cdot r_2 = 0$

At least one radius is zero: Connect a circle to a point, a point to a circle, or a point to a point. The case is used as generic case. The computation of the tangent point T of a point 1 with a circle 2 as illustrated in figure B.1 will be used by the other cases. Let \vec{x}_1 and \vec{x}_2 be the given coordinates of the point and center of the circle. Let l be the distance of these two points,

$$l^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2 \quad (\text{B.104})$$

Let r be the radius of circle 2 and d be the distance from point 1 to the yet unknown tangent point T :

$$l^2 = r^2 + d^2 \quad (\text{B.105})$$

Then the vector connecting points 1 and 2 can be expressed as

$$\vec{x}_{12} = \vec{x}_{1T} + \vec{x}_{T2} \quad (\text{B.106})$$

$$= \vec{x}_{1T} + \frac{r}{d} \vec{x}_{\perp(1T)} \quad (\text{B.107})$$

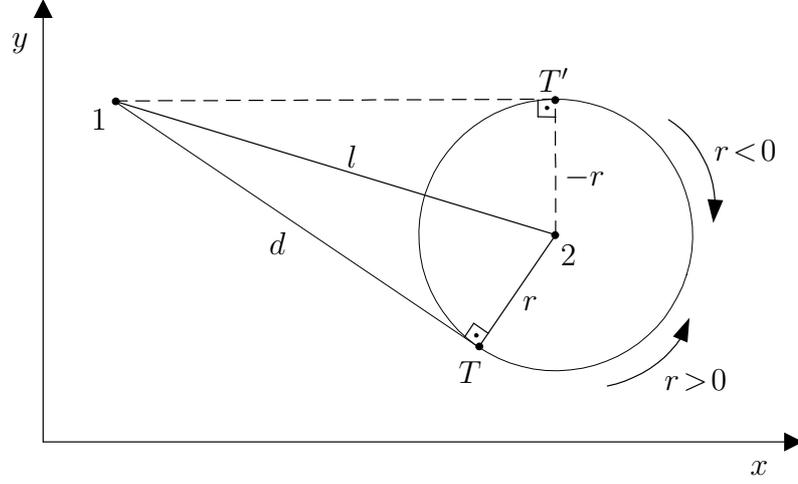


Figure B.1: Circle and tangent shape – case 1: Generic tangent point

with $\vec{x}_{\perp(1T)}$ being a vector that is obtained by rotating vector \vec{x}_{1T} by 90° anticlockwise. The components of the above equation are then

$$x_2 - x_1 = x_T - x_1 - \frac{r}{d} (y_T - y_1) \quad (\text{B.108})$$

$$y_2 - y_1 = y_T - y_1 + \frac{r}{d} (x_T - x_1) \quad (\text{B.109})$$

This set of linear equations can be written as a matrix multiplication:

$$\begin{pmatrix} 1 & -\frac{r}{d} \\ \frac{r}{d} & 1 \end{pmatrix} \begin{pmatrix} x_T \\ y_T \end{pmatrix} = \begin{pmatrix} -\frac{r}{d} y_1 + x_2 \\ \frac{r}{d} x_1 + y_2 \end{pmatrix} \quad (\text{B.110})$$

Inverting the matrix on the left allows calculation of the coordinates of the tangent point T :

$$\begin{pmatrix} x_T \\ y_T \end{pmatrix} = \frac{1}{1 + \frac{r^2}{d^2}} \begin{pmatrix} \frac{r^2}{d^2} x_1 + \frac{r}{d} (y_2 - y_1) + x_2 \\ \frac{r^2}{d^2} y_1 - \frac{r}{d} (x_2 - x_1) + y_2 \end{pmatrix} \quad (\text{B.111})$$

with

$$\frac{r^2}{d^2} = \frac{1}{\frac{l^2}{r^2} - 1} \quad (\text{B.112})$$

$$\frac{1}{1 + \frac{r^2}{d^2}} = 1 - \frac{r^2}{l^2} \quad (\text{B.113})$$

Note that with equation (B.111), the coordinates of point T in figure B.1 will be computed for $r > 0$, it will yield the coordinates of point T' for $r < 0$,

and the coordinates of the circle center for $r=0$. It will also work if a circle 1 is connected to a point 2 ($r_1 \neq 0$ and $r_2=0$).

Case 2: $r_1 \cdot r_2 < 0$

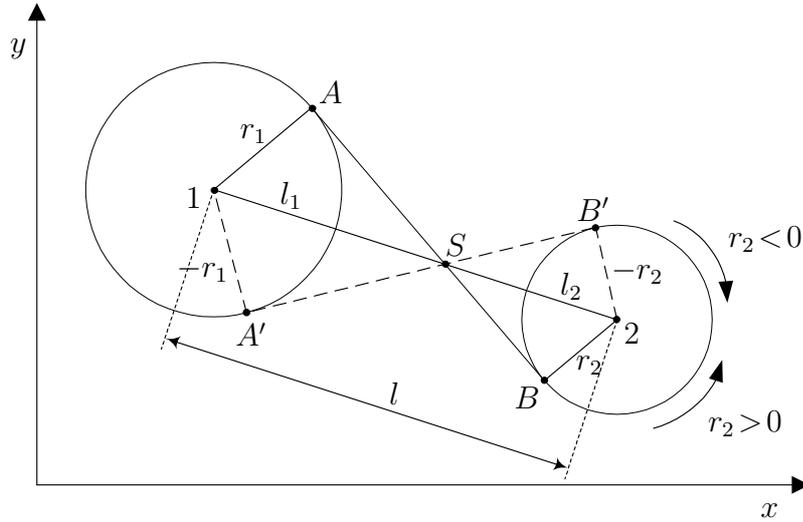


Figure B.2: Circle and tangent shape – case 2: Circle to circle, odd

The radii have a different sign: Connect a circle to a circle with opposite orientation. This case can be reduced to case 1 by computing the coordinates of the intersect point S located on the centerline $\overline{12}$ of the two circles between points 1 and 2, see figure B.2.

$$\vec{x}_S = \vec{x}_1 + \frac{l_1}{l} \vec{x}_{12} \tag{B.114}$$

with

$$\frac{l_1}{l} = \frac{|r_1|}{|r_1| + |r_2|} = \frac{1}{1 - \frac{r_2}{r_1}} \tag{B.115}$$

which takes into account the different signs of r_1 and r_2 .

Now the two tangent points A and B can be obtained in two separate steps, each step can be calculated using case 1:

- Tangent point A : connect circle 1 to point S
- Tangent point B : connect point S to circle 2

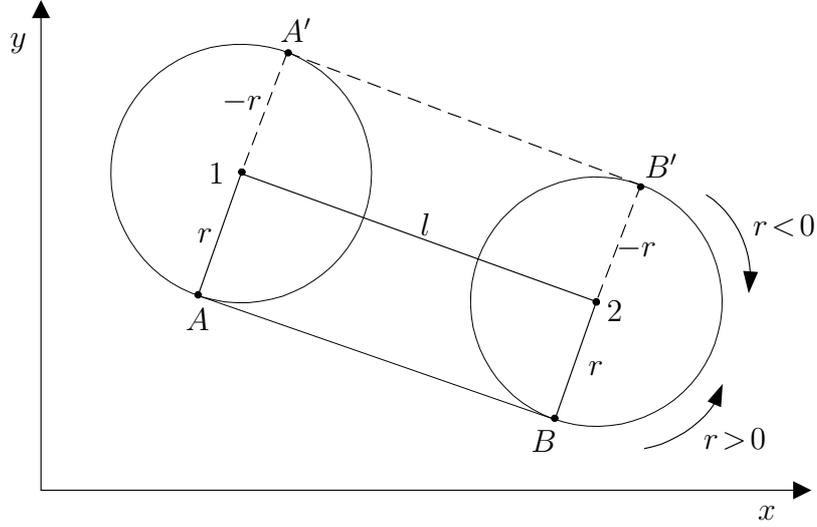


Figure B.3: Circle and tangent shape – case 3a: Circle to equal circle

Case 3a: $r_1 \cdot r_2 > 0$, $r_1 = r_2 = r$

The radii are equal and have the same sign: Connect two equal circles with same orientations. This is a special case that cannot be reduced to case 1 because no intersect point can be constructed. The tangent line \overline{AB} is parallel to the centerline $\overline{12}$, see figure B.3.

$$\vec{x}_A = \vec{x}_1 + \vec{x}_{1A} = \vec{x}_1 + \frac{r}{l} \vec{x}_{\perp(12)} \quad (\text{B.116})$$

$$\vec{x}_B = \vec{x}_2 + \vec{x}_{1A} = \vec{x}_2 + \frac{r}{l} \vec{x}_{\perp(12)} \quad (\text{B.117})$$

with $\vec{x}_{\perp(12)}$ being a vector that is obtained by rotating vector \vec{x}_{12} by 90° clockwise:

$$\vec{x}_{\perp(12)} = \begin{pmatrix} (y_2 - y_1) \\ -(x_2 - x_1) \end{pmatrix} \quad (\text{B.118})$$

Case 3b: $r_1 \cdot r_2 > 0$, $r_1 \neq r_2$

The radii have the same sign but different values: Connect a circle to a circle with equal orientations. This case can be reduced to case 1 by computing the coordinates of the intersect point S located on the centerline $\overline{12}$ of the two circles outside points 1 and 2, see figure B.4:

$$\vec{x}_S = \vec{x}_1 + \frac{l_1}{l} \vec{x}_{12} \quad (\text{B.119})$$

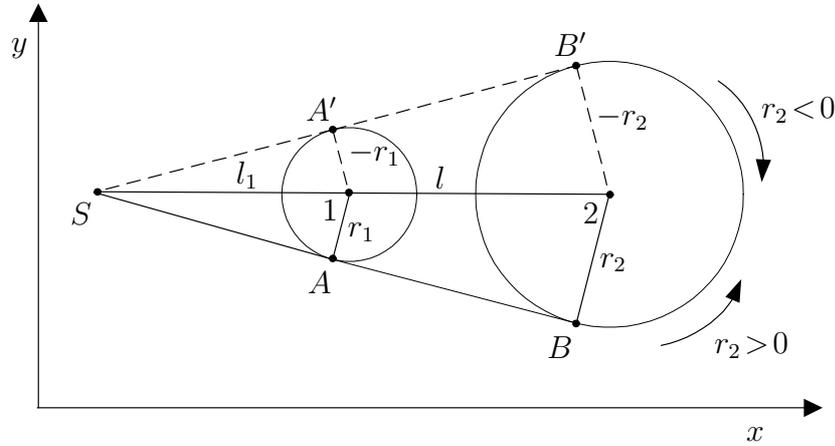


Figure B.4: Circle and tangent shape – case 3b: Circle to circle, even

with

$$\frac{l_1}{l} = \frac{|r_1|}{|r_1| + |r_2|} = \frac{1}{1 - \frac{r_2}{r_1}} \quad (\text{B.120})$$

which takes into account the different signs of r_1 and r_2 .

Now the two tangent points A and B can be obtained in two separate steps, each step can be calculated using case 1:

If $\frac{l_1}{l} < 0$:

- Tangent point A : connect point S to circle 1
- Tangent point B : connect point S to circle 2

If $\frac{l_1}{l} > 1$:

- Tangent point A : connect circle 1 to point S
- Tangent point B : connect circle 2 to point S

B.5 CAT Method: Trajectory Speed

This section describes the computation of the velocity profile along the generic trajectory according to the theory described in section 3.6.2.

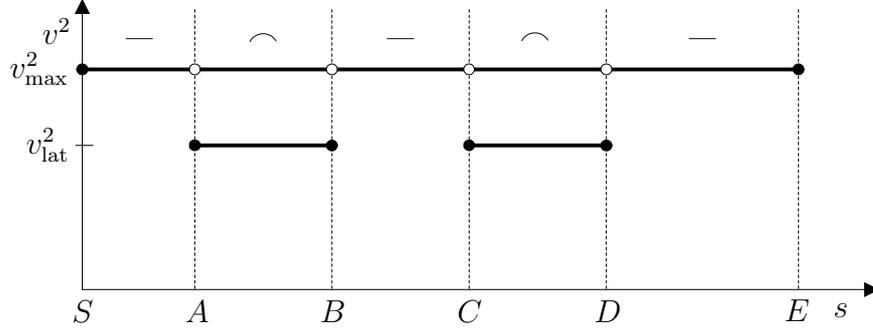


Figure B.5: Circle and tangent speed – step 1: Velocity constraints

Step 1 Velocity Constraints

Figure B.5 illustrates the two constraints on the velocity that can be derived from the physical restrictions of the robot hardware in the velocity squared–distance diagram for the simple example presented in figure 3.5, p. 59. The diagram is divided into five alternating straight and circular sections. The straight sections \overline{SA} , \overline{BC} , and \overline{DE} are marked with a '—', the circular sections \widehat{AB} and \widehat{CD} are marked with a '⌒'. For all sections, the velocity constraint v_{\max}^2 according to equation (3.63), p. 60, is binding because the robot can never travel faster than v_{\max} . For the two circular sections, the lateral acceleration according to equation (3.65), p. 60, additionally constrains the velocity. Note that in this example, both circles have the same radius, resulting in the same velocity constraint v_{lat}^2 for both circular sections. Typically, this constraint is harder than the maximum velocity constraint, $v_{\text{lat}}^2 < v_{\max}^2$. It is furthermore assumed that both start and end velocities are zero.

Step 2 Acceleration Constraints

This step scans the five sections from first to last in ascending order. It will always assume the smallest value that is allowed and possible for the initial velocity of each section. It is the minimum of the two constraints and the end velocity from the previous section:

$$v_{\text{start,section}}^2 = \min(v_{\max}^2, v_{\text{lat}}^2, v_{\text{end,previous section}}^2) \quad (\text{B.121})$$

In figure B.6, the binding constraints for the start velocity in each section are marked with a filled dot. The non-binding constraints are marked with an empty dot. In the first section, acceleration starts at the start velocity 0. The acceleration value is the parameter a_{acc} which is the same for all sections. In this example, the start velocity in the first interval is zero, the velocity

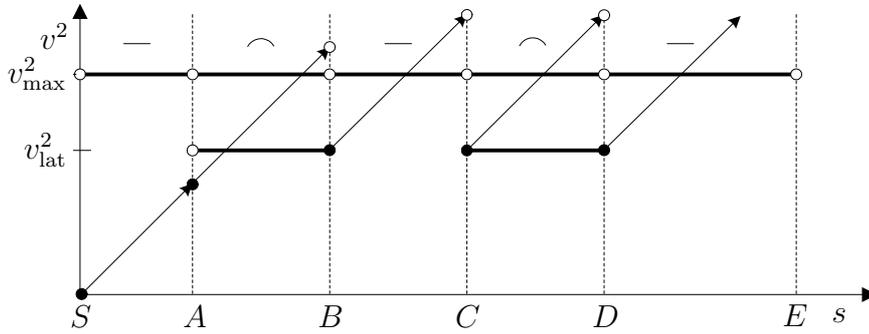


Figure B.6: Circle and tangent speed – step 2: Acceleration constraints

at the end of the first interval does not violate any of the constraints yet. Therefore, the second section starts with the end velocity of the first section because it is smaller than the two constraints at point A . In the course of section two, the velocity grows larger than v_{lat}^2 and v_{max}^2 , but this is ignored in this first step. The starting velocity in the third section is the smallest of the three velocities in point B which is v_{lat}^2 . In the course of section three, the velocity grows larger than v_{max}^2 which is ignored again. In both sections four and five, the lateral acceleration constraint serves as start velocity.

Step 3 Deceleration Constraints

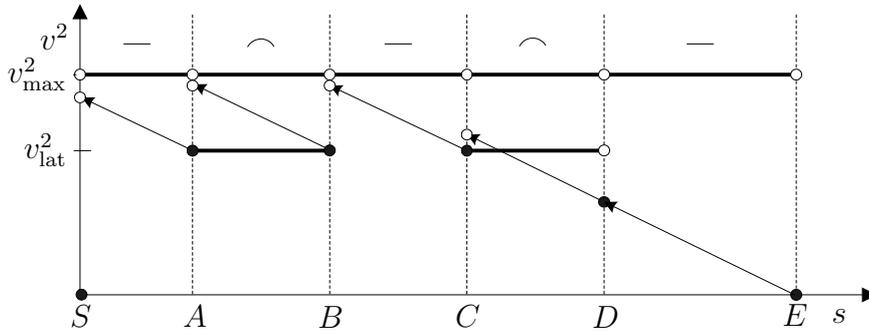


Figure B.7: Circle and tangent speed – step 3: Deceleration constraints

This step scans the five sections from last to first in descending order. The procedure is analog to the one in step 2, only that it moves backwards in time and distance and uses the deceleration value a_{dec} as negative acceleration. Typically, the deceleration value is smaller than the acceleration value because during acceleration, the ball will be pushed towards the robot while during deceleration, it might separate from the robot which must be avoided.

In analogy to step 2, the algorithm will always assume the smallest velocity that is allowed and possible as end velocity of each section and calculate the maximum allowed start velocity. The end velocity is the minimum of the two constraints and the start velocity from the subsequent section:

$$v_{\text{end,section}}^2 = \min(v_{\text{max}}^2, v_{\text{lat}}^2, v_{\text{start,subsequent section}}^2) \quad (\text{B.122})$$

Note that *subsequent* means *later in time* respectively *at larger distance s* . In the fifth section, deceleration must end at zero velocity. In this example, the velocity at the start of the fifth interval determined by permanent deceleration does not violate any of the constraints yet. Therefore, the fourth section ends with the start velocity of the fifth section because it is smaller than the two constraints at point D . In the course of section four (going backwards in time and distance), the velocity grows larger than v_{lat}^2 which is ignored here. The end velocity in the third section is the smallest of the three velocities in point C which is v_{lat}^2 . At the start of section three, the velocity has reached a value larger than v_{lat}^2 . Therefore, section two must end with v_{lat}^2 . The same happens in the first section: The end velocity must be v_{lat}^2 .

Step 4 Final Design

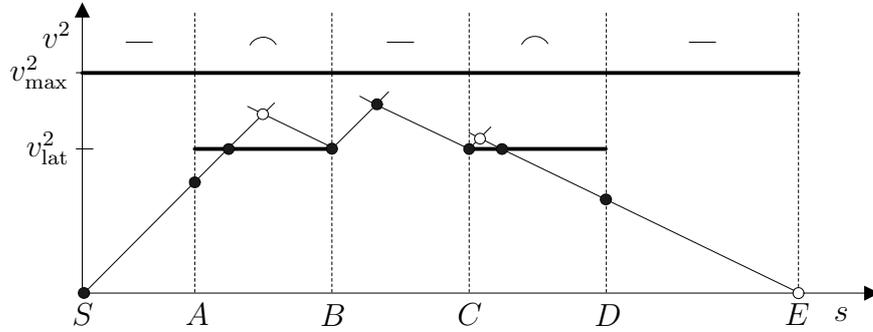


Figure B.8: Circle and tangent speed – step 4: Final design

The input to this step are the minimum possible velocities at the four tangent points obtained from the acceleration and deceleration considerations. Now, intermediate points are calculated that connect intervals with constant acceleration, deceleration, and constant velocity. These intermediate points are the intersects of the acceleration, deceleration, and velocity constraint lines. They are calculated for each section separately with equation (B.123) and the algorithm presented in table B.1.

Let $v_1 \geq 0$ be the velocity at a given distance s_1 with a subsequent acceleration $a_1 \geq 0$. At an unknown distance $s^* \geq s_1$ with an unknown velocity $v^* \geq v_1$, the acceleration changes to a value $a_2 \leq 0$ such that at a given distance $s_2 \geq s^*$, the velocity reaches a value $v_2 \geq 0$. The distance s^* and velocity v^* of the intermediate point can then be calculated with

$$s^* = \frac{\frac{1}{2}(v_1^2 - v_2^2) + a_2 s_2 - a_1 s_1}{a_2 - a_1} \quad (\text{B.123})$$

$$v^{*2} = \frac{a_2 v_1^2 - a_1 v_2^2 + 2a_1 a_2 (s_2 - s_1)}{a_2 - a_1} \quad (\text{B.124})$$

For the calculation of the intersect of an acceleration or deceleration phase with a constraint line, the respective acceleration for the constraint line must be set to zero. Note that the distance of the intersect s^* will always be located in the interval $[s_1, s_2]$ because of the preconditioning of the velocities v_1 and v_2 in steps 2 and 3.

In sections 1 and 5, the intersect points s^* are located on the tangent points A and D, respectively. In the two circular sections 2 and 4, there are intermediate solutions s^* plotted in figure B.8 with empty dots. Their respective velocities v^* are larger than the constraints v_{lat} and are therefore ignored. In the middle section 3, an acceleration and deceleration is possible.

```

LOOP over sections
  calculate  $v^{*2}$  (acceleration/deceleration)
  IF  $v^{*2}$  violates  $v_{\text{max}}^2$  or  $v_{\text{lat}}^2$  THEN
    calculate  $s^*_1$  (acceleration/constraint)
    calculate  $s^*_2$  (constraint/deceleration)
  ELSE
    calculate  $s^*_1$  (acceleration/deceleration)
  END
  append  $s^*_1$  and  $s^*_2$  to list
END LOOP over sections

```

Table B.1: CAT approach: Trajectory speed algorithm

The result of step 4 is a list of intervals with constant acceleration. Adjacent intervals with equal acceleration can be combined to one interval. The start and end distances and velocities of each interval are also known. This allows an explicit computation of the distance and velocity as functions of time. In addition to that, the x - and y -coordinates can also be computed as functions of time. The generic trajectory is supplied as a list of closely spaced $(t, x_T, y_T, s, v, \dot{v})$ -entries. This list is used as a lookup table during

generation of the robot and ball trajectory. The time t in this list is the independent variable during creation of the table. It is increased with a fixed time increment, typically 40 ms. The dependent variables are the coordinates x_T and y_T of the points on the generic trajectory, the distance s along the generic trajectory, the velocity v , and the longitudinal acceleration \dot{v} .

B.6 CAT Method: Robot Trajectory

Both robot and ball trajectories were already provided in section 3.6.3, recall equations (3.72) and (3.73), p. 62:

$$\vec{x}_R = \vec{x}_T - \psi \vec{x}_0(\alpha) \quad (\text{B.125})$$

$$\vec{x}_B = \vec{x}_T + (1 - \psi) \vec{x}_0(\alpha) \quad (\text{B.126})$$

The robot orientation can be obtained by constructing a point S on the generic trajectory with the secant method. This was derived in the same section and yielded equation (3.80), p. 65:

$$\alpha = \arctan\left(\frac{y_S - y_T}{x_S - x_T}\right) \quad (\text{B.127})$$

The coordinates (x_T, y_T) of the point S are obtained with linear interpolation of the lookup table created in appendix B.5. Recall that point S is obtained from point T by adding the secant length \widehat{s} according to equation (3.79), p. 65, to the distance s_T along the generic trajectory:

$$s_S = s_T + \widehat{s} \quad (\text{B.128})$$

$$x_S = x(s_S) = x(s_T + \widehat{s}) \quad (\text{B.129})$$

$$y_S = y(s_S) = y(s_T + \widehat{s}) \quad (\text{B.130})$$

This is all the information required for creating the robot position lookup table (t, x_R, y_R, α) that only contains the time and the robot position and orientation.

This section provides additional information on how to obtain the derivatives of the robot position with respect to time. These were used to compute figures 6.11, p. 113, to 6.17, p. 117, in section 6.1.2. The robot velocity and acceleration are computed by taking the derivative of equation (B.125) with respect to time:

$$\dot{\vec{x}}_R = \dot{\vec{x}}_T - \psi \dot{\vec{x}}_0(\alpha) \quad (\text{B.131})$$

$$\ddot{\vec{x}}_R = \ddot{\vec{x}}_T - \psi \ddot{\vec{x}}_0(\alpha) \quad (\text{B.132})$$

The ball neutral position $\vec{x}_0(\alpha)$ and its derivatives are the same as for the MCL method, compare equations (B.101) to (B.103):

$$\vec{x}_0 = \xi_0 \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix} \quad (\text{B.133})$$

$$\dot{\vec{x}}_0 = \xi_0 \begin{bmatrix} -\dot{\alpha} \sin(\alpha) \\ \dot{\alpha} \cos(\alpha) \end{bmatrix} \quad (\text{B.134})$$

$$\ddot{\vec{x}}_0 = \xi_0 \begin{bmatrix} -\ddot{\alpha} \sin(\alpha) - \dot{\alpha}^2 \cos(\alpha) \\ \ddot{\alpha} \cos(\alpha) - \dot{\alpha}^2 \sin(\alpha) \end{bmatrix} \quad (\text{B.135})$$

The derivatives of the robot orientation α are computed in the following. Using the definitions

$$\hat{x} \stackrel{\text{def}}{=} x_S - x_T \quad (\text{B.136})$$

$$\hat{y} \stackrel{\text{def}}{=} y_S - y_T \quad (\text{B.137})$$

$$f \stackrel{\text{def}}{=} \frac{\hat{y}}{\hat{x}} \quad (\text{B.138})$$

$$g \stackrel{\text{def}}{=} \frac{\hat{x}}{\hat{y}} = \frac{1}{f}, \quad (\text{B.139})$$

equation (B.127) can reformulated:

$$\alpha = \arctan(f) \quad (\text{B.140})$$

It can be shown that

$$\frac{d}{dt} \left(\arctan(f(t)) \right) = -\frac{d}{dt} \left(\arctan\left(\frac{1}{f(t)}\right) \right). \quad (\text{B.141})$$

This means the derivatives of α can be expressed in two ways:

$$\dot{\alpha} = \frac{d}{dt} \left(\arctan(f) \right) = -\frac{d}{dt} \left(\arctan(g) \right) \quad (\text{B.142})$$

For the derivatives of f , two cases are considered for different sectors of the orientation:

The derivatives of α are computed in terms of f for all points on the generic trajectory with $\boxed{|\hat{y}| \leq |\hat{x}|}$. Note that the case $\hat{x} = 0$ is excluded by this condition.

$$\dot{\alpha} = \frac{1}{1+f^2} \dot{f} \quad (\text{B.143})$$

$$\ddot{\alpha} = \frac{1}{1+f^2} \left(\ddot{f} - \frac{2f\dot{f}^2}{1+f^2} \right) \quad (\text{B.144})$$

The derivatives of f are obtained as follows:

$$\dot{f} = \frac{\widehat{x}\dot{\widehat{y}} - \dot{\widehat{x}}\widehat{y}}{\widehat{x}^2} \quad (\text{B.145})$$

$$\ddot{f} = \frac{\widehat{x}(\widehat{x}\ddot{\widehat{y}} - \ddot{\widehat{x}}\widehat{y}) - 2\dot{\widehat{x}}(\widehat{x}\dot{\widehat{y}} - \dot{\widehat{x}}\widehat{y})}{\widehat{x}^3} \quad (\text{B.146})$$

The derivatives of α are computed in terms of g for all points on the generic trajectory with $|\widehat{y}| > |\widehat{x}|$. Note that the case $\widehat{y} = 0$ is excluded by the above condition.

$$\dot{\alpha} = -\frac{1}{1+g^2} \dot{g} \quad (\text{B.147})$$

$$\ddot{\alpha} = -\frac{1}{1+g^2} \left(\ddot{g} - \frac{2g\dot{g}^2}{1+g^2} \right) \quad (\text{B.148})$$

The derivatives of g are obtained as follows:

$$\dot{g} = \frac{\widehat{y}\dot{\widehat{x}} - \dot{\widehat{y}}\widehat{x}}{\widehat{y}^2} \quad (\text{B.149})$$

$$\ddot{g} = \frac{\widehat{y}(\widehat{y}\ddot{\widehat{x}} - \ddot{\widehat{y}}\widehat{x}) - 2\dot{\widehat{y}}(\widehat{y}\dot{\widehat{x}} - \dot{\widehat{y}}\widehat{x})}{\widehat{y}^3} \quad (\text{B.150})$$

For the computation of the derivatives of \widehat{x} and \widehat{y} it is no longer necessary to distinguish between the two cases:

$$\dot{\widehat{x}} = \dot{x}_S - \dot{x}_T \quad (\text{B.151})$$

$$\ddot{\widehat{x}} = \ddot{x}_S - \ddot{x}_T \quad (\text{B.152})$$

At this point it is important to emphasize that point S is always at the same constant distance \widehat{s} in front of point T . This is expressed in equation (B.128). That means, S moves along the trajectory with the same velocity as point T . Therefore, the velocities of point S and point T are equal:

$$v_S = v_T \quad (\text{B.153})$$

This must be strictly distinguished from the local velocities calculated as derivatives of the curve length with respect to time, they are not equal:

$$(v)_S = \left(\frac{ds}{dt} \right)_S \neq \left(\frac{ds}{dt} \right)_T = (v)_T \quad (\text{B.154})$$

In contrast to point S , point T does move along the generic trajectory with the local velocity v and acceleration \dot{v} calculated as derivatives of the curve length with respect to time:

$$(v)_T = v_T \quad (\dot{v})_T = \dot{v}_T \quad (\text{B.155})$$

$$(v)_S \neq v_S \quad (\dot{v})_S \neq \dot{v}_S \quad (\text{B.156})$$

The velocity of point S in x -direction \dot{x}_S can be obtained with a variation δs_T of the distance at point T . According to equation (B.128), this will cause the same variation at point S :

$$\delta s_S = \delta s_T = \delta s \quad (\text{B.157})$$

Assuming that this variation in distance occurs during the time variation δt allows to express the velocity of point T :

$$v_T = \frac{\delta s}{\delta t} \quad (\text{B.158})$$

The variation of the distance of point T will cause the following variation of the x -coordinate of point S :

$$\delta x_S = \left(\frac{dx}{ds} \right)_S \delta s \quad (\text{B.159})$$

The term $\left(\frac{dx}{ds} \right)_S$ is the local derivative of the x -coordinate of the generic trajectory at point S and can be expanded as:

$$\left(\frac{dx}{ds} \right)_S = \left(\frac{dx}{dt} \cdot \frac{dt}{ds} \right)_S = \left(\frac{\dot{x}}{v} \right)_S \quad (\text{B.160})$$

The value of \dot{x}_S can now be computed as

$$\dot{x}_S = \frac{\delta x_S}{\delta t} = \left(\frac{\dot{x}}{v} \right)_S \cdot \frac{\delta s}{\delta t} = \left(\frac{\dot{x}}{v} \right)_S \cdot v_T \quad (\text{B.161})$$

The variation of the x -velocity of point S will yield its acceleration:

$$\begin{aligned} \delta \dot{x}_S &= \left(\frac{\frac{d\dot{x}}{ds} \cdot \delta s \cdot v - \dot{x} \cdot \frac{dv}{ds} \cdot \delta s}{v^2} \right)_S \cdot (v)_T + \left(\frac{\dot{x}}{v} \right)_S \cdot \left(\frac{dv}{ds} \right)_T \cdot \delta s \\ &= \left(\frac{\frac{d\dot{x}}{dt} \frac{dt}{ds} \cdot v - \dot{x} \cdot \frac{dv}{dt} \frac{dt}{ds}}{v^2} \right)_S \cdot v_T \cdot \delta s + \left(\frac{\dot{x}}{v} \right)_S \cdot \left(\frac{dv}{dt} \frac{dt}{ds} \right)_T \cdot \delta s \end{aligned} \quad (\text{B.162})$$

$$\begin{aligned} \ddot{x}_S &= \frac{\delta \dot{x}_S}{\delta t} \\ &= \left(\frac{\frac{d\ddot{x}}{ds} \cdot v - \dot{x} \cdot \frac{d\dot{v}}{ds}}{v^2} \right)_S \cdot v_T \cdot \frac{\delta s}{\delta t} + \left(\frac{\dot{x}}{v} \right)_S \cdot \left(\frac{\dot{v}}{v} \right)_T \cdot \frac{\delta s}{\delta t} \\ &= \left(\frac{\ddot{x} v - \dot{x} \dot{v}}{v^3} \right)_S \cdot v_T^2 + \left(\frac{\dot{x}}{v} \right)_S \cdot \frac{\dot{v}_T}{v_T} \cdot v_T \end{aligned} \quad (\text{B.163})$$

The velocity $(v)_S$ and the longitudinal acceleration $(\dot{v})_S$ are the derivatives of the distance s along the generic trajectory with respect to time at the point S . They are obtained with linear interpolation of the lookup table created in appendix B.5 in analogy to equations (B.129) and (B.130):

$$(v)_S = v(s_S) = v(s_T + \bar{s}) \quad (\text{B.164})$$

$$(\dot{v})_S = \dot{v}(s_S) = \dot{v}(s_T + \bar{s}) \quad (\text{B.165})$$

The resulting derivatives \hat{x} and \hat{y} with respect to time are:

$$\dot{\hat{x}} = \left(\frac{\dot{x}}{v}\right)_S \cdot v_T - \dot{x}_T \quad (\text{B.166})$$

$$\dot{\hat{y}} = \left(\frac{\dot{y}}{v}\right)_S \cdot v_T - \dot{y}_T \quad (\text{B.167})$$

$$\ddot{\hat{x}} = \left(\frac{\ddot{x}v - \dot{x}\dot{v}}{v^3}\right)_S \cdot v_T^2 + \left(\frac{\dot{x}}{v}\right)_S \cdot \dot{v}_T - \ddot{x}_T \quad (\text{B.168})$$

$$\ddot{\hat{y}} = \left(\frac{\ddot{y}v - \dot{y}\dot{v}}{v^3}\right)_S \cdot v_T^2 + \left(\frac{\dot{y}}{v}\right)_S \cdot \dot{v}_T - \ddot{y}_T \quad (\text{B.169})$$

B.7 Image Processing: Robot Self Localization

The distance of a pixel to a geometrical object defined in equations (5.16) and (5.17), p. 91, expands to the following forms for each of the three objects implemented:

$$d_i(O_x^l) = -(\xi_F - \xi_i) \sin(\bar{\alpha}) + (\eta_F - \eta_i) \cos(\bar{\alpha}) - \Delta_x^l \quad (\text{B.170})$$

$$d_i(O_y^l) = (\xi_F - \xi_i) \cos(\bar{\alpha}) + (\eta_F - \eta_i) \sin(\bar{\alpha}) - \Delta_y^l \quad (\text{B.171})$$

$$d_i(O_c^l) = (\xi_F - \xi_i)^2 + (\eta_F - \eta_i)^2 - (\Delta_c^l)^2 \quad (\text{B.172})$$

These distances are inserted into equation (5.19), p. 91, after the derivative with respect to the solution vector according to equation (5.2), p. 88, has been taken. Each residual contains contributions from pixels that have been assigned to one of the three object classes. The number of pixels that has been assigned to object instance O_k^l is referred to as $n_{\text{pix}}(O_k^l)$. The weight factors w_k^l can compensate different scales due to different units of measurement: Lines are in units $[d_i] = \text{m}$, circles are in units $[d_i] = \text{m}^2$. The three entries of the residual vector \mathbf{R} according to equation (5.24), p. 93, then

compute as follows:

$$\begin{aligned}
R_\xi &= \sum_{l=1}^{n(O_x)} w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} \left[(\xi_F - \xi_i) \sin^2(\bar{\alpha}) - (\eta_F - \eta_i) \sin(\bar{\alpha}) \cos(\bar{\alpha}) + \Delta_x^l \sin(\bar{\alpha}) \right] \\
&+ \sum_{l=1}^{n(O_y)} w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} \left[(\xi_F - \xi_i) \cos^2(\bar{\alpha}) + (\eta_F - \eta_i) \sin(\bar{\alpha}) \cos(\bar{\alpha}) - \Delta_y^l \cos(\bar{\alpha}) \right] \\
&+ 2 \sum_{l=1}^{n(O_c)} w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \left[(\xi_F - \xi_i)^3 + (\xi_F - \xi_i)(\eta_F - \eta_i)^2 - (\Delta_c^l)^2 (\xi_F - \xi_i) \right] \\
R_\eta &= \sum_{l=1}^{n(O_x)} w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} \left[-(\xi_F - \xi_i) \sin(\bar{\alpha}) \cos(\bar{\alpha}) + (\eta_F - \eta_i) \cos^2(\bar{\alpha}) - \Delta_y^l \cos(\bar{\alpha}) \right] \\
&+ \sum_{l=1}^{n(O_y)} w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} \left[(\xi_F - \xi_i) \sin(\bar{\alpha}) \cos(\bar{\alpha}) + (\eta_F - \eta_i) \sin^2(\bar{\alpha}) - \Delta_x^l \sin(\bar{\alpha}) \right] \\
&+ 2 \sum_{l=1}^{n(O_c)} w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \left[(\xi_F - \xi_i)^2 (\eta_F - \eta_i) + (\eta_F - \eta_i)^3 - (\Delta_c^l)^2 (\eta_F - \eta_i) \right] \\
R_{\bar{\alpha}} &= \sum_{l=1}^{n(O_x)} w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} \left[\left((\xi_F - \xi_i)^2 - (\eta_F - \eta_i)^2 \right) \sin(\bar{\alpha}) \cos(\bar{\alpha}) \right. \\
&\quad \left. + (\xi_F - \xi_i)(\eta_F - \eta_i) \left(\sin^2(\bar{\alpha}) - \cos^2(\bar{\alpha}) \right) \right. \\
&\quad \left. + \Delta_y^l \left((\xi_F - \xi_i) \cos(\bar{\alpha}) + (\eta_F - \eta_i) \sin(\bar{\alpha}) \right) \right] \\
&+ \sum_{l=1}^{n(O_y)} w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} \left[\left((\eta_F - \eta_i)^2 - (\xi_F - \xi_i)^2 \right) \sin(\bar{\alpha}) \cos(\bar{\alpha}) \right. \\
&\quad \left. - (\xi_F - \xi_i)(\eta_F - \eta_i) \left(\sin^2(\bar{\alpha}) - \cos^2(\bar{\alpha}) \right) \right. \\
&\quad \left. + \Delta_x^l \left((\xi_F - \xi_i) \sin(\bar{\alpha}) - (\eta_F - \eta_i) \cos(\bar{\alpha}) \right) \right]
\end{aligned}$$

Introducing coefficients based on the pixel coordinates according to table B.2, the residuals can be expressed in the following way:

$$\begin{aligned}
R_\xi &= (A_x \xi_F + B_x) \sin^2(\bar{\alpha}) + (A_y \xi_F - B_y) \cos^2(\bar{\alpha}) & (B.173) \\
&- (A \eta_F - C) \sin(\bar{\alpha}) \cos(\bar{\alpha}) + G_x \sin(\bar{\alpha}) - G_y \cos(\bar{\alpha}) \\
&+ 2 \left[\left(A_c (\xi_F^2 + \eta_F^2) + 3D_c + F_c - K_c \right) \xi_F - 2(C_c \xi_F - E_c) \eta_F \right. \\
&\quad \left. - B_c (3 \xi_F^2 + \eta_F^2) - G_c - I_c + L_c \right]
\end{aligned}$$

lines x	lines y	circles
$A_x^l = w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} 1$	$A_y^l = w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} 1$	$A_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} 1$
$B_x^l = w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} \xi_i$	$B_y^l = w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} \xi_i$	$B_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \xi_i$
$C_x^l = w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} \eta_i$	$C_y^l = w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} \eta_i$	$C_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \eta_i$
$D_x^l = w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} \xi_i^2$	$D_y^l = w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} \xi_i^2$	$D_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \xi_i^2$
$E_x^l = w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} \xi_i \eta_i$	$E_y^l = w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} \xi_i \eta_i$	$E_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \xi_i \eta_i$
$F_x^l = w_x^l \sum_{i=1}^{n_{\text{pix}}(O_x^l)} \eta_i^2$	$F_y^l = w_y^l \sum_{i=1}^{n_{\text{pix}}(O_y^l)} \eta_i^2$	$F_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \eta_i^2$
		$G_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \xi_i^3$
		$H_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \xi_i^2 \eta_i$
		$I_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \xi_i \eta_i^2$
		$J_c^l = w_c^l \sum_{i=1}^{n_{\text{pix}}(O_c^l)} \eta_i^3$
$A_x = \sum_{l=1}^{n(O_x)} A_x^l$	$A_y = \sum_{l=1}^{n(O_y)} A_y^l$	$A_c = \sum_{l=1}^{n(O_c)} A_c^l$
$B_x = \sum_{l=1}^{n(O_x)} B_x^l$	$B_y = \sum_{l=1}^{n(O_y)} B_y^l$	$B_c = \sum_{l=1}^{n(O_c)} B_c^l$
$C_x = \sum_{l=1}^{n(O_x)} C_x^l$	$C_y = \sum_{l=1}^{n(O_y)} C_y^l$	$C_c = \sum_{l=1}^{n(O_c)} C_c^l$
$D_x = \sum_{l=1}^{n(O_x)} D_x^l$	$D_y = \sum_{l=1}^{n(O_y)} D_y^l$	$D_c = \sum_{l=1}^{n(O_c)} D_c^l$
$E_x = \sum_{l=1}^{n(O_x)} E_x^l$	$E_y = \sum_{l=1}^{n(O_y)} E_y^l$	$E_c = \sum_{l=1}^{n(O_c)} E_c^l$
$F_x = \sum_{l=1}^{n(O_x)} F_x^l$	$F_y = \sum_{l=1}^{n(O_y)} F_y^l$	$F_c = \sum_{l=1}^{n(O_c)} F_c^l$
		$G_c = \sum_{l=1}^{n(O_c)} G_c^l$
		$H_c = \sum_{l=1}^{n(O_c)} H_c^l$
		$I_c = \sum_{l=1}^{n(O_c)} I_c^l$
		$J_c = \sum_{l=1}^{n(O_c)} J_c^l$
$K_x = \sum_{l=1}^{n(O_x)} A_x^l \Delta_x^l$	$K_y = \sum_{l=1}^{n(O_y)} A_y^l \Delta_y^l$	$K_c = \sum_{l=1}^{n(O_c)} A_c^l (\Delta_c^l)^2$
$L_x = \sum_{l=1}^{n(O_x)} B_x^l \Delta_x^l$	$L_y = \sum_{l=1}^{n(O_y)} B_y^l \Delta_y^l$	$L_c = \sum_{l=1}^{n(O_c)} B_c^l (\Delta_c^l)^2$
$M_x = \sum_{l=1}^{n(O_x)} C_x^l \Delta_x^l$	$M_y = \sum_{l=1}^{n(O_y)} C_y^l \Delta_y^l$	$M_c = \sum_{l=1}^{n(O_c)} C_c^l (\Delta_c^l)^2$

Table B.2: Robot self localization coefficients

$$\begin{aligned}
R_\xi &= (A_x \eta_F - C_x) \cos^2(\bar{\alpha}) + (A_y \eta_F - C_y) \sin^2(\bar{\alpha}) & (B.174) \\
&- (A \xi_F - B) \sin(\bar{\alpha}) \cos(\bar{\alpha}) - G_x \cos(\bar{\alpha}) - G_y \sin(\bar{\alpha}) \\
&+ 2 \left[(A_c (\xi_F^2 + \eta_F^2) + D_c + 3F_c - K_c) \eta_F - 2(B_c \eta_F - E_c) \xi_F \right. \\
&\quad \left. - C_c (3\eta_F^2 + \xi_F^2) - H_c - J_c + M_c \right]
\end{aligned}$$

$$\begin{aligned}
R_{\bar{\alpha}} &= (A(\xi_F^2 - \eta_F^2) - 2(B \xi_F - C \eta_F) + D - F) \sin(\bar{\alpha}) \cos(\bar{\alpha}) & (B.175) \\
&+ ((A \xi_F - B) \eta_F - C \xi_F + E) (\sin^2(\bar{\alpha}) - \cos^2(\bar{\alpha})) \\
&+ ((G_x \xi_F - H_x) - (G_y \eta_F - I_y)) \cos(\bar{\alpha}) \\
&+ ((G_y \xi_F - H_y) + (G_x \eta_F - I_x)) \sin(\bar{\alpha})
\end{aligned}$$

B.8 Image Processing: Ball Localization

The distance of a pixel to the circular ball outline has been defined by equation (5.32), p. 95:

$$d_i = (\mathbf{x}_B - \mathbf{x}_i)^2 + (\mathbf{y}_B - \mathbf{y}_i)^2 - r_B^2$$

Taking the derivative of this distance with respect to the ball solution vector entries according to equation (5.28) and inserting these terms into equation (5.31) yields the following expressions:

$$\frac{dS_B}{d\mathbf{x}_B} = 2 \sum_{i=1}^{n_{\text{pix}}(B)} \left[(\mathbf{x}_B - \mathbf{x}_i)^3 + (\mathbf{x}_B - \mathbf{x}_i)(\mathbf{y}_B - \mathbf{y}_i)^2 - (\mathbf{x}_B - \mathbf{x}_i)r_B^2 \right] \quad (B.176)$$

$$\frac{dS_B}{d\mathbf{y}_B} = 2 \sum_{i=1}^{n_{\text{pix}}(B)} \left[(\mathbf{x}_B - \mathbf{x}_i)^2(\mathbf{y}_B - \mathbf{y}_i) + (\mathbf{y}_B - \mathbf{y}_i)^3 - (\mathbf{y}_B - \mathbf{y}_i)r_B^2 \right] \quad (B.177)$$

$$\frac{dS_B}{d\mathbf{r}_B} = -2 \sum_{i=1}^{n_{\text{pix}}(B)} \left[(\mathbf{x}_B - \mathbf{x}_i)^2 r_B + (\mathbf{y}_B - \mathbf{y}_i)^2 r_B - r_B^3 \right] \quad (B.178)$$

Introducing coefficients based on the pixel coordinates according to table B.3, equations (B.176) to (B.178) can be expressed in the following way:

$$\begin{aligned}
0 &= A_B(\mathbf{x}_B^2 + \mathbf{y}_B^2 - r_B^2) \mathbf{x}_B - B_B(3\mathbf{x}_B^2 + \mathbf{y}_B^2 - r_B^2) \\
&\quad + (3D_B + F_B) \mathbf{x}_B + 2(E_B - C_B \mathbf{x}_B) \mathbf{y}_B - G_B - I_B & (B.179)
\end{aligned}$$

$$\begin{aligned}
0 &= A_B(\mathbf{x}_B^2 + \mathbf{y}_B^2 - r_B^2) \mathbf{y}_B - C_B(\mathbf{x}_B^2 + 3\mathbf{y}_B^2 - r_B^2) \\
&\quad + (D_B + 3F_B) \mathbf{y}_B + 2(E_B - B_B \mathbf{y}_B) \mathbf{x}_B - H_B - J_B & (B.180)
\end{aligned}$$

$$0 = A_B(\mathbf{x}_B^2 + \mathbf{y}_B^2 - r_B^2) r_B + (D_B + F_B - 2(B_B \mathbf{x}_B + C_B \mathbf{y}_B)) r_B \quad (B.181)$$

Equation (B.181) has one solution for $\mathbf{r}_B=0$ which is not a reasonable solution. The remainder of the equation can be solved for the term $A_B(\dots)$ which can be used to remove \mathbf{r}_B from equations (B.179) and (B.180). This yields the linear set of equations

$$0 = \frac{1}{A_B} \left[2(A_B D_B - B_B^2) \mathbf{x}_B + 2(A_B E_B - B_B C_B) \mathbf{y}_B + (B_B(D_B + F_B) - A_B(G_B + I_B)) \right] \quad (\text{B.182})$$

$$0 = \frac{1}{A_B} \left[2(A_B E_B - B_B C_B) \mathbf{x}_B + 2(A_B F_B - C_B^2) \mathbf{y}_B + (C_B(D_B + F_B) - A_B(H_B + J_B)) \right] \quad (\text{B.183})$$

for the variables \mathbf{x}_B and \mathbf{y}_B which can be solved explicitly:

$$\mathbf{x}_B = \frac{1}{\mathbf{f}_B} (\mathbf{b}_B \mathbf{d}_B - \mathbf{e}_B \mathbf{c}_B) \quad (\text{B.184})$$

$$\mathbf{y}_B = \frac{1}{\mathbf{f}_B} (\mathbf{a}_B \mathbf{e}_B - \mathbf{d}_B \mathbf{c}_B) \quad (\text{B.185})$$

$$\mathbf{r}_B^2 = \mathbf{x}_B^2 + \mathbf{y}_B^2 + \frac{1}{A_B} (D_B + F_B - 2(B_B \mathbf{x}_B + C_B \mathbf{y}_B)) \quad (\text{B.186})$$

$A_B = \sum_{i=1}^{n_{\text{pix}}(B)} 1$	$B_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{x}_i$	$C_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{y}_i$
$D_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{x}_i^2$	$E_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{x}_i \mathbf{y}_i$	$F_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{y}_i^2$
$G_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{x}_i^3$	$H_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{x}_i^2 \mathbf{y}_i$	$I_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{x}_i \mathbf{y}_i^2$
$J_B = \sum_{i=1}^{n_{\text{pix}}(B)} \mathbf{y}_i^3$		
$\mathbf{a}_B = 2(A_B D_B - B_B^2)$	$\mathbf{d}_B = A_B(G_B + I_B) - B_B(D_B + F_B)$	
$\mathbf{b}_B = 2(A_B F_B - C_B^2)$	$\mathbf{e}_B = A_B(H_B + J_B) - C_B(D_B + F_B)$	
$\mathbf{c}_B = 2(A_B E_B - B_B C_B)$	$\mathbf{f}_B = \mathbf{a}_B \mathbf{b}_B - \mathbf{c}_B^2$	

Table B.3: Ball localization coefficients

Bibliography

- [1] ROBOCUP. *Overview: What is RoboCup*,
<http://www.robocup.org/overview/21.html>, accessed 24.01.2005
- [2] MSL TECHNICAL COMMITTEE 1997-1999. *Middle Size Robot League - Rules and Regulations for 1999*,
<http://www.ida.liu.se/ext/RoboCup-99/>, accessed 5.12.1999
- [3] MSL TECHNICAL COMMITTEE 1997-2004. *Middle Size Robot League - Rules and Regulations for 2004, Draft Version pre-8.1*,
<http://www.er.ams.eng.osaka-u.ac.jp/rc2004msl/msl-rules-2004.pdf>, accessed 24.01.2005
- [4] ROBOCUP. *Official Homepage*,
<http://www.robocup.org>, accessed 24.01.2005
- [5] PIONEER MOBILE ROBOT PLATFORM. *Official Homepage*,
<http://www.activrobots.com/>, <http://www.rwii.com>, accessed 24.01.2005
- [6] NOMAD SUPER SCOUT MOBILE ROBOT PLATFORM. *Official Homepage*,
<http://www.robots.com/nsuperscout.htm>, accessed 5.12.1999
- [7] S. CORADESCHI, T. BALCH, G. KRAETZSCHMAR, P. STONE (EDITORS). RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6,
<http://www.ep.liu.se/ea/cis/1999/006/>, accessed 24.01.2005
- [8] T. BANDLOW, R. HANEK, M. KLUPSCH, THORSTEN SCHMITT. *Agilo RoboCuppers: RoboCup Team Description, AGILO*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4

- (1999):6, pages 90-97,
<http://www.ep.liu.se/ea/cis/1999/006/16/>, accessed 24.01.2005
- [9] D. NARDI, G. ADORNI, A. BONARINI, R. CASSINIS, A. CHELLA, G. CLEMENTE, E. PAGELLO, M. PIAGGIO. *Azzurra Robot Team - ART*, *ART*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 99-106,
<http://www.ep.liu.se/ea/cis/1999/006/17/>, accessed 24.01.2005
- [10] F. YONG, B. NG, K. LOH, E. ONG, K. TEO, K. LOH. *Alpha++*, *BengKiat*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 107-112,
<http://www.ep.liu.se/ea/cis/1999/006/18/>, accessed 24.01.2005
- [11] B. NEBEL, J.-S. GUTMANN, W. HATZACK. *The CS Freiburg '99 Team*, *CS-Freiburg*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 113-117,
<http://www.ep.liu.se/ea/cis/1999/006/19/>, accessed 24.01.2005
- [12] M. JAMZAD. *CS-Sharif ROCS99 team in middle-sized robots league*, *CS-Sharif*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 118-126,
<http://www.ep.liu.se/ea/cis/1999/006/20/>, accessed 24.01.2005
- [13] P. JONKER, E. CORTEN, N. POLYKARPOVA, F. GROEN. *The Dutch F2000 RoboCup Team*, *Dutch-Team*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 127-133,
<http://www.ep.liu.se/ea/cis/1999/006/21/>, accessed 24.01.2005
- [14] A. BREDENFELD, T. CRISTALLER, W. GÖHRING, H. GÜNTHER, H. JAEGER, H.-U. KOBIALKA, P.-G. PLÖGER, P. SCHÖLL, A. SIEGBERG, A. STREIT, C. VERBEEK, J. WILBERG. *Behavior engineering with "dual dynamics" models and design tools*, *GMD-Robots*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 134-145,
<http://www.ep.liu.se/ea/cis/1999/006/22/>, accessed 24.01.2005

- [15] W.-M. SHEN, B. SALEMI, J. MODI, R. ADOBBATI. *DREAMTEAM 99: Team Description Paper, ISI-DreamTeam99*, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 146-149, <http://www.ep.liu.se/ea/cis/1999/006/23/>, accessed 24.01.2005
- [16] R. VENTURA, P. APARICIO, P. LIMA, L. CUSTODIO. *ISocRob – Intelligent Society of Robots, ISocRob, RoboCup-99 Team Descriptions. Small and Middle Leagues*, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 150-159, <http://www.ep.liu.se/ea/cis/1999/006/24/>, accessed 24.01.2005
- [17] S. ENOKIDA, M. FUKUDA, T. OHASHI, T. YOSHIDA, T. EJIMA. *KIRC Team, KIRC, RoboCup-99 Team Descriptions. Small and Middle Leagues*, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 160-162, <http://www.ep.liu.se/ea/cis/1999/006/25/>, accessed 24.01.2005
- [18] K. DEMURA, K. MIWA, Y. ASANO, H. IGARASHI, D. ISHIHARA. *Matto: Towards a Pass-Based Tactics, Matto, RoboCup-99 Team Descriptions. Small and Middle Leagues*, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 163-169, <http://www.ep.liu.se/ea/cis/1999/006/26/>, accessed 24.01.2005
- [19] T. NAKAMURA, K. TERADA, H. TAKEDA, A. EBINA, H. FUJIWARA. *Team Description of the RoboCup-NAIST, NAIST, RoboCup-99 Team Descriptions. Small and Middle Leagues*, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 170-174, <http://www.ep.liu.se/ea/cis/1999/006/27/>, accessed 24.01.2005
- [20] A. RIBEIRO, C. MACHADO, I. COSTA, S. SAMPAIO. *Patriarcas/MINHO Football Team, Patriarcas, RoboCup-99 Team Descriptions. Small and Middle Leagues*, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 175-180, <http://www.ep.liu.se/ea/cis/1999/006/28/>, accessed 24.01.2005
- [21] J. BRUSEY, A. JENNINGS, M. MAKIES, C. KEEN, A. KENDALL, L. PADGHAM, D. SINGH. *RMIT Robocup Team, RMIT, RoboCup-99 Team Descriptions. Small and Middle Leagues*, Linköping Electronic

- Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 181-188,
<http://www.ep.liu.se/ea/cis/1999/006/29/>, accessed 24.01.2005
- [22] C. TEOH. *Team Description of SP-Wisely, SP-Wisely*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 189-194,
<http://www.ep.liu.se/ea/cis/1999/006/30/>, accessed 24.01.2005
- [23] N. OSWALD, M. BECHT, T. BUCHHEIM, G. HETZEL, G. KINDERMANN, R. LAFRENTZ, M. MUSCHOLL, M. SCHANZ, M. SCHULÉ, P. LEVI. *CoPS-Team Description, Stuttgart-CoPS*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 195-199,
<http://www.ep.liu.se/ea/cis/1999/006/31/>, accessed 24.01.2005
- [24] M. PLAGGE, R. GÜNTHER, J. IHLENBURG, D. JUNG, A. ZELL. *The Attempto RoboCup Robot Team, Team-Tuebingen*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 200-209,
<http://www.ep.liu.se/ea/cis/1999/006/32/>, accessed 24.01.2005
- [25] S. SUZUKI, H. ISHIZUKA, H. KAWANISHI, M. YANASE, Y. TAKAHASHI, E. UCHIBE, M. ASADA. *The Team Description of Osaka University "Trackies-99", Trackies*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 210-210,
<http://www.ep.liu.se/ea/cis/1999/006/33/>, accessed 24.01.2005
- [26] S. SABLATNÖG, S. ENDERLE, M. DETTINGER, T. BOSS, M. LIVANI, M. DIETZ, J. GIEBEL, U. MEIS, H. FOLKERTS, A. NEUBECK, P. SCHÄFFER, M. RITTER, H. BRAXMEIER, D. MASCHKE, G. KRAETZSCHMAR, J. KAISER, G. PALM. *The Ulm Sparrows 99, Ulm-Sparrows*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 211-216,
<http://www.ep.liu.se/ea/cis/1999/006/34/>, accessed 24.01.2005

- [27] P. COSTA, A. MOREIRA, A. SOUSA, P. MARQUES, P. COSTA, A. MATOS. *5dpo-2000 Team description, 5dpo*, RoboCup-99 Team Descriptions. Small and Middle Leagues, Linköping Electronic Articles in Computer and Information Science, ISSN 1401-9841; Vol. 4 (1999):6, pages 217-220,
<http://www.ep.liu.se/ea/cis/1999/006/35/>, accessed 24.01.2005
- [28] MSL ORGANIZATION COMMITTEE 2004. *Questionnaire Summary*,
<http://www.er.ams.eng.osaka-u.ac.jp/rc2004msl/index.cgi?page=Questionnaire+Summary>, accessed 24.01.2005
- [29] OSAKA UNIVERSITY "TRACKIES 2004". *Y. Takahashi, S. Takamuku, M. Yuba, M. Asada*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
<http://www.er.ams.eng.osaka-u.ac.jp/robocup/trackies/team04/trackies04.pdf>, accessed 13.03.2006
- [30] P. LIMA, L. CUSTÓDIO, P. PINHEIRO, H. COSTELHA, G. NETO, V. PIRES, M. ARROZ, B. VECHT. *ISocRob 2004: Team Description Paper*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
<http://socrob.isr.ist.utl.pt/tdps-tfcs/isocrob-tdp2004.pdf>, accessed 13.03.2006
- [31] ALLEMANIACS 2004 TEAM DESCRIPTION. *A. Ferrein, C. Fritz, G. Lakemeyer*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
<http://robocup.rwth-aachen.de/allemaniacs2004tdp.ps.gz>, accessed 13.03.2006
- [32] THE ATTEMPTO TÜBINGEN ROBOT SOCCER TEAM. *P. Heinemann, A. Treptow, A. Zell*, RoboCup-2004: Robot Soccer World Cup VIII, Lecture Notes in Computer Science, Volume 3276, (CD-Supplement), Springer, 2005
<http://www-ra.informatik.uni-tuebingen.de/forschung/robocup/publications/heinemann05attempto.pdf>, accessed 13.03.2006
- [33] AGILO ROBOCUPPERS 2004. *F. Stulp, A. Kirsch, S. Gedikli, M. Beetz*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
<http://www9.in.tum.de/publications/2004/ROBOCUP-SYMP-04-AGILO-TDP-Stulp-etal.pdf>, accessed 13.03.2006

- [34] MOSTLY HARMLESS TEAM DESCRIPTION 2004. *G. Steinbauer, C. Deutsch, G. Fraser, M. Hagler, A. Mühlenfeld, S. Richter, G. Wöber, J. Wolf*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
http://www.robocup.tugraz.at/downloads/TDP_04.pdf, accessed 13.03.2006
- [35] H. FUJII, Y. OHDE, M. KATO, F. OTSUKA, N. SEMA, M. KAWASHIMA, E. SUGIYAMA, S. NIIZUMA, K. YOSHIDA. *EIGEN Team Description*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
<http://www.yoshida.sd.keio.ac.jp/~robocup/images/tdp2004.pdf>, accessed 13.03.2006
- [36] ISEP/PORTO ROBOTIC SOCCER TEAM FOR ROBOCUP 2004: THE ROAD TO REAL COOPERATIVE PLAY.. *J. M. Almeida, A. Martins, E. P. Silva, J. Baptista, A. Patacho, L. Lima, V. Cerqueira, R. Picas, N. Dias, A. Dias, N. Matos, H. Silva, P. Santos*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
http://www.lsa.isep.ipp.pt/publications/ISepPortoTDP04_submitted.pdf, accessed 13.03.2006
- [37] 5DPO-2000 TEAM DESCRIPTION FOR YEAR 2004. *A. P. Moreira, P. Costa, A. Sousa, L. P. Reis*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
http://paginas.fe.up.pt/~robosoc/sci/5dpo_F2000_TDP_2004.pdf, accessed 13.03.2006
- [38] JIAOLONG2004 TEAM DESCRIPTION. *Q. Cao, W. Chen, Y. Huang, J. Wang, J. Jia, Z. Luo, Z. Zhang, L. Pan, Z. Qian, Z. Fu, Y. Sun, S. Miao, X. Chen, F. Zhang, L. Yu, Y. Zhang*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
<http://www.robot.sjtu.edu.cn/robocup/2004/doc/JiaoLong2004-TDP.pdf>, accessed 13.03.2006
- [39] A CONSTRUCTIVE FEATURE DETECTION APPROACH FOR ROBOTIC VISION. *F. von Hundelshausen, M. Schreiber, R. Rojas*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
<http://robocup.mi.fu-berlin.de/docs/robocup04.pdf>, accessed 13.03.2006
- [40] TRACKING REGIONS. *F. von Hundelshausen, M. Schreiber, R. Rojas*, Proceedings of the International RoboCup Symposium, Lisbon,

- July 2004,
<http://robocup.mi.fu-berlin.de/docs/RegionTrackingRoboCup03.pdf>, accessed 13.03.2006
- [41] T. BUCHHEIM, G. KINDERMANN, R. LAFRENTZ, H. RAJAIE, M. SCHANZ, F. SCHREIBER, O. ZWEIGLE, P. LEVI. *Team Description Paper 2004 CoPS Stuttgart*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
<ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstr1.ustuttgart.fi/INPROC-2004-71/INPROC-2004-71.pdf>, accessed 13.03.2006
- [42] THE CLOCKWORK ORANGE TEAM 2004. *P. Jonker, B. Terwijn, B. van Driel*, Proceedings of the International RoboCup Symposium, Lisbon, July 2004,
http://www.ph.tn.tudelft.nl/People/albert/papers/clockwork_orange_2004.pdf, accessed 13.03.2006
- [43] R. NORDMANN. *Maschinenelemente und Mechatronik, Skript Block A*, ISBN 3-8265-8051-6, Shaker Verlag, Aachen (2001)
- [44] R. NORDMANN. *Mechatronische Systeme im Maschinenbau I*, ISBN 3-8265-8957-2, Shaker Verlag, Aachen (2001)
- [45] H. T. LA. *Omnidirectional Vehicle*, United States Patent, US4237990, Dec. 9, 1980
- [46] W. HAUGER, W. SCHNELL, D. GROSS. *Technische Mechanik 1, Statik, vierte Auflage* ISBN 3-540-55286-3, Springer Verlag, Berlin (2001)
- [47] W. HAUGER, W. SCHNELL, D. GROSS. *Technische Mechanik 3, Kinetik, vierte Auflage* ISBN 3-540-56323-7, Springer Verlag, Berlin (2001)
- [48] T. R. KANE, D. A. LEVINSON. *Dynamics: Theory and Applications* ISBN 0-07-037846-0, McGraw-Hill, Inc., New York (1985)
- [49] PROF. DR.-ING. B. BREUER. *Kraftfahrzeuge I, Skriptum zur Vorlesung* Darmstadt: TU Darmstadt (1997)
- [50] PROF. DR.-ING. B. BREUER. *Kraftfahrzeuge II, Skriptum zur Vorlesung* Darmstadt: TU Darmstadt (2001)

- [51] MAXON MOTOR. *maxon motor control, 4-Q-EC Servoamplifier DES 50/5* maxon motor ag, Sachseln, CH (2002)
<http://www.maxonmotor.com>, accessed 24.01.2005
- [52] B. C. KUO. *Automatic Control Systems, seventh edition*, ISBN 0-13-304-759-8, Prentice-Hall, Inc., Englewood Cliffs (1995)
- [53] S. HARGREAVES. *Allegro, A game programming library*, 1994/97
<http://www.delorie.com/djgpp/>, accessed 24.01.2005