

Efficient Graph-Based V2V Free Space Fusion

Stefan Luthardt*, Chao Han*, Volker Willert*, Matthias Schreier†

Abstract—A necessary prerequisite for future driver assistance systems as well as automated driving is a suitable and accurate representation of the environment around the vehicle with a sufficient range. To extend the range of the environment representation, sharing the environment detections of multiple vehicles via vehicle-to-vehicle (V2V) communication is a promising approach. In this paper, we present a method to fuse shared free space detections from multiple vehicles. The detections are represented as Parametric Free Space (PFS) maps, which are especially suitable for real-time radio V2V-transmission due to their compactness.

A graph-based algorithm to fuse PFS maps is proposed that solves possible contradictions between the maps and incorporates the maps' uncertainty attributes. By solely operating on the contour, the fusion can be carried out by a simple path search in a fusion graph that is constructed from the maps. This results in an efficient method that finds the fusion result within few iterations.

To account for possible errors in the relative poses between the PFS maps, we furthermore present an adapted Iterative Closest Point (ICP) matching to align the maps before the fusion. Therein we employ a modified soft-assign scheme for robust outlier rejection, and incorporate the PFS maps' boundary orientation to improve the matching process. We show the capabilities of our method by presenting results on real test drive data.

I. INTRODUCTION

A key aspect for Advanced Driver Assistance Systems (ADAS) is the environment perception. ADAS available today have already contributed to road safety. However, systems for future intelligent vehicles will demand a more comprehensive perception of the environment. Most of the present ADAS have their own environment perception solely used for their specific task, e. g. blind spot detection. In future vehicles, the different assistance systems will have to be integrated cooperatively and rely on a common base of environment perception. This approach was investigated within the Proreta 3 project [1], which included the development of an appropriate commonly usable environment representation that describes static scenery and relevant dynamic objects. Such representation was proposed in [2]–[4] in form of a Parametric Free Space (PFS) map combined with a dynamic object map. In this paper, we focus on the PFS map, which describes the static environment around the vehicle by the boundary of traversable free space.

The free space represented in the PFS map has a limited range due to the physical limitations of the sensors it is generated from. However, for ADAS functions like path planning or collision avoidance a high range of free space perception is desirable. This paper explores an approach to

enhance the free space range by fusing the perceived free space from multiple vehicles that interchange environment data via vehicle-to-vehicle (V2V) transmission. The PFS map description is especially favorable for this task. It is highly compact and therefore suitable for high frequency radio exchange with a small demand to the transmission bandwidth. Furthermore, boundaries to unknown environment are specifically labeled, which is very useful in the fusion process, since it is clear where additional free space is allowed to be attached.

In Sec. II we show that the fusion of PFS maps can be conducted by converting the maps to a *fusion graph*, and searching for specific cycles within this graph following a simple set of rules. These rules are inferred from the semantic of the PFS map and allow an efficient logic-based fusion process.

In order to fuse two PFS maps, it is necessary to know their relative pose. This pose can be obtained from the pose measurements of the vehicles that have generated the maps. However, in many real world applications these poses can probably not be measured with sufficient accuracy. To account for this problem, we use a matching between the PFS maps to estimate the true relative pose between the maps. The matching process is described in Sec. III and utilizes a modified version of an advanced point matching method presented by [5], which we enhance to incorporate the orientation of free space boundaries.

For both methods we will present results on data from real driving scenarios to show their capabilities.

A. Related Work

For the fusion of environment representations from multiple agents, like cars or robots, there is a broad variety of approaches available. Usually these fusion approaches can be split in two steps: first an alignment of the maps, and second a merging of the aligned maps. Both parts highly depend on the environment representations the agents use. An overview of commonly used environment representations is given in [4] and [6].

One possible representation are *occupancy grid maps* (OGM), that state the occupancy probability for each cell in a rectangular grid of equally sized 2D cells. In recent years they have become quite popular, since they are easy to interpret and can be generated from various sensors, e. g. stereo cameras, laser range finders or radars [7], [8]. For this specific map type, a variety of fusion methods has been proposed, which mainly differ in the alignment step. The alignment can be performed by matching keypoints or contours extracted from the OGM [9], [10], cross correlation

*Technische Universität Darmstadt, Control Methods and Robotics

†Continental AG

of image spectra of the OGM [10], [11], or minimizing a suitable objective function [6]. In all methods, new matching features have to be extracted first, or the whole OGM has to be processed several times, which causes high computational costs.

To overcome the issues of OGMs, contour-based representations were recently proposed. They capture the essential information of a grid map, i.e. the location of free space. This is achieved by describing the outer contour of this free space with geometric shapes that are defined by a small number of coordinates. In our approach we use the PFS map introduced by Schreier et al. in 2012 [2], which was further developed in the following years [3], [4]. More details of this representations will be explained in Sec. I-B. A similar approach to PFS maps was later proposed by Kubertschak et al. in 2014 [12], where free space boundaries and obstacles are described by polygons and polylines called fences.

In this paper, we present a method to fuse free space in contour-based representations. The compact form of this representations eases both steps of the fusion. In the alignment step, the contours themselves can be utilized as characteristic features. The subsequent merging can also be performed in a straight forward manner, since only the contours have to be considered and not the whole free space area. In our approach, this is done in a graph-based algorithm described in Sec. II.

Utilizing graphs for map fusions problems is a common technique. So called pose graphs offer a possibility to model the agents' poses and observations, especially in SLAM-like problems [13]–[15]. Map regions and their neighborhood relations can also be described in a graph, thereby providing a more mathematical view to region merging problems like in [16]. However, in our approach the graph represents the free space contour and its attributes, as opposed to representing agent poses or regions like in existing methods known to us.

B. Parametric Free Space Maps

In this section, we present the basic principles of the PFS map, which we employ as environment representation. Further details are published in [2], [3] and [4].

Two examples of PFS maps are presented in Fig. 1. A PFS map describes relevant traversable free space, i.e. areas that the vehicle can reach from its current position and are in principle traversable with regard to the vehicles dimensions. This free space is described by its boundary, which is modeled as a closed curve consisting of B-Splines¹ and the shape of the boundary is described by the control points $\{\mathbf{q}_i\}$ of these splines.

The boundary of the free space is not always induced by barriers or obstacles. A boundary may also occur due to the limited detection range of the employed sensors. Since such boundary segments should be treated differently, they are labeled with the specific attribute “unknown environment boundary”. Contrarily, boundary segments induced by real obstacles are labeled as “obstacle boundary”.

¹Refer to [2] or [17] for further details on B-Splines.

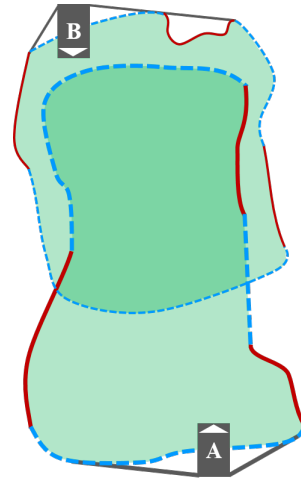


Fig. 1. Example of two overlapping PFS maps enclosing sensed free space (green areas). The boundary of the free space sensed by the ego-vehicle A is depicted with thick lines, obstacle boundaries in red (—) and unknown environment boundaries in blue (---). The oncoming other vehicle B also senses free space (thin lines), likewise with obstacle (—) and unknown environment boundaries (---).

The task of our proposed fusion algorithm is to infer the combined free space from both PFS maps (all green areas), thus extending the range of known free space for the ego-vehicle A. The fusion process for this example maps is shown in the following figures 2, 3 and 4.

In some cases, the free space contains “holes”, i.e. static objects that should not be driven over, but can be driven around, e.g. a traffic island. These holes are represented by geometric primitives, i.e. rectangles or circles.

In conclusion, the PFS map consists of the set of spline control points $\{\mathbf{q}_i\}$, with corresponding binary attribute labels $\{l_i\}$, and a set of geometric primitives $\{\mathbf{x}_{G,j}\}$. If 16 bit-floats are used, this leads to a size of about 330 Bytes² per PFS map or a bandwidth demand of 5.2 kBytes/s for a transmission with 16 Hz.

In [2] a method was introduced to generate a PFS map representation from an occupancy grid map. This was done by applying several image filters to the grid map to obtain a binary free space image, tracing its boundary and temporally filtering the boundary with a Kalman filter. However, the PFS map representation can also be obtained in other ways.

II. GRAPH-BASED FUSION OF PFS MAPS

A. Fusion Task and Method Overview

In this section, we describe a graph-based fusion method for PFS maps. The starting point of this method are two overlapping PFS maps with known relative pose like shown in Fig. 1. If the relative pose between the maps is not available with sufficient accuracy, the matching method described in Sec. III may be applied beforehand.

The target of the fusion is to infer the largest possible free space from the given input maps with the following requirements:

- The result of the fusion has to be a PFS map, since we want to keep the advantages of this representation, i.e.

²For this example computation the realistic case of 70 control points and five inner objects is assumed (c.f. computation in [3]).

the result of the fusion has to be a closed boundary with semantic attributes and potentially inner objects.

- The fusion method should respect the semantic attribution of the free space boundary, i. e. “obstacle boundary” or “unknown environment boundary” (see Sec. I-B).
- If there is a contradiction between the free space maps, the map from the ego-vehicle takes precedence over the map from the other vehicle. We take this decision, because we assume that the ego vehicle’s sensor-setup is better fault-monitored and therefore more trustworthy. Please note that this precedence rule is only one possible choice. Depending on the fusion scenario and application, another precedence rule may be reasonable.

To fulfill the requirements listed above, we developed a graph-based method for fusing two intersecting PFS maps. The first step of this method is the construction of a graph from the intersecting maps, denoted as *fusion graph*. The fusion graph contains all relevant topological structure needed for the fusion task and hides unnecessary details like the precise course of the map boundaries. This simplifies the fusion task and thereby reduces the computational cost. In the following sections we first introduce the fusion graph representation (Sec. II-B) before explaining the fusion process that consist of the following steps:

- 1) Fusion graph construction from PFS maps (Sec. II-C),
- 2) Fusion graph conciliation (Sec. II-D),
- 3) Path search in the fusion graph (Sec. II-E), and
- 4) Conversion of the result into a PFS map (Sec. II-F).

Finally we conclude with a fusion result for real world data in Sec. II-G.

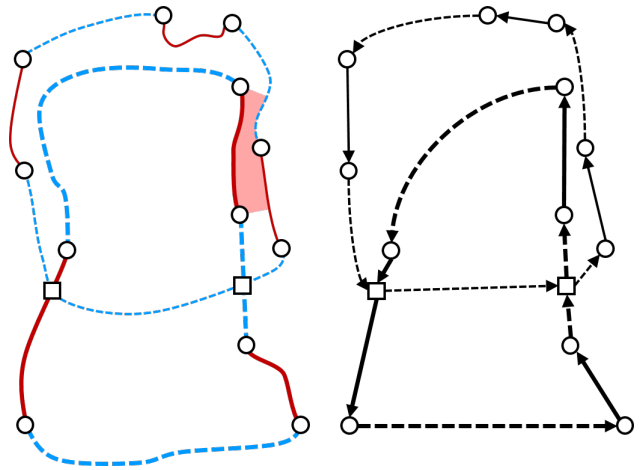
B. Fusion Graph Representation

The *fusion graph* is a directed graph $F = (V, E)$, consisting of vertices V and directed edges E . Both types of elements are described in detail below. Furthermore we introduce some functions to traverse on the graph. The basics of graph theory, which are the foundation of the fusion graph, can be found in [18]. To illustrate the graph representation, Fig. 2(b) depicts an example fusion graph corresponding to the PFS maps in Fig. 2(a).

1) *Edges*: The edges E of the fusion graph represent the boundaries of the PFS maps. The boundaries include the outer boundary of the free space and the boundaries of inner obstacles. Please note that neither length nor shape of the boundaries are represented in the edges.

The direction of the edges indicates the boundary side with free space. Following the direction of the edge, the area on the left-hand side is always free space. Contrarily, the area on the right-hand side is either an obstacle or unknown area.

Furthermore the edges carry two attributes. They are either an “obstacle boundary” (continuous line) or “unknown environment boundary” (dashed line). According to their data origin they are additionally marked as “from ego-vehicle” (thick line) or “from other vehicle” (thin line). Using this attribution, we define the set $E^* \subset E$ containing all boundaries from the ego-vehicle and the set $O^* \subset E^*$ with all obstacle boundaries from the ego-vehicle.



(a) PFS maps overlaid with fusion graph vertices. Conflict area in light red. (b) Fusion graph for the overlapping PFS maps in (a).

Fig. 2. Example from Fig. 1 continued: fusion graph construction. For an explanation of the vertices and edges see Sec. II-B.

2) *Vertices*: There are three type of vertices. The *switch vertex* (\circ) represents a switch in a semantic attribute of the free space boundary, i. e. it is placed at the connection of a boundary segment with the attribute “obstacle boundary” to a segment with the attribute “unknown environment boundary” or vice versa. The *intersection vertex* (\square) represents an intersection between boundaries from two different maps. All intersection vertices belong to the subset $X \subset V$. The third type is the *link vertex* (\triangle). It is inserted in the graph to make it “conciliated”, which will be explained in Sec. II-D.

3) *Traversing functions*: We will use the following graph functions in our algorithms:

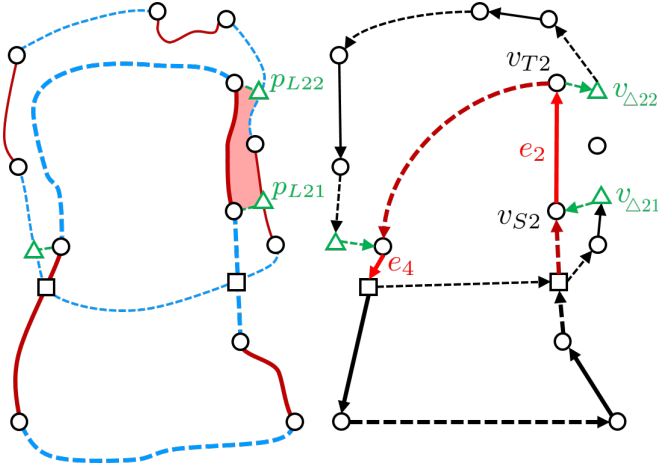
- $e_L = \varepsilon_L(v)$ returns the leftmost³ outgoing edge e_L of the vertex v . If v has no outgoing edge, it returns the empty set \emptyset .
- $e_R = \varepsilon_R(v)$ returns the rightmost³ outgoing edge e_R of the vertex v .
- $v_T = \varphi(e)$ returns the target vertex v_T of the directed edge $e = (v_S, v_T)$.
- $v_S = \varphi^{-1}(e)$ returns the source vertex v_S of the directed edge $e = (v_S, v_T)$.

C. Fusion Graph Construction

With the basic elements from the previous section, the fusion graph can be easily constructed from two intersecting PFS maps by applying the following procedures:

- 1) Insert a *switch vertex* (\circ) at every point where the boundary attribute changes from “obstacle boundary” to “unknown environment boundary” or vice versa.
- 2) Insert an *intersection vertex* (\square) at every point where two boundaries intersect.
- 3) Connect the vertices with edges considering the attributes of the corresponding boundary segment and the

³“left” and “right” in these definitions have to be interpreted in relation to the direction of the incoming and outgoing edges.



(a) Corresponding PFS map for the conciliated fusion graph. (b) Conciliated fusion graph.

Fig. 3. Example from Fig. 2 continued: All red edges in (b) are checked for conflicts by the conciliation algorithm (Algorithm 1). For the edges e_2 and e_4 (light red) a conciliation is necessary. To conciliate the conflict induced by e_2 , the link vertices $v_{\Delta 21}$ and $v_{\Delta 22}$ are placed on the outer boundary and linked with the source vertex v_{S2} respectively target vertex v_{T2} of e_2 with new edges (green arrows).

proper edge direction with regard to the location of free space area.

The construction process can be comprehended in Fig. 2(b), where the fusion graph is constructed for the example PFS maps shown in Fig. 2(a).

D. Fusion Graph Conciliation

In Sec. II-A we decided that the ego-vehicle map takes precedence over the map from the other vehicle. Thus we absolutely trust in all known obstacle boundaries originating from the ego-vehicle, regardless of the free space statements from the other vehicle. Consequentially all the obstacle boundaries from the ego-vehicle (subset O^* , thick continuous lines) have to be part of the fused boundary. This becomes problematic if such a segment lies inside of the other-vehicle's free space area. An example of this situation is shown in Fig. 2(a) with the conflict area highlighted in light red. In this case, we have two conflicting claims: The ego-vehicle map claims the area behind the boundary to be occupied, while the other-vehicle map claims it to be free.

We deal with such a situation by assuming that the area behind the obstacle boundary is unknown. Therefore we have to insert new unknown environment boundaries surrounding this area. The result of this correction process for our example is shown in Fig. 3. New unknown environment boundaries are added at the source vertex v_{S2} and the target vertex v_{T2} of the O^* -boundary e_2 . They are connected to link vertices (v_{Δ}) that are placed on the outer boundary. Since the conflicting claims are "conciliated" by this correction, we call the fusion graph obtained after this correction a *conciliated fusion graph*.

In general, such conflicts can be found and solved by Algorithm 1. Conflicts can be found by traversing all segments with ego-vehicle edges ($e \in E^*$) that lie inside the

other vehicles free space (all red edges in Fig. 3(b)). Since free space lies always on the left-hand side of the boundary, such segments have to start at an intersection vertex v_{\square} as the left outgoing edge. Consequently we have to search for $v \in X$ where $\mathcal{E}_L(v) \in E^*$. The found vertices v_{\square} are the start of potentially critical segments.

All these segments have to be checked for possible conflicts. If an edge e on a critical segment is an obstacle boundary, i. e. $e \in O^*$, it will cause a conflict (light red edges). A found conflict is then conciliated by adding new links (green edges) between the edge e and the outer boundary.

The new link vertices v_{Δ} on the outer boundary are created by the function $(v_{\Delta}, e_B) \leftarrow \delta_B(v)$. This function determines the closest point p_L on the outer boundary of the original PFS map, measured from the vertex v . A new link vertex v_{Δ} is then created at p_L (c. f. example in Fig. 3). Furthermore, the edge e_B is determined that corresponds to the outer boundary section, p_L is located on. The edge e_B is shortened such that the outer boundary becomes connected with v_{Δ} . This whole conciliation procedure is applied to every O^* -edge found in the investigated critical segment.

The result of Algorithm 1 is a conciliated fusion graph, where each O^* -edge has become a part of the outer boundary. The conciliated fusion graph for our example is depicted in Fig. 3(b). In the example, the conciliation process has to be applied to the edge e_2 and to the edge e_4 .

It should be emphasized that after the conciliation process, the specific position of the vertices is no longer relevant for the fusion process, as long as the basic topology is maintained. This is equivalent to prohibiting the displacement of vertices over edges.

E. Path Search in the Conciliated Fusion Graph

If the fusion problem is represented in a conciliated fusion graph, the actual fusion can be performed by matters of a

Algorithm 1 Conciliation of the fusion graph

```

Input:  $F = (V, E)$  // fusion graph
for  $\forall v_{\square} \in \{v \mid v \in X \wedge \mathcal{E}_L(v) \in E^*\}$  do
   $e \leftarrow \mathcal{E}_L(v_{\square})$  // start with outgoing  $E^*$ -edge of vertex  $v_{\square}$ 
  repeat // check critical segment for conflicts
     $v_S \leftarrow \varphi^{-1}(e)$  // source node of  $e$ 
     $v_T \leftarrow \varphi(e)$  // target node of  $e$ 
    if  $e \in O^* \wedge v_S \notin X$  then
      // insert link to source of  $O^*$ -edge  $e$ 
       $(v_{\Delta}, e_B) \leftarrow \delta_B(v_S)$ 
       $V \leftarrow V \cup v_{\Delta}$ 
       $E \leftarrow E \setminus e_B \cup (\varphi^{-1}(e_B), v_{\Delta}) \cup (v_{\Delta}, v_S)$ 
    end if
    if  $e \in O^* \wedge v_T \notin X$  then
      // insert link from target of  $O^*$ -edge  $e$ 
       $(v_{\Delta}, e_B) \leftarrow \delta_B(v_T)$ 
       $V \leftarrow V \cup v_{\Delta}$ 
       $E \leftarrow E \setminus e_B \cup (v_T, v_{\Delta}) \cup (v_{\Delta}, \varphi(e_B))$ 
    end if
     $e \leftarrow \mathcal{E}_L(v_T)$  // repeat with the next edge on the segment
  until  $v_T \in O^*$  // end of critical segment reached
end for
Output:  $F_C = (V, E)$  // conciliated fusion graph

```

simple path search. The rules for this search can be deduced directly from the requirements stated in Sec. II-A. We search for a PFS map representation, i. e. a set of closed boundaries that enclose free space. Therefore, the path we are searching for, has to be a closed path, i. e. a *cycle* [18, Chapter 1]. To find the largest possible area, we always have to take the rightmost outgoing edge of a vertex. Since free space is always located on the left-hand side, turning right ensures the largest possible free space area is enclosed. We also decided that the ego-vehicle map takes precedence, which translates to the rule: Always proceed on a O^* -edges if one is available. This rule has higher priority than the rule to follow the rightmost outgoing edge.

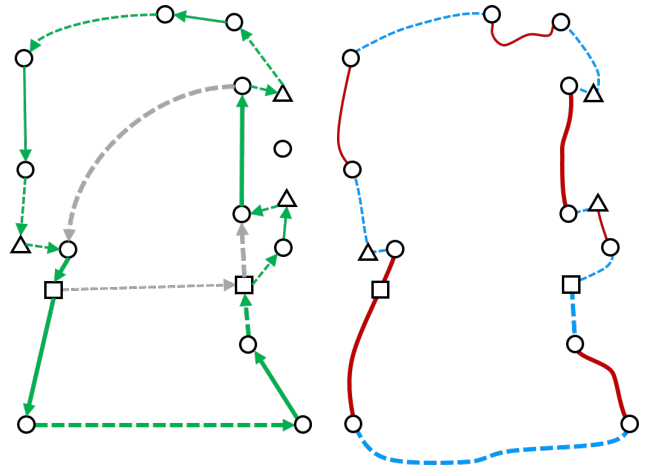
All this rules are incorporated into the fusion algorithm presented in Algorithm 2. This algorithm finds boundaries by starting at a random edge and following a path according to the rules until a complete cycle is found. If a cycle is found, it is removed from the graph and the search procedure starts all over again. It is also possible that a dead end is reached, i. e. a vertex with no outgoing edge. In this case, the found path is an irrelevant boundary that lies either on the inside or outside of an already found valid free space and is therefore removed.

Fig. 4(a) shows the result of the graph fusion for our example from Fig. 3. Independent from the start edge, the algorithm finds in the very first iteration of the path search the cycle highlighted in green. This cycle represents the outer boundary of the fused free space. In the two consecutive iterations the two remaining irrelevant paths (gray edges) are removed from the fusion graph and the algorithm terminates.

In general, this search algorithm terminates rather quickly,

Algorithm 2 Map fusion by cycle search in the conciliated fusion graph

Input: $F_C = (V, E)$ // conciliated fusion graph
 $i \leftarrow 1$
repeat // search for cycles
 $e \leftarrow \text{random } e \in E$
 $B_i \leftarrow \emptyset$ // (ordered set)
loop
 if $e \in B_i$ **then** // complete cycle found
 $B_i \leftarrow B_i \setminus \{u \mid u \in B_i \wedge u \text{ before } e\}$
 $E \leftarrow E \setminus B_i$ // remove cycle from graph
 $i \leftarrow i + 1$
 break
 end if
 $B_i \leftarrow B_i \cup e$
 $v_T \leftarrow \varphi(e)$ // target vertex of e
 if $\varepsilon_L(v_T) = \emptyset$ **then**
 $E \leftarrow E \setminus B_i$ // dead end, remove path from graph
 break
 end if
 if $\varepsilon_L(v_T) \in O^*$ **then**
 $e \leftarrow \varphi(\varepsilon_L(v_T))$ // proceed on high priority edge
 else
 $e \leftarrow \varphi(\varepsilon_R(v_T))$ // proceed on rightmost edge
 end if
end loop
until $E = \emptyset$ // all edges processed
Output: $\{B_i\}$ // fused boundaries as graph cycles



(a) The path search in the conciliated fusion graph finds the cycle highlighted in green that forms the fused boundary B_1 . Irrelevant edges belonging to dead-end paths are colored in gray.
(b) Fused PFS map reconstructed from the fused boundary B_1 in (a).

Fig. 4. Example from Fig. 3 continued: map fusion.

since the algorithm reduces the graph with every iteration by some edges, and usually has to process only a small number of paths.

F. Conversion of the Fusion Result into a PFS Map

To obtain the fused PFS map, the fused boundaries $\{B_i\}$ found by Algorithm 2 have to be converted back to the spline boundary representation of a PFS map. Solving this task is straight forward if we store which spline control points \mathbf{q} belong to which edge e during the fusion graph construction. At the end of the fusion process, the list of spline control points for every boundary B_i can be created simply by concatenating the \mathbf{q} of all edges in the ordered set B_i . For inner objects the boundary can be transformed to geometric primitives to obtain a fused PFS map in the form established in Sec. I-B.

For our example, the fused map is shown in Fig. 4(b). The result of our graph based fusion method is a PFS map that represents the largest possible free space that can be inferred from the two input PFS maps.

G. Fusion Results on Real World Data

Figure 5 depicts the fusion process for two PFS maps from a real test drive in a suburban scenario. The maps have been aligned beforehand with the matching method described in Sec. III (c. f. Fig. 6(d)). The fusion result is shown in Fig. 5(d). As desired, the known free space for the ego-vehicle is enlarged by appending the sensed free space from the other vehicle at the upper left.

III. MAP ALIGNMENT UTILIZING ICP MATCHING

A. Alignment Task

When two PFS maps shall be fused, it is necessary to know their relative pose in terms of a 2D translation \mathbf{t} and

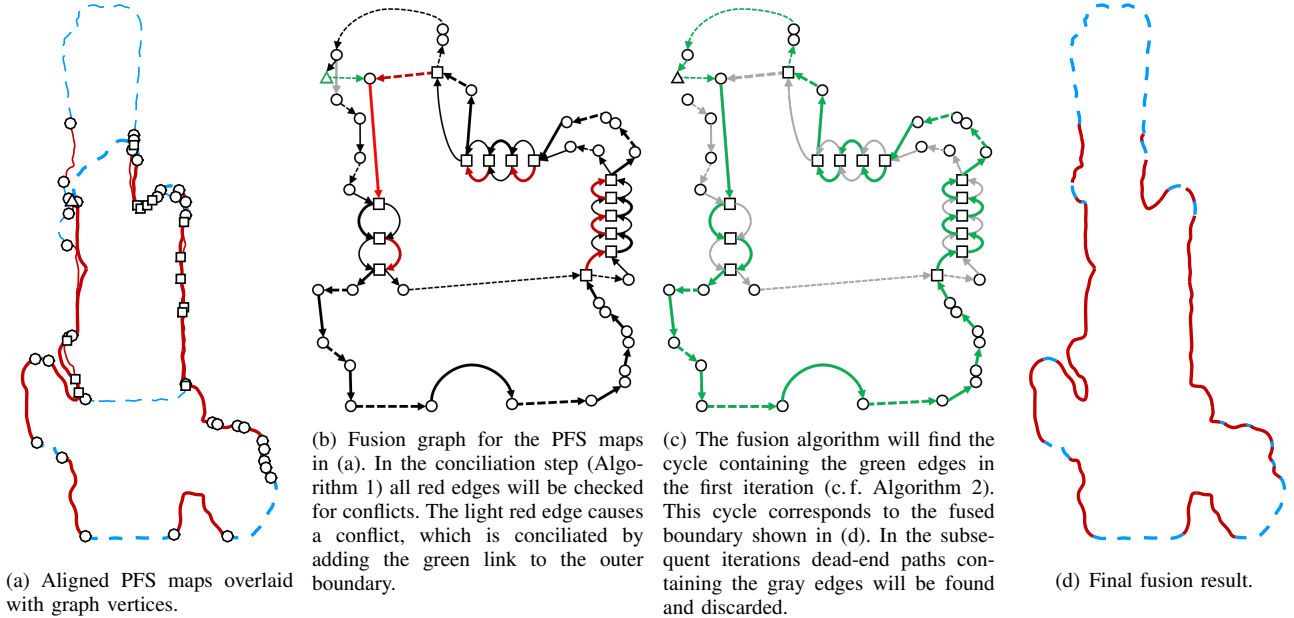


Fig. 5. Fusion example for two PFS maps from a real test drive.

a 2D rotation \mathbf{R} . In a real world application this relative pose can be estimated with an accuracy of a few meters by using global navigation satellite systems. This accuracy though would be insufficient for a reliable free space fusion. Therefore we apply a matching procedure before the fusion to reduce the pose bias and align the maps properly. Therein the obstacle boundaries contained in the PFS maps are used as matching subjects. In principle, the relative position estimation could also be done by matching some other environment characteristics, like visual landmarks or occupancy grid maps. However, there are some advantages when using the PFS map boundaries as matching subjects:

- The maps have low transmission bandwidth (see Sec. I-B) and can therefore be interchanged between vehicles with high frequency and low cost.
- Since the map has a very compact structure, only representing the boundaries, the matching is limited to a small number of points which reduces computational costs as opposed to matching dense representations.
- The free space boundaries are stable and recognizable structures.
- The orientations of the boundaries provide additional characteristics for the matching.

A main challenge in the matching process is an effective outlier rejection. Depending on the situation, the maps have an overlap of 20% to 50%. Hence there can be several boundaries that appear only in one of the maps and have to be recognized as outliers in the matching. To account for this, we employ an advanced variant of the ICP (Iterative Closest Point) matching algorithm taken from [5]. We modify it to fit to our rigid 2D matching problem and furthermore incorporate the boundary orientation information. This is described in the following Sec. III-B. In Sec. III-C we present some matching results for real world data.

B. ICP Map Matching with Orientation Aided Softassign and Deterministic Annealing

The B-spline control points \mathbf{q}_i of the PFS map boundaries are not suitable as matching subjects, since they are irregularly distributed along the boundaries. Therefore the first step of our matching process is an equidistant sampling of the obstacle boundaries in both PFS maps. We obtain the sample points $\mathbf{p}_{A,j}$ from the ego-vehicle's obstacle boundaries and the points $\mathbf{p}_{B,k}$ from the obstacle boundaries provided by the other vehicle. The aim of the matching process is to determine the transformation $(\mathbf{R}_{AB}, \mathbf{t}_{AB})$ from the other-vehicle's coordinate system to the ego-vehicle's coordinate system such that both maps are properly aligned at each other. This is achieved by minimizing the energy function $E(\mathbf{M}, \mathbf{R}_{AB}, \mathbf{t}_{AB})$ given by

$$\sum_{j=1}^J \sum_{k=1}^K m_{j,k} \|\mathbf{p}_{A,j} - (\mathbf{R}_{AB} \mathbf{p}_{B,k} + \mathbf{t}_{AB})\|^2 - \alpha \sum_{j=1}^J \sum_{k=1}^K m_{j,k} \quad (1)$$

subject to $\forall j \sum_{k=1}^K m_{j,k} \leq 1, \quad \forall k \sum_{j=1}^J m_{j,k} \leq 1$
and $\forall j, k \ m_{j,k} \in [0, 1]$.

Therein α is a distance tolerance parameter (c.f. [5]) and $m_{j,k}$ is the assignment value of point $\mathbf{p}_{A,j}$ to point $\mathbf{p}_{B,k}$. The matching matrix \mathbf{M} contains all the assignment values $m_{j,k}$ and thereby captures the mapping between the points.

1) *Softassign*: For a sensible handling of matching outliers, we adopt the softassign scheme from [5], which was originally introduced in [19]. It softens the assignment of points in the matching matrix \mathbf{M} by allowing values between 0 and 1. In this manner, the uncertainty in the assignment is considered and the influence of false assignments can be diminished. Furthermore, there are outlier elements $m_{j,K+1}$

respectively $m_{J+1,k}$ to mark a point as outlier, i. e. having no corresponding point in the other map. The assignment values $m_{j,k}$ are determined by

$$m_{j,k} = e^{-\beta Q_{j,k}}$$

with $Q_{j,k} = \frac{\partial E}{\partial m_{j,k}} = \|\mathbf{p}_{A,j} - (\mathbf{R}_{AB}\mathbf{p}_{B,k} + \mathbf{t}_{AB})\|^2 - \alpha.$ (2)

An iterative alternating normalization of the row and column sums of \mathbf{M} is applied afterwards to ensure the constraints in (1).

2) *Incorporating the Boundary Orientation:* As mentioned in Sec. III-A, we can utilize the orientation of a boundary as additional characteristic in the matching process. For each boundary point \mathbf{p}_j we determine the normal vector \mathbf{n}_j that is perpendicular to the boundary in this point and is pointing inside the free space. The boundary orientation in \mathbf{p}_j is defined as the ascent angle θ_j of this vector \mathbf{n}_j . To incorporate the orientation in the matching, $Q_{j,k}$ in (2) is replaced by

$$\tilde{Q}_{j,k} = Q_{j,k} + \gamma (\delta_{j,k} - \frac{\pi}{2}).$$
 (3)

The orientation difference $\delta_{j,k} = \text{mod}(|\theta_{A,j} - \theta_{B,k}|, \pi)$ will increase the distance between points by an extra penalty amount if the orientations are unsimilar. This orientation penalty can be adjusted by the parameter γ and prevents the assignment of points with different orientations thus leading to better matching results.

3) *Deterministic Annealing:* With the parameter β in (2) the softness of the assignment can be controlled. Following the approach of [5] the ICP matching with soft assign is repeated several times with different β values. Therein the found transformation $(\mathbf{R}_{AB}, \mathbf{t}_{AB})$ of each iteration is the starting point for the next iteration. The process starts with a small β , i. e. a soft assignment, and β is increased with each iteration which leads to a harder assignment. At the end, large β values are reached that cause a hard assignment, i. e. the $m_{j,k}$ values are either very close to 0 or very close to 1. This process is called *deterministic annealing* and is also used in other fields to solve optimization problems, e. g. in the context of Markov Random Fields [20].

The complete matching procedure using the three ICP enhancements described above is presented in Algorithm 3.

C. Alignment Results on Real World Data

We tested our alignment method on real world driving data. Fig. 6(a) and 6(c) show two examples of PFS map pairs from the test. The first map in each pair was captured on a route in a suburban area. The second map was captured on roughly the same spot when passing by a second time a couple of minutes later. As we had only one vehicle for our experiments available, we choose this way to emulate a V2V scenario. For both example pairs our ICP-based method is capable of realigning the PFS maps in a reasonable manner as shown in Fig. 6(b) and 6(d). As desired, the common obstacle boundaries (red continuous lines) from both maps are well aligned.

IV. CONCLUSION AND FUTURE WORK

In this paper, a process to fuse free space maps from multiple vehicles is proposed. The PFS maps from [2] are chosen as input of the fusion process due to their compact form, which still captures all relevant free space information.

The core of our fusion method is a graph-based algorithm where the intersecting map boundaries are converted to a fusion graph, which captures the essential information for the fusion process. First, a ‘‘conciliation’’ algorithm is applied to this graph to treat conflicting free space claims between the maps. On the obtained conciliated fusion graph the actual fusion is conducted by a path search following a set of simple rules. All cycles found by this path search represent the fused free space boundaries. Usually, the fused boundaries are found after very few iterations, since the graph is substantially reduced within each iteration. The proposed method thereby offers an efficient way to fuse free space areas.

Furthermore, a alignment method for PFS maps is presented to compensate for a possibly biased pose measurement between two maps. For the alignment an ICP matching algorithm with softassign and deterministic annealing from [5] is used to obtain proper outlier handling and convergence behavior. In a modified point assignment the orientation information of the PFS maps is incorporated to further improve outlier handling. We demonstrate the capability of this method to compensate pose inaccuracies on real world examples and achieve reasonable matches. To further reduce false point assignments, we are planning to incorporate geometric relationships between the matching points in the future, like it is done in [22]. Thereby the similarity of connections between neighboring boundary points is additionally considered during the assignment process.

Although our graph-based fusion algorithm is designed for V2V-interchanged PFS maps, it can also be utilized for free space fusion in other domains, e. g. mobile robotics. The only

Algorithm 3 PFS map matching

(modified version of algorithm from [5])

Input: $\{\mathbf{p}_A\}, \{\mathbf{p}_B\}$ //sets of sampled boundary points
Set $\mathbf{R}_{AB}, \mathbf{t}_{AB}, \mathbf{M}$ and β to there initial values.

while $\beta \leq \beta_{\text{final}}$ **do** //deterministic annealing

while \mathbf{M} does not converge **do** //ICP matching

$m_{j,k} \leftarrow e^{-\beta \tilde{Q}_{j,k}}$ with $\tilde{Q}_{j,k}$ from (3).

Normalize rows and columns of \mathbf{M} iteratively with:

$$m_{j,k}^1 = \frac{m_{j,k}^0}{\sum_{k=1}^K m_{j,k}^0}, \quad m_{j,k}^2 = \frac{m_{j,k}^1}{\sum_{j=1}^J m_{j,k}^1}.$$

Minimize $E(\mathbf{M}, \mathbf{R}_{AB}, \mathbf{t}_{AB})$ w. r. t. \mathbf{R}_{AB} and \mathbf{t}_{AB} .*

$\mathbf{p}_B \leftarrow \mathbf{R}_{AB}\mathbf{p}_B + \mathbf{t}_{AB}$. //Apply the found transformation

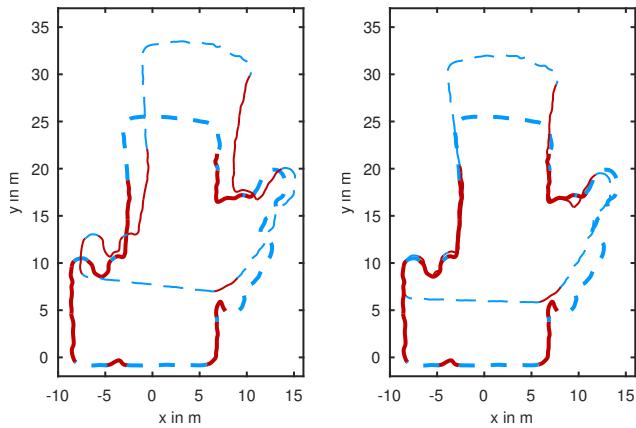
end while

$\beta \leftarrow \beta_r \beta$. //increase β with rate β_r for next ICP

end while

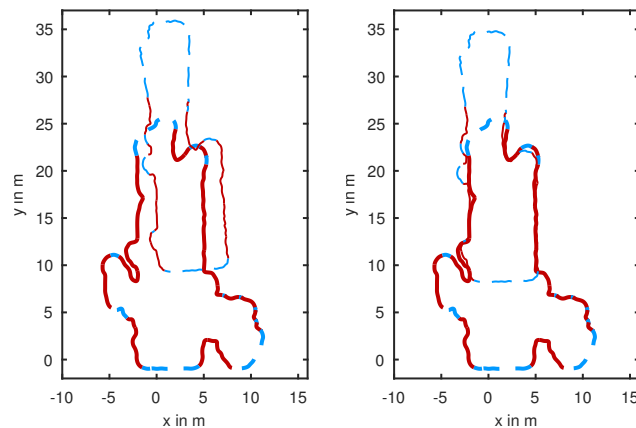
Output: $(\mathbf{R}_{AB}, \mathbf{t}_{AB})$ //final transformation after matching

*This is done by using the singular value decomposition method from [21].



(a) Example pair of PFS maps from two different passes of the same spot.

(b) PFS map pair from (a) after alignment.



(c) Another example pair of PFS maps from two different passes of another spot.

(d) PFS map pair from (c) after alignment.

Fig. 6. Alignment results for PFS maps from a suburban driving scenario.

necessary prerequisite is the representation of overlapping free space areas by attributed boundaries similar to PFS maps. For example, our method could be applied with few modifications to the fences representation from [12].

A fusion of more than two free space areas is also possible by applying the proposed fusion process repeatedly. Thus vehicles can increase their field of view substantially by interchanging and fusing PFS maps. The resulting high range environment model can be provided to several ADAS or automated driving functions and enable them to perform more foresighted planning.

ACKNOWLEDGMENTS

We kindly thank Continental AG for their great cooperation within Proreta 4, which is a joint research project of Technische Universität Darmstadt and Continental AG to investigate future concepts for intelligent and learning driver assistance systems.

REFERENCES

- [1] H. Winner, F. Lotz, E. Bauer, U. Konigorski, M. Schreier, J. Adamy, M. Pfromm, R. Bruder, S. Lücke, and S. Cieler, "Proreta 3 - an integrated ADAS concept: Comprehensive driver assistance by safety corridor and cooperative automation," in *Tagungsband 1. Internationale ATZ-Fachtagung Fahrerassistenzsysteme*, 2015.
- [2] M. Schreier and V. Willert, "Robust free space detection in occupancy grid maps by methods of image analysis and dynamic B-Spline contour tracking," in *2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2012, pp. 514–521.
- [3] M. Schreier, V. Willert, and J. Adamy, "From grid maps to parametric free space maps: A highly compact, generic environment representation for ADAS," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 938–944.
- [4] —, "Compact representation of dynamic driving environments for ADAS by parametric free space and dynamic object maps," *IEEE Trans. Intell. Transport. Syst.*, vol. 17, no. 2, pp. 367–384, 2016.
- [5] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness, "New algorithms for 2D and 3D point matching," *Pattern Recognition*, vol. 31, no. 8, pp. 1019–1031, 1998.
- [6] H. Li, M. Tsukada, F. Nashashibi, and M. Parent, "Multivehicle cooperative local mapping: A methodology based on occupancy grid map merging," *IEEE Trans. Intell. Transport. Syst.*, vol. 15, no. 5, pp. 2089–2100, 2014.
- [7] J. Miura, Y. Negishi, and Y. Shirai, "Mobile robot map generation by integrating omnidirectional stereo and laser range finder," in *International Conference on Intelligent Robots and Systems (IROS)*, 2002, pp. 250–255.
- [8] R. Grewe, A. Hohm, S. Hegemann, S. Lueke, and H. Winner, "Towards a generic and efficient environment model for ADAS," in *2012 IEEE Intelligent Vehicles Symposium (IV)*, 2012, pp. 316–321.
- [9] S. Topal, "A novel map merging methodology for multi-robot systems," in *Proceedings of the World Congress on Engineering and Computer Science 2010, Vol 1*, 2010, pp. 383–387.
- [10] S. Saeedi, L. Paull, M. Trentini, and H. Li, "Multiple robot simultaneous localization and mapping," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 853–858.
- [11] S. Carpin, "Fast and accurate map merging for multi-robot systems," *Autonomous Robots*, vol. 25, no. 3, pp. 305–316, 2008.
- [12] T. Kubertschak, M. Maehlich, and H. J. Wuensche, "Towards a unified architecture for mapping static environments," in *17th International Conference on Information Fusion (FUSION)*, 2014, pp. 1–8.
- [13] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.
- [14] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 3185–3192.
- [15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-slam: A versatile and accurate monocular slam system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.
- [16] J. Cousty, G. Bertrand, M. Couprie, and L. Najman, "Fusion graphs, region merging and watersheds," in *Discrete geometry for computer imagery*, ser. Lecture Notes in Computer Science, A. Kuba, Ed. Springer, 2006, vol. 4245, pp. 343–354.
- [17] S. Biswas and B. C. Lovell, *Bézier and Splines in Image Processing and Machine Vision*. London: Springer, 2008.
- [18] R. Diestel, *Graph theory*, 3rd ed., ser. Graduate texts in mathematics. Berlin and Heidelberg and New York: Springer, 2006, vol. 173.
- [19] A. Yuille and J. Kosowsky, "Statistical physics algorithms that converge," *Neural Computation*, no. Vol. 6, pp. 341–356, 1994.
- [20] D. Geiger and F. Giosi, "Parallel and deterministic algorithms from MRF's: Surface reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 5, pp. 401–412, 1991.
- [21] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 5, pp. 698–700, 1987.
- [22] X. Hu and N. Ahuja, "Matching point features with ordered geometric, rigidity, and disparity constraints," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, no. 10, pp. 1041–1049, 1994.