Vehicle Localization by Matching Triangulated Point Patterns

Jan-Henrik Haunert
Institut für Informatik
Universität Würzburg
Am Hubland, 97074 Würzburg
jan.haunert@uni-wuerzburg.de

Claus Brenner
Institut für Kartographie und Geoinformatik
Leibniz Universität Hannover
Appelstraße 9a, 30167 Hannover
claus.brenner@ikg.uni-hannover.de

ABSTRACT

We consider the problem of localizing a moving vehicle based on landmarks that were detected with a vehicle-mounted sensor. Landmarks are represented as points; correspondences of these points with the ones in a reference database are searched based on their geometric configurations. More specifically, we triangulate the landmark points and we match the obtained triangles with triangles in a reference database according to their geometric similarity. We maximize the number of triangle matches while considering the topological relations between different triangles, for example, if two triangles share an edge then the corresponding reference triangles must share an edge. Our method exploits that the observed points typically form a certain configuration: They appear at a limited distance from the vehicle's trajectory, thus the typical point pattern has a large extent in the driving direction and a relatively small lateral extent. This characteristic allows us to triangulate the observed point set such that we obtain a triangle strip (a sequence of triangles) in which each two consecutive triangles share one edge and each triangle connects three points that are relatively close to each other, that is, the triangle strip appropriately defines a neighborhood relationship for the landmarks. The adjacency graph of the triangles becomes a path; this allows for an efficient solution of our matching problem by dynamic programming. We present results of our method with data acquired with a mobile laser scanning system. The landmarks are objects of cylindric shape, for example, poles of traffic signs, which can be easily detected with the employed sensor. We tested the method with respect to its running time and its robustness when imposing different types of errors on the data. In particular, we tested the effect of non-rigid distortions of the observed point set, which are typically encountered during dead reckoning. Our matching approach copes well with such errors since it is based on local similarity measures of triangles, that is, we do not assume that a global non-rigid transformation between the observed point set and the reference point set exists.

Manuscript submitted to ACM GIS 2009.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization; H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*; I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles, Sensors*

General Terms

Algorithms, Measurement

Keywords

Positioning, Point Pattern Matching, Dynamic Programming

1. INTRODUCTION

Vehicle self-positioning can be enabled with different systems, for example, with the satellite-based system GPS. Classically, GPS is used in cars to assist drivers in their wayfinding tasks. Today, however, positioning systems become increasingly important as components of driver assistance systems that actively control the vehicle's motion, for example, a car may be forced to slow down when approaching a turn in the road. As a long-term goal, many researchers envision fully autonomous vehicles. Obviously, the development of such vehicles requires a high attention to safety risks; therefore, redundant positioning systems are needed. Since GPS is not reliable in shadowed areas such as forests and street canyons, it needs to be combined with other types of positioning systems. In this paper, we present a positioning system based on a vehicle-mounted laser scanner. Laser scanners may be used to locate a vehicle in an indoor environment, for example, by matching the observations with a polygonal reference map [10]. Our approach, however, is to first extract a set of landmark points from the observations and to use the landmarks for positioning. We solve the positioning problem by point pattern matching (PPM), that is, by finding a transformation that maps each of the observed landmark points onto or sufficiently close to a corresponding point in a reference map. Our PPM method is deterministic, efficient, and tailored for the vehicle positioning task. We think that our landmark-based approach is especially useful for the urban outdoor environment, where we find many landmarks that can be represented as points, for example, poles of traffic signs.

Point pattern matching has been applied to a wide range of applications and it has been attacked with different approaches – a detailed review is given in [12]. Depending on the specific application, a PPM method needs to cope with inaccuracy and incompleteness of the datasets and with point motions. Often the problem is to find a global transformation that preserves all intra-distances and intra-angles of a point set, that is, a translation and rotation [2]; the Hausdorff distance between both point sets can be used to measure the quality of the transformation [4]. In many applications, however, the intra-angles and intra-distances of a point set may change, for example, because the points represent objects with non-rigid motions or because of non-linear sensor distortions. Often changes of long intra-distances may be relatively large while distances between neighboring points do not change much. A typical application in which small local deformation can lead to huge global distortion is fingerprint verification. The problem can be attacked with triangulation-based methods [1, 11], which require that the distances of an observed point to its neighbors are similar to the distances of the corresponding reference point to its neighbors.

Local distortions that have a global effect also appear in vehicle positioning: Figure 1 displays the track of a vehicle (black arc). At certain locations (white dots) the vehicle observes a landmark (black dots), whose coordinates (in a local coordinate system) are calculated based on the landmark's polar coordinates with respect to the vehicle and the vehicle's own position and orientation – we assume that the vehicle's position and orientation are estimated by dead reckoning, that is, by integrating information received from motion sensors such as odometers, gyroscopes or accelerometers. After the vehicle observed a certain number of landmarks, it tries to determine its current position (in a global coordinate system) by finding the observed configuration of points in a reference database, that is, by solving the PPM problem. In this use case, the transformation between the observed point set and the reference set cannot be expressed as a global rotation and translation because, due to erroneous measurements of the motion sensors, the estimated path (gray arc) of the vehicle may not coincide with the actual track. The point set may be deformed. The distances between neighboring landmark points, however, may be relatively accurate.

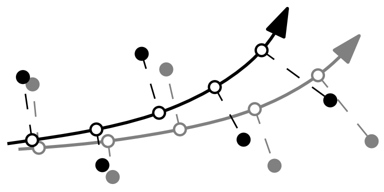


Figure 1: Deformation of a point set due to differences between the estimated vehicle path (gray) and the actual path (black).

Our new PPM method for vehicle positioning is similar to the existing triangulation-based approaches for fingerprint verification. The newly developed method, however, is tailored for the positioning task: The points represent landmarks that are located at the roadside, for example, poles of traffic signs or trees. Such objects are easily identifiable in point clouds acquired with vehicle-mounted laser scanners [3]. The landmarks appear along the track, thus the typical point pattern has a large extent in the driving direction and a relatively small lateral extent, see Fig. 2. Our approach is to connect the observed points with a set of triangles such that the triangles form a path, see Fig. 3. We argue that the typical point configuration allows us to construct such a triangle set without introducing extremely long triangles. The presented PPM method exploits the path-like structure of the triangle set.

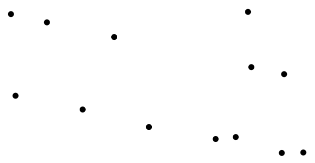


Figure 2: A typical configuration of points observed from a moving vehicle.

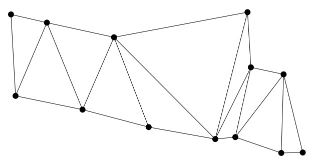


Figure 3: A set of triangles connecting the points in Fig. 2; the adjacency graph of the triangles is a path.

The method matches triangles with triangles in a reference database according to their geometric similarity. Topological constraints are ensured, for example, if two triangles share an edge then their two corresponding triangles in the reference dataset must share an edge. Because of these constraints, the method can be classified as a graph matching method. A review of graph matching methods in the pattern recognition context is given in [5]. Often graph matching problems are computationally very demanding, for example, the largest common subgraph problem is NP-hard [9]. However, we define a graph matching problem that allows for an efficient and exact solution if one of the graphs is a path, which we ensure with our special triangulation method.

In the remaining part of the paper we present our method for point pattern matching (Sect. 2) and discuss some experimental results (Sect. 3). Finally, we conclude the paper and give recommendations for future research (Sect. 4).

2. MATCHING METHODOLOGY

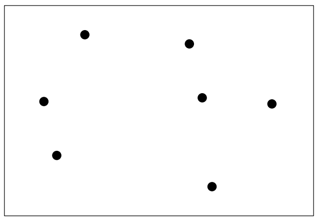
Let $P = \{p_i \in \mathbb{R}^2\}$ with $i = 1, 2, \ldots, m$ be the set of observed points and $P' = \{p'_j \in \mathbb{R}^2\}$ with $j = 1, 2, \ldots, n$ be the set of reference points. The matching problem is to find a set $\pi \subseteq P \times P'$ of correspondences between the points in P and P'. Our method for solving this task comprises the following steps:

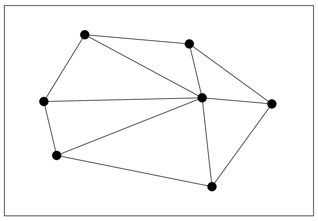
1. We define the set T', in which each element is a triangle of three distinct points in P'. For the moment it suffices to assume that T' contains all such triangles. We will discuss in Sect. 3 how to keep T' small.

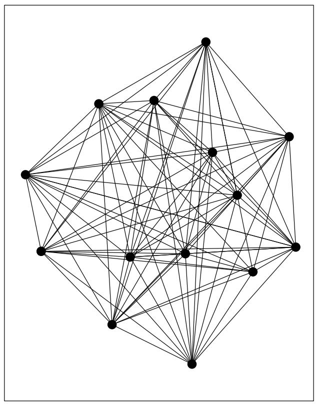
2. After we observed the point set P, we triangulate the points in P, which yields the set of triangles T.

3. We search a set $\theta \subseteq T \times T'$ of triangle matches. Each element $(t, t') \in \theta$ means that $t \in T$ corresponds to the reference triangle $t' \in T'$. We aim to match triangles that are relatively similar and we want to avoid contradictions between different matches in θ by considering the topological relations of the triangles. The triangle matches θ define the point matches π.

We define the graph $G(T, E)$ with E containing an element for each pair of triangles in T that share an edge. Accordingly, we define the graph $G'(T', E')$ with E' containing an element for each pair of triangles in T' that share an edge.


(a) observed point set P


(b) Triangulation T of P


(c) reference point set P' with edges forming reference triangles T'

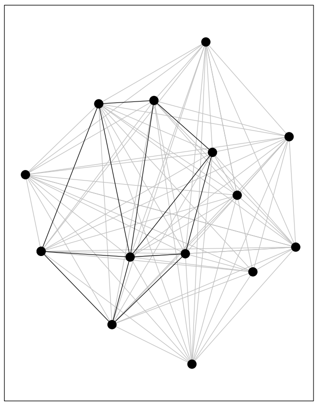

(d) matching result

Figure 4: Approach to find a point configuration in a reference database.

Obviously, we need to specify our approach; Figure 4 illustrates one possibility. The points in P were triangulated according to the Delaunay criterion. Basically, we could solve the matching problem in step 3 of our method by finding a subgraph isomorphism $f : T \to T'$, where $f(t)$ is restricted to triangles that, according to some geometric measure, are sufficiently similar to $t \in T$. If the reference database is incomplete, T may contain triangles that do not have a corresponding triangle in T'. In this case we would fail to find a subgraph isomorphism f but instead we could solve the largest common subgraph problem, that is, we would need to find an isomorphism $g : T_{\text{sub}} \to T'$ with $T_{\text{sub}} \subseteq T$ and $|T_{\text{sub}}|$ as high as possible. By solving the largest common subgraph problem we would consider both topological and geometric information. However, there is a drawback of this approach: The largest common subgraph problem (as well as the subgraph isomorphism problem) is NP-hard [9], which means that we cannot hope for an efficient and exact solution. In order to overcome this drawback, we propose an alternative triangulation method (Sect. 2.1). Furthermore, we define a relation between the matched subgraphs of G and G' that is weaker than an isomorphism (Sect. 2.2). We then present an efficient algorithm that solves the matching problem while ensuring the defined relation (Sect. 2.3).

2.1 Triangulating the observed point set

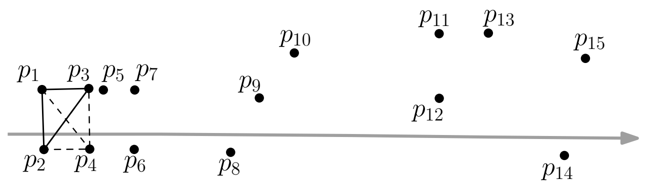
The common definition of a triangulation requires that the union of all triangles equals the convex hull of the input points [8]. We relax this requirements. Instead, we only require that each point in P belongs to at least one triangle in T and that the graph G is connected. More specifically, we triangulate the points P such that the graph G becomes a path, that is, we define a sequence of consecutively adjacent triangles. Furthermore, we require that the triangle sequence can be dynamically updated: whenever we observe a new landmark we would like to append one triangle to the triangle sequence constructed so far. Subject to these requirements we would like to obtain a set of well-shaped triangles. In other words, the triangulation should be 'similar' to a Delaunay triangulation. We achieve this by trying to avoid overlapping triangles (triangles whose interiors intersect) and by preferring triangles with large interior angles. In order to formalize our triangulation algorithm, we assume that the point sequence $(p_1, p_2, \ldots, p_m)$ with $m \geq 3$ is sorted according to the order in which the points were observed along the vehicle's track. Furthermore, we denote the smallest angle of a triangle by $\alpha_{\text{Min}} : P^3 \to \mathbb{R}_0^+$. Algorithm 1 yields the triangle sequence $(t_1, t_2, \ldots, t_{m-2})$; these triangles constitute the set T.

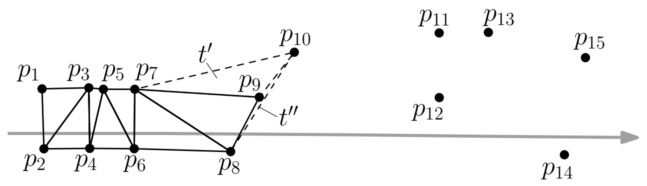
Algorithm 1 Algorithm for triangulating a point sequence

```
1:  $t_1 \leftarrow (p_1, p_2, p_3)$
2:  $p' \leftarrow p_1$
3:  $p'' \leftarrow p_2$
4:  for $i \leftarrow 4$ to $m$
5:      $t' \leftarrow (p', p_{i-1}, p_i)$
6:      $t'' \leftarrow (p'', p_{i-1}, p_i)$
7:      if $t'$ overlaps $t_{i-3}$ and $t''$ does not overlap $t_{i-3}$
8:          $t_{i-2} \leftarrow t''$
9:          $p' \leftarrow p_{i-1}$
10:     else if $t'$ does not overlap $t_{i-3}$ and $t''$ overlaps $t_{i-3}$
11:         $t_{i-2} \leftarrow t'$
12:         $p'' \leftarrow p_{i-1}$
13:     else if $\alpha_{\text{Min}}(t') < \alpha_{\text{Min}}(t'')$
14:         $t_{i-2} \leftarrow t''$
15:         $p' \leftarrow p_{i-1}$
16:     else
17:         $t_{i-2} \leftarrow t'$
18:         $p'' \leftarrow p_{i-1}$
19:     end if
20: end for
```

Figure 5 shows the progress of the algorithm for an example. In any case the triangle (p_1, p_2, p_3) is the first triangle in the sequence. For any point p_i with $i > 3$ we consider two candidate triangles that can be added, namely t' and t'', which are defined in lines 5 and 6 of algorithm 1, respectively. Both t' and t'' append the triangulation at an edge that was added in the last iteration: t' contains the edge (p', p_{i-1}) and t'' contains the edge (p'', p_{i-1}). If only one of these two triangles overlaps the last triangle in the sequence, the other triangle is added to it (lines 7–12). Otherwise the triangle that has the largest minimum angle among t' and t'' is added (lines 13–18).

Figure 6 shows a second instance processed with algorithm 1. The example shows that, in some cases, we cannot avoid overlapping triangles. Our matching method, however, copes with such cases. Therefore we accept such triangle sequences. In any case, we obtain a sequence of consecutively adjacent triangles. Since we append the triangle sequence by maximizing the minimum angle, we obtain well-shaped triangles.


(a) in the first iteration ($i = 4$) of algorithm 1 the triangles $t' = (p_1, p_3, p_4)$ and $t'' = (p_2, p_3, p_4)$ are tested; the triangle t'' is added to T because t' overlaps (p_1, p_2, p_3)


(b) in the seventh iteration ($i = 10$) of algorithm 1 the triangles $t' = (p_7, p_9, p_{10})$ and $t'' = (p_8, p_9, p_{10})$ are tested; the triangle t' is added to T because $\alpha_{\text{Min}}(t'') < \alpha_{\text{Min}}(t')$

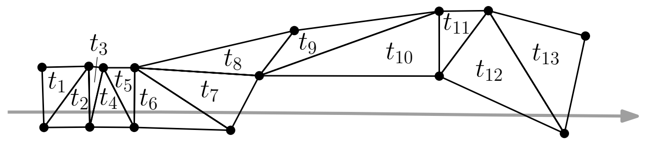
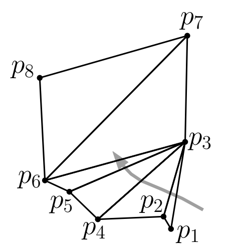

(c) final result

Figure 5: Progress of algorithm 1. The gray arc is the vehicle's trajectory.


(a) the triangle sequence after processing points p_1–p_8

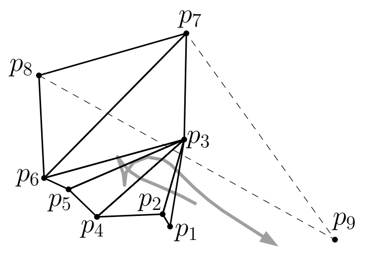

(b) both $t' = (p_7, p_8, p_9)$ and $t'' = (p_6, p_8, p_9)$ overlap the last triangle (p_6, p_7, p_8) in the sequence; t' is selected since $\alpha_{\text{Min}}(t'') < \alpha_{\text{Min}}(t')$

Figure 6: An instance for which algorithm 1 produces overlapping triangles: point p_9 appears after a turn-over.

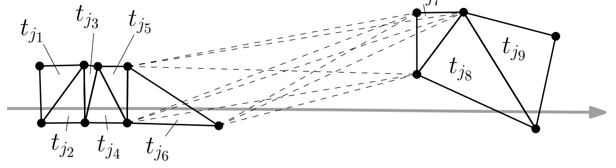


Figure 7: The idea behind constraint (C5b) illustrated for the example in Fig. 5. The triangles t_7 to t_{10} are not matched. Therefore, the distances (dashed lines) between the triangles $t_{j_6} = t_6$ and $t_{j_7} = t_{11}$ are compared with the distances between the matching reference triangles.

2.2 Matching problem

A problem instance of the matching problem is defined by the triangle sequence $(t_1, t_2, \ldots, t_{m-2})$ obtained with algorithm 1 from the observed points, the reference triangle set T', and a tolerance $\varepsilon \in \mathbb{R}^+$ that defines whether two triangles are sufficiently similar or not. The problem is to find the set θ of triangle matches.

For each match (t, t') in θ, we want that the triangles t and t' are sufficiently similar. Therefore we define constraint (C1).

(C1) Let $s_{\max} : T \cup T' \to \mathbb{R}_0^+$, $s_{\text{med}} : T \cup T' \to \mathbb{R}_0^+$, and $s_{\min} : T \cup T' \to \mathbb{R}_0^+$ be the lengths of the sides of $t \in T' \cup T$ (with $s_{\max}(t) \geq s_{\text{med}}(t) \geq s_{\min}(t)$). For each $(t, t') \in \theta$ we require that $|s_{\max}(t) - s_{\max}(t')| \leq \varepsilon$, $|s_{\text{med}}(t) - s_{\text{med}}(t')| \leq \varepsilon$, and $|s_{\min}(t) - s_{\min}(t')| \leq \varepsilon$.

Since we do not want that any two elements in θ are conflicting we define additional constraints. For all $(a, a') \in \theta$ and $(b, b') \in \theta$ we require:

(C2) if $a = b$ then $a' = b'$, that is, a triangle in T must not be matched to more than one reference triangle, and

(C3) if a and b share a common edge then a' and b' must share a common edge.

Note that, for the largest common subgraph problem, we would need to replace the 'if' in constraints (C2) and (C3) by 'if and only if'. Our problem statement is in a way weaker, for example, it allows two distinct triangles to be matched with the same reference triangle. We cannot strictly forbid this case with our dynamic programming approach in Sect. 2.3. However, we can introduce the two additional constraints (C4) and (C5), which make our problem statement stronger. With these constraints we assume that we avoid most incorrect triangle matches.

Let $t_{i_1}, t_{i_2}, \ldots, t_{i_{|\theta|}}$ with $i_j \in \{1, 2, \ldots, m-2\}$ and $i_j < i_{j+1}$ for $j = 1, 2, \ldots, |\theta| - 1$ be the triangles in T that are matched to a reference triangle. Furthermore, let t'_j be the reference triangle matched with t_{i_j}. In other words, the sequence $S = ((t_{i_1}, t'_1), (t_{i_2}, t'_2), \ldots, (t_{i_{|\theta|}}, t'_{|\theta|}))$ of the matches in θ is sorted according to the position of the triangles in the triangle strip. For all $j = 1, 2, \ldots, |\theta| - 1$ we require:

(C4) the two triangles t_{i_j} and $t_{i_{j+1}}$ must not be matched to the same reference triangle, that is, $t'_j \neq t'_{j+1}$, and

(C5) if t_{i_j} and $t_{i_{j+1}}$ do not share a common edge then

(a) the reference triangles t'_j and t'_{j+1} matched to t_{i_j} and $t_{i_{j+1}}$ must not share a common edge.

(b) the distances between the vertices of t_{i_j} and the vertices of $t_{i_{j+1}}$ must be *sufficiently similar* to the distances between the vertices of t'_j and the vertices of t'_{j+1}.

We use the error tolerance ε to state constraint (C5b) more precisely: For each triangle pair we can calculate nine distances between the vertices of the two different triangles. We do this for the triangle pair $(t_{i_j}, t_{i_{j+1}})$ and sort the distances, and we do the same for the triangle pair (t'_j, t'_{j+1}). We require that the longest distance for $(t_{i_j}, t_{i_{j+1}})$ does not differ more than ε from the longest distance for (t'_j, t'_{j+1}). The same we require for the second to ninth longest distances. Figure 7 illustrates constraint (C5b) on an example.

Constraints (C1)–(C5) define the set of feasible solutions to our problem. We measure the quality of a feasible solution according to two objectives:

(O1) The number of triangle matches in θ should be maximized.

(O2) Among the solutions that maximize (O1), we search the one that minimizes $\sigma^2 = \sum_{(t,t')\in\theta} c(t,t')$ with

$$c(t,t') = \Big(\big(s_{\max}(t) - s_{\max}(t')\big)^2 + \big(s_{\text{med}}(t) - s_{\text{med}}(t')\big)^2 + \big(s_{\min}(t) - s_{\min}(t')\big)^2 \Big).$$

2.3 Matching algorithm

In order to solve the matching problem defined in Sect. 2.2, we define the directed acyclic graph $G_{\text{match}}(V_{\text{match}}, A_{\text{match}})$, in which each path corresponds to a feasible solution to our problem. The optimal solution is found by searching a path of maximum weight in G_{match}. For directed acyclic graphs, this problem can be solved by dynamic programming [7], which requires $\mathcal{O}(|V_{\text{match}}| + |A_{\text{match}}|)$ time. The nodes of the obtained path define the set of triangle matches.

The node set V_{match} is the set of candidate matches, that is, it contains all matches (t,t') such that, according to constraint (C1), the triangles $t \in T$ and $t' \in T'$ are sufficiently similar. In order to find V_{match}, we create an index for the triangles in T'. We use a kd-tree of three dimensions, in which each triangle $t' \in T'$ is represented by the vector $\big(s_{\max}(t'), s_{\text{med}}(t'), s_{\min}(t')\big)$. Indexing triangles based on their side lengths is generally considered the best approach to find geometrically similar triangles [6]. We define the function $f_{\text{cand}} : T \to 2^{T'}$, such that, for each $t \in T$, the set $f_{\text{cand}}(t) \subseteq T'$ contains those reference triangles that are sufficiently similar to t. We obtain the set $f_{\text{cand}}(t)$ from the kd-tree by applying a range query. The set V_{match} contains a match (t,t') for each $t \in T$ and $t' \in f_{\text{cand}}(t)$. The set A_{match} contains an arc from node $(t_i, a') \in V_{\text{match}}$ to node $(t_j, b') \in V_{\text{match}}$ with $i, j = 1, 2, \ldots m-2$ if and only if

(R1) $j > i$ *and*

(R2) $a' \neq b'$ *and*

(R3) t_i and t_j share the same number of edges as a' and b' *and*

(R4) in the case that t_i and t_j share no edges then the distances between the vertices of t_i and t_j and the distances between the vertices of a' and b' are sufficiently similar according to Sect. 2.2, constraint (C5b).

We introduce node weights $w : V_{\text{match}} \to \mathbb{R}^+$ such that the path of maximum weight corresponds to the optimal solution to the matching problem.

$$w(t,t') = m - c(t,t')/c_{\max} \quad \text{with} \quad c_{\max} = \max\{c(t,t') \mid (t,t') \in V_{\text{match}}\}.$$

We now prove the correctness of our approach.

Theorem 1. *The path of maximum weight in G_{match} is the optimal solution to the matching problem that was defined in Sect. 2.2.*

Proof. We need to show that there is a path in G_{match} with nodes θ if and only if θ is a feasible solution to the matching problem and that the path of maximum weight satisfies objectives (O1) and (O2). The set θ satisfies constraint (C1) if and only if θ is a subset of V_{match}. However, not all subsets of V_{match} define a path in G_{match}, just as not all subsets of V_{match} satisfy constraints (C2)–(C5).

First, we show that each feasible solution to the matching problem defines a path in G_{match}. If we are given a solution θ that satisfies constraints (C2)–(C5) we can sort the matches in θ according to the positions of the triangles in the triangle strip. We obtain the corresponding sequence S of matches. Clearly S is a path in G_{match}, since each pair $a = \big((t_{i_j}, t'_j), (t_{i_{j+1}}, t'_{j+1})\big)$ of consecutive matches in S is contained in A_{match}: constraint (C2) implies that a satisfies requirement (R1); constraint (C3) implies that a satisfies requirement (R3) if $i_j + 1 = i_{j+1}$, that is, if t_{i_j} and $t_{i_{j+1}}$ share a common edge; constraint (C5a) implies that a satisfies requirement (R3) if $i_j + 1 < i_{j+1}$, that is, if t_{i_j} and $t_{i_{j+1}}$ do not share a common edge; constraint (C4) implies that a satisfies requirement (R2); constraint (C5b) implies that a satisfies requirement (R4).

Second, we show that each path p in G_{match} constitutes a feasible solution to the matching problem. Because of requirement (R1), G_{match} is acyclic and the path cannot contain two distinct matches (t,t') and (u,u') with $t = u$; this implies that constraint (C2) is satisfied. Since we triangulated the points in P with algorithm 1, two triangles t_i and t_j with $j > i$ only share an edge if $j = i+1$. Therefore, constraint (C3) is satisfied if it is satisfied for each two consecutive matches in p. This, however, is assured with requirement (R3). In our problem definition, constraints (C4) and (C5) are only active for two subsequent matches $\big((t_{i_j}, t'_j), (t_{i_{j+1}}, t'_{j+1})\big)$. Therefore, they are satisfied by imposing requirements (R2), (R3), and (R4) on arcs.

Together, the first part and the second part of our proof imply that there is indeed a path in G_{match} with nodes θ if and only if θ is a feasible solution to the matching problem. To prove the correctness of our weight setting we first consider an alternative setting: By defining unit weights for all nodes in V_{match} the path of maximum weight will be optimal according to objective (O1) since a maximum number of matches will be selected. The same is true if we define a large constant weight for all nodes, for example, we can set the weight to m, that is, the number of observed points. In our actual weight setting w this value is reduced by $c(t,t')/c_{\max}$ to also consider objective (O2). Since a path in G_{match} contains at most $m-2$ nodes and $c(t,t')/c_{\max} \leq 1$ we have

$$\sum_{(t,t')\in\theta} c(t,t')/c_{\max} \leq m - 2 < w(u,u') \quad \text{for all } (u,u') \in \theta.$$

This implies that the weight of a path with k nodes is always greater than the weight of a path with $k-1$ nodes, which implies that objective (O1) is assured by searching the path of maximum weight. If there are two or more paths with the same number of nodes, then the path that minimizes $\sum_{(t,t')\in\theta} c(t,t')$ among the paths with a maximum number of nodes has the highest weight. Consequently, the path of maximum weight yields the optimal solution to the matching problem. $\square$

3. EXPERIMENTAL RESULTS

The developed matching method was tested for a dataset of points representing poles of traffic signs, trunks, or other cylindric objects in the city of Hanover, Germany. The point features were automatically detected in a point cloud that was captured with a streetmapper system on a 22 km long track, see Fig. 8. A detailed description of the system and the operator used for the detection of point features can be found in [3]. We present the set-up of our tests in Sect. 3.1 and the results in Sect. 3.2.

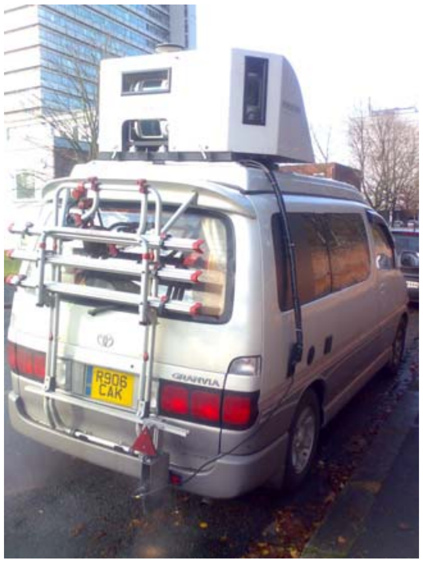


Figure 8: The streetmapper system used to acquire the scan in which point features (poles) were detected. The system comprises four terrestrial laser scanners, a GPS antenna (on top of the van), an odometer (mounted close to the bumper), an IMU, and a control computer.

3.1 Experimental set-up

The complete set of 2658 points defined the reference set P' for all our experiments. We used the same dataset to define several sample sets; each of these sample sets defined the input set P of observed points for one experiment. In order to test our method under conditions close to reality, we added noise to these points. More precisely, we defined the samples for P as follows:

1. We fragmented the track of 22 km into 88 sub-tracks, each corresponding to a drive of 40 seconds.
2. We simulated a drive along each sub-track and constructed the sequence of observed points by assuming a viewing distance of 25 m and a field of view of 100° (viewing direction equal to direction of motion).
3. We deformed the observed point sets – similar to the model sketched in Fig. 1 – by subjecting them to random, normally distributed errors of four types, namely errors in measuring
   (a) the driving distance (for example, using the vehicle's odometer) – here we assumed a standard deviation of 5 cm per m.
   (b) the vehicle's orientation (for example, using the vehicle's gyroscope) – here we assumed a standard deviation of 0.1° per m.
   (c) the distance between the vehicle and the landmark (for example, using a vehicle-mounted laser scanner) – here we assumed a standard deviation of 0.2 m.
   (d) the direction angle of the landmark from the vehicle (for example, using the same scanner) – here we assumed a standard deviation of 0.5°.

We consider this setting rather pessimistic – or realistic if we assume that the vehicle observing P is equipped with low-cost sensors. Definitely, we do not need the streetmapper system in Fig. 8 to achieve this accuracy.

For five sub-tracks the set P contained less than three point – obviously this implies that we cannot construct any triangle. This problem occurred on a highway with noise protection walls to both sides. We only consider the remaining 83 sub-tracks in our test statistics.

To keep the number of triangles T' in the reference database small, we did not include large triangles. More precisely, we included each triangle in $(P')^3$ whose smallest enclosing disc has a radius not exceeding a user-defined threshold $r_{\mathrm{Max}} \in \mathbb{R}^+$. The idea behind this approach is that, assuming an immobile sensor with viewing distance r_{Max}, all triangles constructed for the observed point set P will have this property. In this case we will not miss any matching reference triangle with our approach. In our experiment, however, the sensor was moving. Therefore, we set r_{Max} higher than the viewing distance, that is, we set $r_{\mathrm{Max}} = 50$ m. We cannot guarantee that there is a matching reference triangle for each triangle in T. However, it is still reasonable to exclude large triangles, since these do not contribute much information: Long distances are very inaccurate when assuming our error model. With our approach the set T' contained 643247 triangles.

Finally, we need to specify the parameter ε that defines whether two triangles are sufficiently similar. We tested four settings, namely, $\varepsilon := 0.25$ m, $\varepsilon := 0.50$ m, $\varepsilon := 0.75$ m, and $\varepsilon := 1.00$ m. We present results of these tests in the next section.

3.2 Test results

Table 1 summarizes our experimental results for the tests defined in the last section. The smallest instance that we processed contained three landmarks, that is, the set T contained only one triangle. The largest of our instance contained 127 landmarks – this situation occurred when the track followed an alley with many trees at both sides. Due to regular distances between the trees, we obtained many matching candidates for the triangles in T. Therefore, this instance was not only the instance with most landmarks but also the instance with the largest number of nodes in G_{match}. The time for processing this instance was relatively high (see second row of Table 1) but with all our settings for ε the solution only contained correct matches. The third and second row of Table 1 show average results for all 83 instances. Obviously, the number of candidate matches per triangle increases with a higher value of ε because more reference triangles are considered sufficiently similar. If we allow two triangles to be matched even if their edge lengths differ by one meter, we obtain many (58.1 on average) matching can-

	$\varepsilon := 0.25$ m	$\varepsilon := 0.50$ m	$\varepsilon := 0.75$ m	$\varepsilon := 1.00$ m
solution time for the largest instance (127 landmarks)	0.28s	5.46s	33.74s	129.11s
average number of candidate matches (matches satisfying constraint (C1)) per triangle of observed points	1.5	9.2	26.7	58.1
average solution time (including triangulation and matching)	0.04s	0.41s	2.75s	11.88s
total number of unmatched triangles of observed points	2772	1685	385	107
percentage of unmatched triangles of observed points	91.5%	55.6%	12.7%	3.5%
total number of correctly matched triangles of observed points	235	1335	2631	2906
percentage of correctly matched triangles of observed points	7.8%	44.1%	86.9%	96.0%
total number of incorrectly matched triangles of observed points	22	9	13	16
percentage of incorrectly matched triangles of observed points	0.7%	0.3%	0.4%	0.5%
number of instances with any incorrect match	20	6	6	6
number of instances where majority of matches is correct	64	80	81	80

Table 1: Experimental results for our 83 sub-tracks. Computation times are in seconds CPU time. All experiments were performed on a Windows PC with 64 bits, 8 GB RAM, and a 2.93 GHz Intel CPU.

didates. The average time for solving the problem instances increases with higher values for ε but even with $\varepsilon := 1.00$ m it is far below 40 s, that is, the duration of the drive along the track. If we can develop an algorithm that starts when the first three landmarks were observed and dynamically updates the matching result when a new landmark is observed, we can hope for a practicable solution. We discuss this idea of a dynamic version of our algorithm in the next section.

The fifth to ninth row of Table 1 allow us to assess the reliability of our matching method. In our experiments with $\varepsilon := 0.25$ m and $\varepsilon := 0.5$ m most of the triangles in T were unmatched because most of the triangles were deformed more than the specified tolerance. Nevertheless, in both cases we obtain only a few incorrect matches and the number of correct matches is much higher. In particular, for $\varepsilon := 0.5$ m we observed incorrect matches only for six instances. For the other 77 instances all matches found were correct – this should suffice to localize the vehicle. If we count the instances for which the majority of triangle matches is correct than we obtain an even higher number (80). Presumably we can find the right location for all these instances. If we assess the experiments for $\varepsilon := 0.75$ m and $\varepsilon := 1.00$ m than we obtain much more correct triangle matches. The number of instances with incorrect matches, however, does not decrease. Certainly, the higher number of triangle matches will improve the accuracy of the estimated coordinates of the vehicle. The localization can be performed, however, also when a few correct triangle matches are found.

Finally, we discuss Fig. 9, which shows the running time for all our experiments with $\varepsilon := 0.75$. Only for four instances the solution required more than 10 seconds. Certainly we can keep the running time small by restricting the size of P, for example, by only considering the last k landmarks observed, with k being a user-defined threshold.

Obviously, if the reference dataset is much larger than in our experiment (for example, it covers a hole country) then we need to think about how to keep the set of matching candidates for triangles small. If the position is approximately known, however, we could restrict the number of candidate matches for the triangles in T to those reference triangles in T' that are in a certain neighborhood of the estimated position, for example, by partitioning the reference dataset into tiles.

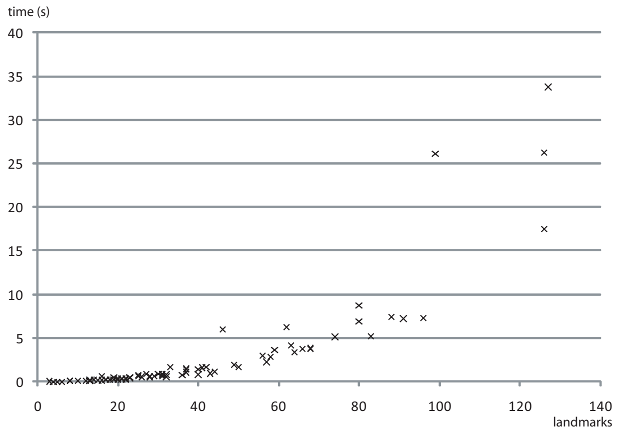


Figure 9: Running time of our matching algorithm with $\varepsilon := 0.75$ m. In each experiment the method was applied with another set P of observed points.

4. CONCLUSION AND OUTLOOK

We have presented a new deterministic and efficient method for point pattern matching that is designed to enable the self-positioning of vehicles. The method matches a set of triangles with a a set of reference triangles by considering the geometric similarity of triangles and the topological relations between them. A special triangulation algorithm is applied to enforce that the triangles connecting the observed points form a path – this property allows us to efficiently solve the matching problem. We argued that our path-like triangulation appropriately defines a neighborhood relation of points if the observed point set has a configuration that is typical for the positioning task: The observed points appear to the sides of a vehicle's track, thus the point patters has a large extent in direction of driving and a small lateral extent.

We tested our method for samples of up to 127 points and a reference dataset of 2658 points. Our tests allow us to conclude that the method is fast enough for application in practice and robust against random errors that are typical when recording the track by dead reckoning.

Future research will address the following research questions:

1. How robust is the method against incompleteness of the reference database? We will conduct further tests to answer this question.

2. How can we extend the method to allow for efficient and accurate updating of a previously found position? Again assuming that the point pattern has the characteristic large extent in one direction, we could add triangles at the front of the current triangulation whilst the vehicle is driving. When calculating a new position, only the candidate matches for the added triangles need to be found. Furthermore, our dynamic programming approach for solving the matching problem by finding the path of maximum weight in a graph is useful for this task: matching solutions found for the sequence of triangles 1 to $m-2$ can be reused when searching the solution for the triangles 1 to $m-1$.

3. In addition to the landmark points currently handled by our method, how can we enable the positioning based on planes, which are well detectable in the laser scanner point cloud?

5. REFERENCES

[1] G. Bebis, T. Deaconu, and M. Georgiopoulos. Fingerprint identification using Delaunay triangulation. In *Proceedings of the 1999 International Conference on Information Intelligence and Systems (ICIIS)*, pages 452–459, 1999.

[2] A. Bishnu, S. Das, S. C. Nandy, and B. B. Bhattacharya. Simple algorithms for partial point set pattern matching under rigid motion. *Pattern Recognition*, 39(9):1662–1671, 2006.

[3] C. Brenner. Extraction of features from mobile laser scanning data for future driver assistance systems. In *Advances in GIScience, Proceedings of the 12th AGILE Conference*, pages 25–42, 2009.

[4] M. Cho and D. M. Mount. Improved approximation bounds for planar point pattern matching. *Algorithmica*, 50(2):175–207, 2008.

[5] D. Conte, P. Foggia, C. Sansone, and M. Vento. Graph matching applications in pattern recognition and image processing. In *Proceedings of the 10th International Conference on Image Processing (ICIP 2003)*, volume II, pages 21–24, 2003.

[6] C. B. Cranston and H. Samet. Indexing point triples via triangle geometry. In *Proceedings of the 23rd IEEE International Conference on Data Engineering*, pages 936–945, 2007.

[7] S. Dasgupta, C. H. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw Hill Book Co, New York, 2006.

[8] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, Germany, 2008.

[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[10] O. Karch and T. Wahl. Relocalization – theory and practice. *Discrete Applied Mathematics*, 93:89–108, 1999.

[11] Z. M. Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1266–1276, 2000.

[12] B. Li, Q. Meng, and H. Holstein. Point pattern matching and applications - a review. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 729–736, 2003.