2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Robust and Accurate Road Map Inference

Gabriel Agamennoni, Juan I. Nieto and Eduardo M. Nebot

Abstract— **Over the last ten years, electronic vehicle guidance systems have become increasingly popular. However, their performance is subject to the availability and accuracy of digital road maps. Most current digital maps are still inadequate for advanced applications in unstructured environments. Lack of detailed up-to-date information and insufficient accuracy and refinement of the road geometry are among the most important shortcomings. The massive use of inexpensive GPS receivers, combined with the rapidly increasing availability of wireless communication infrastructure, suggests that large volumes of data combining both modalities will be available in a near future. The approach presented here draws on machine learning techniques to process logs of position traces to consistently build a detailed and accurate representation of the road network and, more importantly, extract the actual paths followed by vehicles. Experimental results with data from large mining operations are presented to validate the algorithm.**

I. INTRODUCTION

Lately, the wide availability of commercial digital road maps has enabled numerous vehicle navigation and guidance applications. Commercial vector maps cover to a great extent the major road networks in urban areas, achieving a level of accuracy in the order of a few meters. The combination of these digital maps with precise positioning systems has allowed the development of numerous in-vehicle applications providing navigation aid and driving assistance. However, there is a much wider range of potential applications beyond those aimed at enhancing driver convenience. Accurate road maps could also be used to improve road safety [1] by means of lane keeping, rollover warning, obstacle detection and collision avoidance systems.

A fundamental limitation of commercially available road maps is their poor accuracy. High-end road safety applications require decimeter accuracy, and currently this quality can only be achieved via probe vehicles and surveying methods. Therefore, maps are costly to produce and update. Many authors [2], [3], [4], [5] propose that, instead of a single high-precision probe, a large number of inexpensive low-precision information sources yield similar or even better results. With the emergence of low-cost positioning devices and the accelerated development of wireless communication systems, integrated data gathering and processing becomes feasible at a very large scale. This allows maps to be learnt from large volumes of position data. Even though data is polluted with noise and outliers, its abundance compensates for its lower quality. The resulting road map it not only more accurate and detailed, but can also be updated continuously as new information becomes available.

Australian Centre for Field Robotics, University of Sydney, Sydney 2006, NSW, Australia, e-mail: g.agamennoni, j.nieto, e.nebot @acfr.usyd.edu.au.

A. *Road maps and road safety*

Accurate road maps enable numerous road safety applications. An important area is mining safety. Every year a large number of accidents occur involving collisions between haul trucks and other resources such as light vehicles, graders and loaders. Many of them are fatal and incur a substantial loss of equipment and production. Amongst the numerous causes, the most important one is the operator's lack of situational awareness [6], [7], [8]. Open-pit mines are very dynamic and hostile environments. Drilling, loading and stockpiling areas are constantly changing and haul roads are continuously modified accordingly. Visibility is extremely limited by the bulk of the machinery and the cabin configuration, and is often severely impaired by harsh environmental conditions such as fog, dust, rain and snow. This poses a great safety hazard, especially in difficult areas such as intersections with tall banks and several different grades.

Fig. 1 shows a snapshot of a typical large-scale mine with many intersections, loading and drilling areas. During operation, heavy machinery such as haul trucks, graders, drills and shovels are closely interacting with light vehicles. Location "A" is a loading area where haul trucks queue as they wait to be loaded with ore by the shovel. Location "B" is an intersection where two roads with different grades converge Both roads are bounded on each side by a berm, which is taller than the light vehicles, meant to prevent trucks from veering off the edge[1]. Immediately below this intersection, the road is being widened to provide access to the drilling area. Once drilling has finished, the area will turn into a loading site similar to "A". As this figure shows, the opencast mine scenario can be very complex.

Predicting and preventing potential collisions requires more than just road maps. A comprehensive approach to risk assessment must incorporate information about the full standard paths of the vehicles within the map, including paths taken at the intersections. Fig. 2 shows an example map where each road is split into a pair of paths, one for each direction. Paths inside the intersection region reflect the route that vehicles normally take. Notice how each pair of paths connect to each other. This clearly shows that, even if a road map is available, what is really needed is the actual path that the haul trucks take. In the mine scenario, the latter turns out to be highly variable, since it changes according to modifications in the haul roads. The algorithm presented in this work accurately constructs the current path of the trucks

[1]Notice the relative size of light vehicles and trucks. A standard 4WD is less than 4 meters long and 2 meters high, whereas a haul truck is typically more than three times as large. In fact, the blind spot at the front of a haul truck is large enough to shadow a 4WD completely.

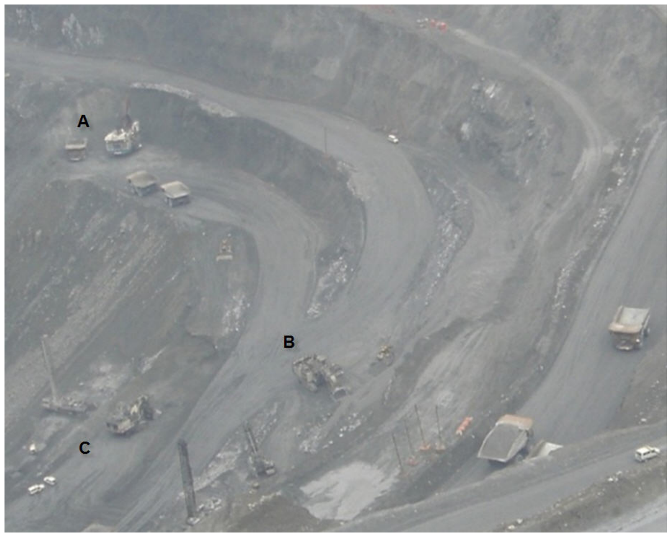

Fig. 1. Snapshot of the operation of a large opencast mine. Loading and drilling areas are marked with letters. See text for details.

in all of the mine including intersections. This is an essential component for evaluating risk in such complex and dynamic environments.

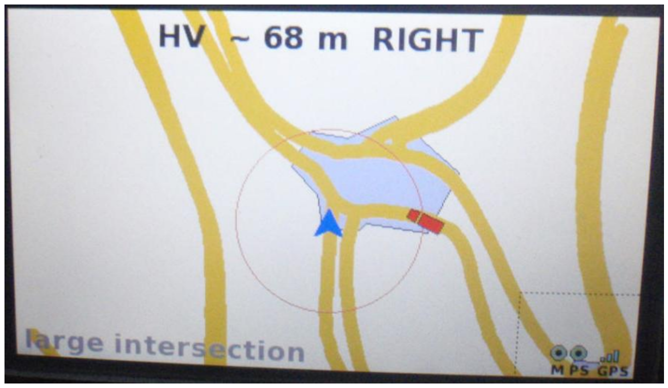

Fig. 2. Photograph of the operator interface module of a safety system. The module is fitted inside the cabin of a haul truck and provides the driver with situation awareness capabilities. Roads are depicted as thick yellow lines and a large intersection is represented as a light blue polygon. The blue arrow denotes the vehicle where the module is fitted and the red box represents a haul truck. Both vehicles are entering an intersection area. In this situation, knowing the set of possible paths for the vehicles is more important for the driver than the actual road map.

B. Related work

The problem of automatically generating road maps has received a lot of attention. Methods abound in the literature, and two main paradigms can be distinguished according to the type of data. On one hand, there has been significant effort placed on constructing road maps from aerial and SAR images. Some of these [9], [10] define a fully probabilistic model of the road network. Road generation is then cast as an inference problem, which is accomplished using numerical optimization or statistical sampling techniques. Also, the work in [5] presents a method that uses image processing tools to generate road maps from GPS data.

The second class of approaches are based purely on vector data. Some [2] assume the existence of an initial base map. This initial map is progressively refined by fusing it with information from GPS receivers. Other approaches assume no prior knowledge. For instance, in [2] the authors apply the sweep-line algorithm from the field of computational geometry to efficiently build a travel graph from vector data. In [4] the authors introduce an incremental graph-matching technique specifically designed for real-time applications. One of the approaches that is most relevant to the one presented here is the work of [3]. In this case, the strategy consists of placing road seeds and linking them together according to transition counts.

This paper is organized as follows. The fundamental components of the map building algorithm are presented in Section II and Section III. Results are presented in Section IV along with a discussion and a comparison to other mapping algorithms. Finally, Section V draws conclusions and discusses future research directions.

II. SAMPLING

This section and the next develop the map-building algorithm step by step in two separate parts. Each part is focuses on a particular processing stage. A basic outline of the two stages of the algorithm can be seen in Fig. 3. First, the road network is sampled at a number of nodes along the centerline. Afterwards, these nodes are incrementally linked together, yielding the final graph.

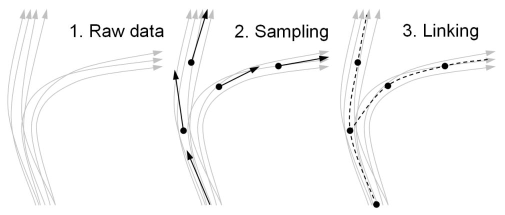

Fig. 3. Steps in the map building process. The algorithm takes raw position data as input and proceeds in two consecutive steps. During the first step, a set of nodes is constructed by sampling the road network at a series of points along the centerline. In the second step, the nodes are linked together to yield a directed graph.

A. Road centerline

Commercial digital road maps usually consist of an attributed undirected graph. Junctions or intersections are represented as vertices, whereas roads are depicted as a sequence of edges. Since the graph is undirected, paths possess no orientation and hence two-way roads are considered as a single path. The present approach will depart from this convention, adopting the view of [3], [4] instead. Roads will be regarded as a sequence of directed edges, essentially splitting each bidirectional road into a pair of unidirectional roads. Taking direction information into account will yield much better discrimination of the dominant directions in cluttered areas.

The road centerline is defined as the geometric space of all points that run through the middle of the road. Any given point on the centerline must be self-consistent in the sense that it must equal the statistical mean of all points on the road that project to it. In this sense, the centerline can be regarded as a principal curve [11]. As a matter of fact, the road inference procedure presented here is inspired on the polygonal line algorithm for principal curves.

Suppose that an estimate of the centerline at a given point is available. Consider that the centerline is estimated to pass through $\mathbf{p}$ tangent to $\mathbf{q}$ at some point. This pair of vectors define a plane π oriented in the direction of $\mathbf{q}$. Also, assume a set of position data traces is also given. Each trace $x = \{\mathbf{x}_n\}$ is composed of a finite sequence of position samples $\mathbf{x}_n$ ordered according to time. The sequence of points is interpolated by a curve $\mathbf{z}_n$, called the trajectory. Because trajectories must be curves and hence must be continuous, interpolation can be linear or polynomial.

The road centerline is estimated as follows. Let $z_n = \{\mathbf{z}_n(t)\}$ be the trajectory, parameterized by t, that interpolates the n-th position trace in the data set. Let T_n be the set of all parameter values in z_n whose corresponding points intersect π with strictly positive orientation,

$$T_n = \left\{ t : (\mathbf{z}_n(t) - \mathbf{p})^T \mathbf{q} = 0, \frac{d\mathbf{z}_n}{dt}(t)^T \mathbf{q} > 0 \right\} \quad (1)$$

The union of all the images under z_n and dz_n/dt,

$$X = \bigcup_n \mathbf{z}_n (T_n) = \{\mathbf{x}_k\},$$

$$Y = \bigcup_n \frac{d\mathbf{z}_n}{dt} (T_n) = \{\mathbf{y}_k\},$$

each contain a finite number of column vectors of the same dimension and hence can be arranged in the rows of a pair $\mathbf{X}$, $\mathbf{Y}$ of matrices.

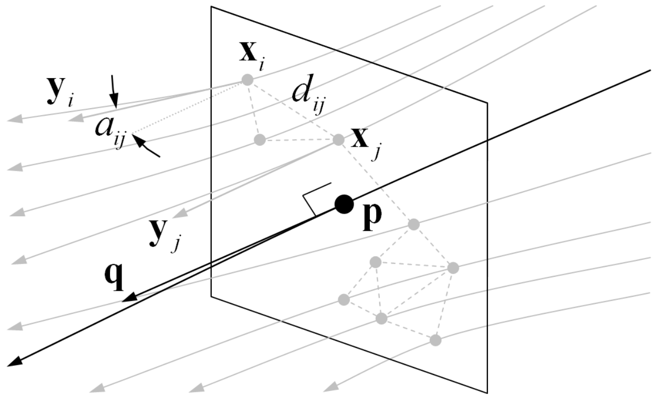

Fig. 4. Cross-section of the road centerline. Trajectories intersect the normal plane at a series of points, depicted as grey dots. Pairwise similarities are drawn between them to derive a weighted graph, represented as grey dashed edges. This graph is then used to cluster the points on the plane and construct a partition.

An illustration of the point sets can be seen in Fig. 4. The set X provides a picture of the cross-section profile of the road at $\mathbf{p}$. A road network is composed of many roads, each with its own centerline. It is not known beforehand which data trace follows which road. These correspondences must be inferred as part of the map-building process. However, it is certain that points in the same road will tend to lie close to each other, or in other words, they will tend to form clusters in the normal plane. The next step consists of performing clustering on π in order to derive a partition of X and Y into points that belong to $(\mathbf{p}, \mathbf{q})$ and points that do not.

B. Clustering

The dominant set framework [12] is a pairwise clustering method particularly effective at finding compact groups of points. To apply this method, a suitable similarity measure must be derived. Thorough experimentation led to the conclusion that distance and angle are the only two critical factors for clustering on the normal hyperplane. Let

$$d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|, \quad a_{ij} = \cos^{-1} \frac{\mathbf{y}_i \cdot \mathbf{y}_j}{\|\mathbf{y}_i\| \|\mathbf{y}_j\|}$$

denote the Euclidean distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$ in X, and the angle between their corresponding derivative vectors in Y, respectively (see Fig. 4). It was found by trial and error that the expression

$$w(i,j) \propto e^{-d_{ij}^2/\epsilon^2 + \kappa\,(1 - \cos a_{ij})} \quad (2)$$

yields very good results. This similarity function arises from the product of an isotropic Gaussian and a circular Von Mises density. Both ϵ and κ are positive scalars that control the concentration of the distribution around its mean.

These parameters account for uncertainty in the model. The first one corresponds to the standard deviation of the Gaussian distribution and has units of distance, while the second is the angular concentration of the Von Mises density[2] and is dimensionless. Both parameters quantify the scattering in position and the dispersion in direction. Increasing the value of ϵ, or decreasing the value of κ, equates to admitting a higher noise level in the data. However, their choice also involves a compromise between accuracy and noise immunity. Too large a value can result in significant errors, whereas a low value can lead to over-segmentation.

The similarity measure is used to build an undirected graph. Applying Eq. 2 to all elements of (X, Y) yields a similarity matrix. A cluster is extracted from the graph via a simple fixed point iteration (refer to [12] for details). This yields a a weight vector $\mathbf{w}$ having as many elements as X and Y and dwelling on the closed standard simplex. As explained in [12], the elements of the weight vector are nonnegative and sum to one. Its support $\sigma(\mathbf{w}) = \{k : w_k \neq 0\}$ is dominant with respect to the index set of X and Y. Furthermore, w_k reflects the extent to which the k-th point belongs to the cluster. That is, w_k can be interpreted as the belief that $\mathbf{x}_k$ and $\mathbf{y}_k$ belong to the current road. Hence the weight vector provides a measure of uncertainty in trace-to-road association.

[2]In practice, an equivalent and more intuitive parameter is used instead. The circular deviation δ relates to the angular concentration via

$$\delta^2 + \left(\frac{I_1(\kappa)}{I_0(\kappa)} \right)^2 = 1,$$

where I_j denotes the modified Bessel function of order j.

C. Closing the iteration

In order to close the iteration cycle, the centerline estimate must be updated. The update is performed using the weight vector obtained from the clustering procedure by assigning

$$\mathbf{p} \leftarrow \sum_i w_i \mathbf{x}_i = \mathbf{w}^T \mathbf{X}, \quad \mathbf{q} \leftarrow \sum_i w_i \mathbf{y}_i = \mathbf{w}^T \mathbf{Y}. \tag{3}$$

Notice how each element in the sum is weighted by its corresponding belief, meaning that each estimate is assigned the weighted average of all points that project to it. Therefore, upon convergence, $\mathbf{p}$ and $\mathbf{q}$ will be self-consistent with respect to X and Y respectively.

Algorithm 1 Centerline sampling routine.

```
1:  function SAMPLE($\{\mathbf{z}_n\}, \mathbf{p}, \mathbf{q}$)
2:      repeat
3:          $\hat{\mathbf{p}} \leftarrow \mathbf{p}, \quad \hat{\mathbf{q}} \leftarrow \mathbf{q}$
4:          Construct $T_n$, $X$ and $Y$ as in (1).
5:          for $\mathbf{x}_i \in X$, $\mathbf{y}_i \in Y$ do
6:              $w_i \propto$ SIMILARITY($\mathbf{x}_i, \mathbf{y}_i, \mathbf{p}, \mathbf{q}$)
7:              for $\mathbf{x}_j \in X$, $\mathbf{y}_j \in Y$ do
8:                  $W_{ij} =$ SIMILARITY($\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}_j, \mathbf{y}_j$)
9:              end for
10:         end for
11:         $\mathbf{w} =$ CLUSTER($\mathbf{W}, \mathbf{w}$)
12:         $\mathbf{p} \leftarrow \mathbf{w}^T\mathbf{X}, \quad \mathbf{q} \leftarrow \mathbf{w}^T\mathbf{Y}$
13:     until SIMILARITY($\hat{\mathbf{p}}, \hat{\mathbf{q}}, \mathbf{q}, \mathbf{q}$) $< \tau/2$
14: end function
```

Algorithm 1 summarizes the centerline estimation routine. This routine accepts an initial estimate as input and iteratively improves it until reaching a tolerance threshold τ. Function SIMILARITY evaluates the similarity in (2), and routine CLUSTER returns a fixed point of the replicator dynamic map. Deriving a convergence bound for this routine is nontrivial, the greatest difficulty being the fact that the data set is not finite[3]. Moreover, in some cases it may not converge. This happens for certain geometric configurations of the data that generate empty X and Y sets. Even so, in practice it has been observed that convergence is achieved over 99% of the time, with a tolerance of 10^{-3} rarely requiring more than four iterations.

So far, this section focused on a single point on the road centerline. It showed how an initial estimate can be refined to yield it self-consistent. In order to capture the full extent of the road network, a number of points $\{\mathbf{p}_m\}$ and $\{\mathbf{q}_m\}$ must be deployed along all of the traces. A simple but effective strategy consists of uniformly sampling the road network by taking each data trace in turn and placing estimates regularly along its trajectory. Once all the estimates are in place, Algorithm 1 is called for each of them in turn. Upon completion, a set of self-consistent centerline points and a corresponding set of weight vectors is obtained. Weight vectors quantify the uncertainty in the association of roads to traces and are of central importance in the second stage of the algorithm. This stage deals with how to link points together to generate the network topology and is described in the following section.

[3]While data traces are finite, the trajectories that interpolate them contain uncountably many points.

III. Linking

The preceding section described the first stage of the algorithm. During this phase, a number of estimates of the road centerline were drawn. These estimates will now serve as anchor points upon which the rest of the map will be built. In the next step they will be linked together to capture the network topology. This section explains the details.

A. Transition matrix

Linking is performed by connecting points to one another with directed edges. Together, the set of nodes and edges form a directed graph that comprises the skeleton of the road map. The decision of whether two given points should be connected or not is based on a measure that quantifies the strength of the connection. This measure, called the transition likelihood, reflects how certain it is for that edge to form part of the graph. A high likelihood means the edge is bound to belong to the graph, whereas a low likelihood indicates that the corresponding points are weakly coupled and should not be connected.

Recall the intersection set defined in (1). This set is composed of all parameter values in trajectory n that map to points on the normal plane with strictly positive orientation. For each $t_k \in T_n$, it is said that the n-th trajectory *passes* through the centerline point at time t_k if its corresponding weight w_k is nonzero. Let $\mathbf{A}$ be the transition count matrix for all points. That is, A_{ij} equals the number of times a trajectory passed through point i and subsequently through j. The matrix is row-normalized,

$$A_{ij} \leftarrow \frac{A_{ij}}{\sum_k A_{ik}}, \tag{4}$$

so that its elements can be interpreted as transition probabilities. This yields the transition matrix.

B. Linkage criterion

The transition matrix provides a measure of the strength of each connection. Its support $\sigma(\mathbf{A}) = \{(i,j) : A_{ij} \neq 0\}$ is a set of edges connecting nodes together, each edge weighted by its corresponding transition likelihood. Due to noise and ambiguity present in the association of traces to nodes, $\sigma(\mathbf{A})$ may contain edges that are superfluous. Therefore, instead of incorporating all of them, only a subset $E \subseteq \sigma(\mathbf{A})$ of non-superfluous edges is selected. This set constitutes the edge set of the final graph.

The linking problem can be formulated as follows. Let $\{G_E\}$ be a family of graphs parameterized by their edge set. Each member is a directed weighted graph $G_E = (V, E, w)$. All members share a common vertex set V and a unique weight function defined as $w : (i,j) \mapsto A_{ij}$. Also, every edge set is constrained to range over the support of $\mathbf{A}$,

$$E \subseteq \sigma(\mathbf{A}) = \{(i,j) \in V \times V : A_{ij} > 0\},$$

meaning that members of $\{G_E\}$ cannot have more connections than the original graph $G_{\sigma(\mathbf{A})}$.

The search for the most suitable graph is cast a maximization problem. Specifically, it is formulated as the problem of finding the graph with the largest overall weight,

$$\max_{E\subseteq\sigma(\mathbf{A})} \prod_{(i,j)\in E} A_{ij}, \tag{5}$$

amongst all possible subgraphs of $G_{\sigma(\mathbf{A})}$. As it is, this optimization is ill-posed. Because all elements of $\mathbf{A}$ lie on the unit interval, the objective function monotonically decreases with the cardinality of the edge set. Therefore, it can be trivially maximized by setting $E=\emptyset$. To avoid such vacuous solution, a set of constraints is added.

The idea is to restrict the set of possible graphs so that connectivity is preserved. That is, if two vertices of $G_{\sigma(\mathbf{A})}$ are connected, then the corresponding vertices of G_E must also be connected. Mathematically, this constraint can be expressed in terms of the transitive closure of G_E,

$$\lambda_E(i,j) \geq \lambda_{\sigma(\mathbf{A})}(i,j) \quad \forall\, i,j \in V, \tag{6}$$

where λ_E is the local edge connectivity defined in (7). The constraint set ensures that the optimum graph does not become disconnected as edges are pruned.

The complete minimization problem consists of solving (5) with respect to E, subject to (6). This problem is combinatorially hard and there is no general algorithm for solving it efficiently. However, it is possible to find a feasible suboptimal solution by means of a simple greedy strategy. By performing a locally optimal search over possible subgraphs, the worst-case complexity becomes linear in the number of vertices and quadratic in the number of edges.

C. Suboptimal linkage

Having defined the objective function, the next step consists of deriving an efficient method for solving it. Notice that each edge $e=(i,j)$ must be the path with the largest weight between vertices i and j. This suggests a simple iterative method for monotonically increasing the overall weight. Namely, starting from the edge e with the largest transition likelihood, the shortest path between its endpoints is computed. If this path is not equal to e, then the edge still satisfies the constraints in (6) and hence yields a feasible graph. Therefore, removing it increases the overall weight in (5), bringing the graph closer to the optimum.

Algorithm 2 Linking routine.

```
1: function LINK($\mathbf{A}$)
2:     $E \leftarrow$ SORT($\sigma(\mathbf{A})$)          ▷ Sort in descending order.
3:     for $(i,j) \in E$ do
4:         $E \leftarrow E - \{(i,j)\}$
5:         if $\lambda_E(i,j) < \lambda_{\sigma(\mathbf{A})}$ then          ▷ Test connectivity.
6:             $E \leftarrow E \cup \{(i,j)\}$
7:         end if
8:     end for
9: end function
```

Pseudo-code for the linking routine can be found in Algorithm 2. Routine SORT arranges the edges in descending order according to their likelihood. Once sorted, each edge is then tested by removing it from the graph and checking how its connectivity changes. If it decreases with respect to the complete graph, the edge is incorporated back again into the edge set. Otherwise, it is discarded. A skeleton of the map is obtained once the linking routine has converged. This skeleton is the final representation of the road network. It consists of a set of road centerline estimates linked together by the directed edge set.

IV. Results

This section shows experimental results. A large and rich data set is used to test the performance of the algorithm in several different settings. Comparisons between this and other approaches are also presented.

A. Experimental data

Over 10 megabytes of raw double-precision data was used as a benchmark for the algorithm. Position data was collected at an opencast mine in Western Australia and corresponds to five days of operation of more than 15 resources. Around 400 position traces were constructed with almost no preliminary processing. The only preprocessing performed is a simple outlier rejection method based on distance. Namely, consecutive position samples that lay more than 100 meters apart were considered as outliers and removed from the data.

Fig. 5 shows the final road map overlaid on an aerial photograph of the mine. The photo spans an area of 10.5 by 3.5 kilometers. The only two factors that strongly affect the outcome of the algorithm are ϵ and κ. The standard deviation ϵ was set to 15 meters, roughly equal to the width of the haul trucks. The circular dispersion κ was selected as $1/8$, corresponding to $\delta=\pi/4$ radians, after a small number of tuning instances. Initial point estimates of the road centerline were placed 30 meters apart.

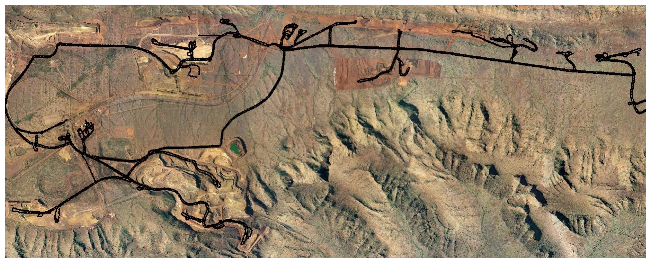
Fig. 5. Final road map superimposed on an aerial photograph of the mine. The map is drawn in black. Although details are not appreciable at this scale, it serves to contemplate the magnitude of the problem.

The algorithm was implemented in MatLab® and tested on an Intel® Core™ Duo CPU with a 2.33 GHz processor and 2 Gb of RAM. Execution took 83 minutes, with Algorithm 1 accounting for almost all of the running time. Over 4100 seeds were placed during sampling, giving an average of roughly 1.2 seconds per sample. This amount is not excessive given the dimensions of the map and considering that no specialized storage structure was used for implementing Algorithm 1.

B. Details of the inferred road map

Fig. 6 shows a typical road junction from the data set. Position traces exhibit a considerable amount of dispersion along the cross section. Notice how drivers tend to take either very sharp or very wide turns and often cross over to the opposite lane. A similar scenario is depicted in Fig. 7, which shows what is possibly the most difficult intersection area found in the data set. Here, the high degree of clutter makes inference a very challenging task. Even so, in both cases the mapping algorithm is able to reconstruct the network.

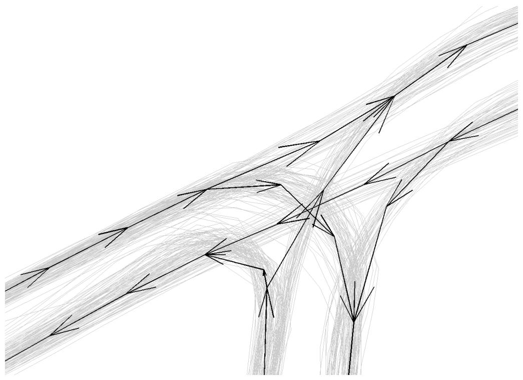

Fig. 6. Typical road junction inferred from the data set. The road map is represented as a directed graph. It is depicted as a set of black arrows, each one connecting two centerline point estimates together.

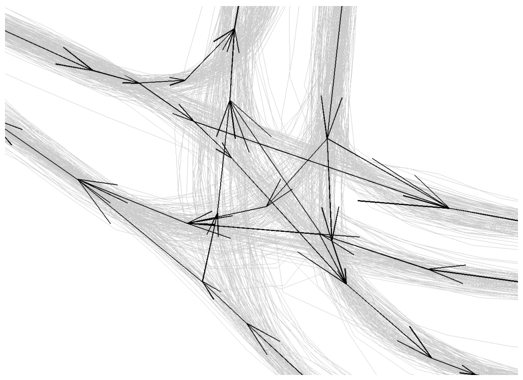

Fig. 7. Complex intersection area from the data set. This area is particularly difficult to infer because of the large amount of clutter in the middle.

A different situation is presented in Fig. 8. Here, a heavily traveled road forks into two at a junction. Examination of the data reveals that the road was blocked for a period of time. A temporary obstruction, possibly another vehicle, caused truck drivers to swerve and cross change lanes in order to avoid a collision. Still, the road map was correctly inferred. The centerline follows the most intuitive path and does not appear biased towards the middle of the road. This robustness stems from the update equation in the road sampling scheme.

Even when data does not abound, the approach remains robust to noise and outliers. The crossing in Fig. 9 contains only a small amount of data traces, some of which

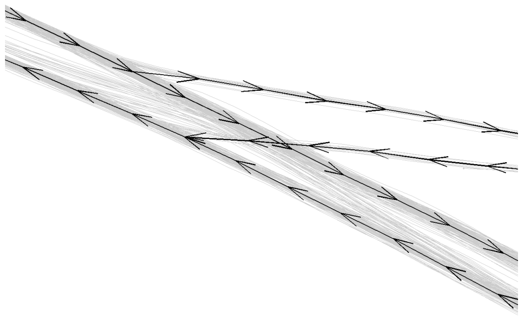

Fig. 8. Heavily traveled junction with an obstruction. A two-way road splits into two at a junction. An obstruction was present at the intersection, temporarily blocking the heavily transited road.

are relatively noisy. Trucks approaching the crossing tend to reduce their speed almost to a complete halt. Because measurements are less accurate at lower speeds[4], this data is often noisier than normal. However, the approach still manages to accurately trace the centerline and capture the underlying topology.

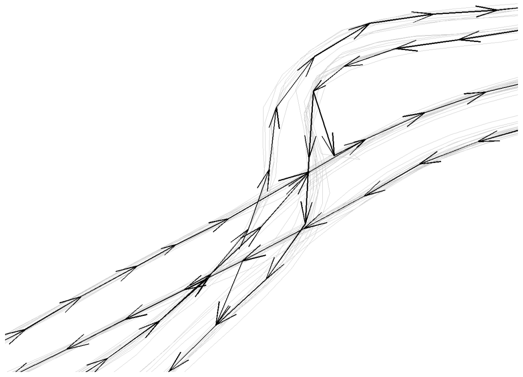

Fig. 9. Road crossing with sparse data. Two double-lane roads intersect each other, creating a narrow crossing. Data in this area is less abundant and tends to be noisier.

It is also interesting to test the algorithm in particular regions of the mine where vehicles take multiple paths. Fig. 10 shows a wide tipping area where trucks drop their load. Drivers back against the berm and dump the ore over the edge. Data traces fan out due to trucks tipping on different spots along the perimeter. The road appears over-segmented because the algorithm cannot find any dominant direction. This is not surprising, since it is specifically designed to detect coherence within the data.

C. Comparison with other methods

Two other road mapping methods were also coded. The first is the image-based algorithm of [5]. Segment histogram resolution was set to 2 meters, which is more than enough given the dimensions of the trucks. A linear squared exponential filter was used to smooth the histogram. After some tuning, a filter width of 1.5 meters was found to give the

[4]Speed estimation in a GPS unit is more accurate at higher speeds, when the ratio of positional error to positional change is lower.

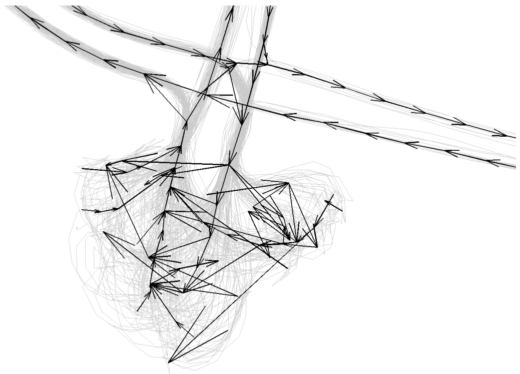

Fig. 10. Large open tipping area. The approach presented in this article does not perform well in this case. Situations like these could be handled with techniques specifically aimed at detecting these areas.

best results. The segmentation threshold was also selected by trial and error, being 0.1 an acceptable value. Execution required 27 minutes in total.

The result of applying the method of [5] is shown in Fig. 11. The approach does not use heading information and hence the resulting graph is undirected. The topology of the intersection is captured at a large scale. However, the inferred map lacks fine-grained information about road direction and vehicle paths. During parameter tuning, a compromise was made between granularity and over-segmentation. Smaller values for the filter width and segmentation threshold reveal more fine-grained information but at the same time yield a much noisier map.

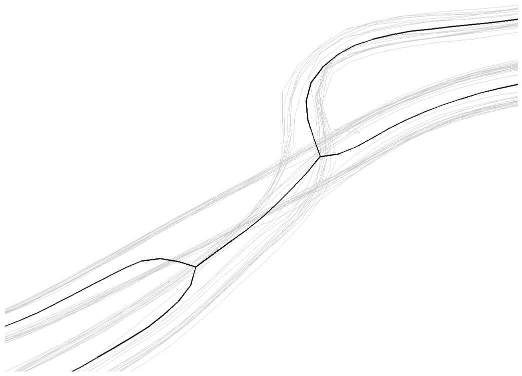

Fig. 11. Road crossing with sparse data inferred by the method of [5]. Roads are represented as undirected black segments instead of arrows.

The second method is the one described in [3]. Out of all the techniques in the literature, this is most similar to the one presented here. It constructs the map in a similar fashion, by placing seeds along the traces and linking them together according to transitions. Complexity was reduced by removing the lane structure inference step because the mine environment is much less structured than an urban highway. Roads do not split into lanes and hence clustering lane offsets is unnecessary. The same parameters were used for sampling the road centerline, namely $\epsilon = 15$ meters and $2\pi\delta = \pi/4$ radians, and seeds were placed 30 meters apart. Overall, the algorithm took 52 minutes to process the data.

Fig. 12 shows the sparse road crossing inferred by the algorithm in [3]. Although it reconstructs most of the crossing successfully, one of the intersections appears displaced from its true position. The two roads originating from the north-east corner intersect at a very narrow angle. The algorithm mislays the intersection point and places it further ahead from its real location. This mistake is caused by the clustering routine, which fails to discriminate between both roads until they are a long distance from each other.

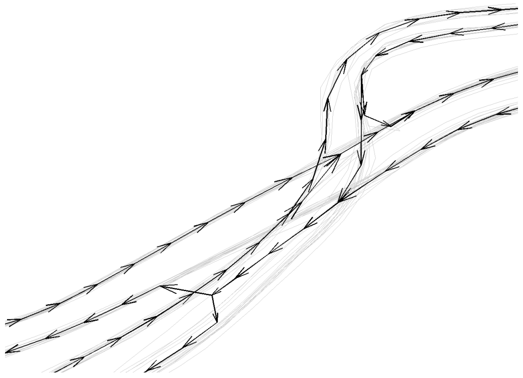

Fig. 12. Sparse road crossing extracted by the method of [3]. The algorithm successfully recovers a large portion of the crossing but misplaces one of the intersection points.

The same happens in Fig. 13. Here, the method somewhat recovers the shape of the junction but misplaces both the road split and merge. As before, the clustering routine is unable to recognize the true position at which the roads meet. Furthermore, it biases one of the lanes in the main road. The northbound lane on the heavily transited road is slanted inwards, bending towards the middle of the road. Again, this bias stems from the clustering routine.

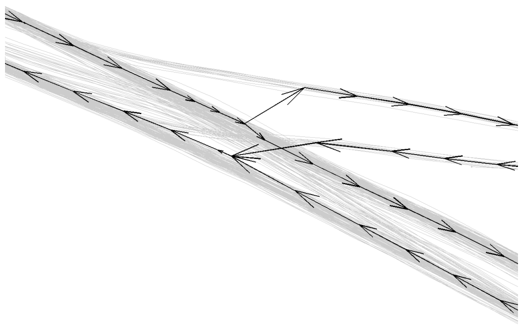

Fig. 13. Method of [3] applied to the obstructed road junction with heavy traffic. Again, the algorithm recovers the overall topology but wrongfully places lane splits and merges.

Last of all, Fig. 14 shows the performance on the complex intersection. Here, even the inferred topology is not entirely accurate. There are a number of redundant edges that should be discarded or fused together. Unfortunately, the simple linkage criterion used in [3] does not take edge redundancy into account. It links every pair of endpoints with at least one nonzero transition, failing to acknowledge the possibility of spurious connections.

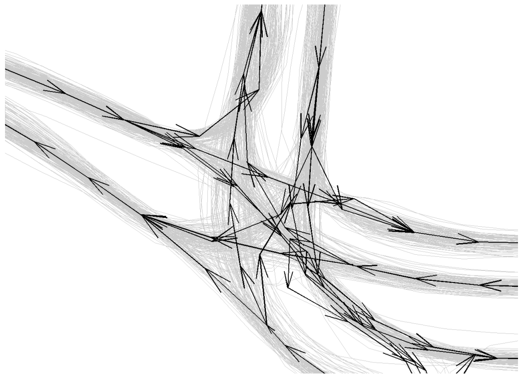

Fig. 14. Algorithm of [3] applied to the complex intersection. In this case the approach is not able to correctly recover the topology.

V. Conclusion and Future Work

This paper described an approach for automatically inferring high-precision road maps from position data. The algorithm was tested with data from an operating mine. Experimental results clearly show that the desired performance is achieved. Although it is not possible to quantify the accuracy of the results since there is no absolute ground-truth, it is arguable that the ground truth can be inferred by visual inspection of the trajectories of the vehicles. The inferred roads agree with intuition and they follow what a human user would draw. The algorithm also succeeds at capturing the network's shape and topology. In addition, it compares favorably to other existing methods in the literature.

The mapping algorithm is scalable to large data sets. Experiments show that the algorithm obtains high-quality maps for a medium-sized mine in approximately two hours without any code optimization. Furthermore, it is not limited by the size of the data. Even though no experiments were conducted using data spanning more than five days, this is not due to memory constraints. The reason is that roads change relatively frequently, and changes are sometimes visible within the time scale of one week. Also, notice that the dimensionality of the data is not present in any of the equations. The algorithm can also be applied using height data to construct three-dimensional maps.

Current research is looking into ways of improving efficiency and linking performance, as well as performing place recognition. With respect to place recognition, it was seen that performance downgrades in large open areas such as tipping sites. This feature could be used to detect distinct places. Also, previous approaches have used position information [13], [14], [15] to recognize them. These are all valuable tools that could be used to complement the mapping process and pose promising directions for future research.

Appendix I
Transitive Closure of a Graph

Transitive closure is defined in terms of binary relations. For a weighted graph, several possible definitions exist. The one adopted here can be expressed in terms of local graph connectivity. Let $G = (V, E, w)$ be a weighted graph with vertex set V and edge set $E \subseteq V \times V$ equipped a weight function $w : E \rightarrow [0, 1]$. Two vertices $i, j \in V$ are said to be connected if there exists a path starting in i and ending in j. A set of paths are said to be edge-independent if no two of them share a common edge.

Let $P(i, j)$ be the set of all edge-independent paths between $i, j \in V$. Local edge connectivity is defined as

$$\lambda(i,j) = \sum_{p \in P(i,j)} \prod_{e \in p} w(e). \tag{7}$$

The closure of G is defined as the graph $\bar{G}$ with weight function λ and edge set equal to the closure of E.

Connectivity can be evaluated via simple matrix algebra. Let $\mathbf{A}$ be the adjacency matrix of G. It can be shown that the adjacency matrix of $\bar{G}$ is $\mathbf{A} + \ldots + \mathbf{A}^n$, where n is the number of vertices. Although time complexity $O(n^3)$, in practice it can be made much smaller. Algorithm 2 tests edge redundancy by checking whether the total weight of all paths connecting i and j is larger than a given threshold. Hence, running time is much smaller in practice since the threshold is often exceeded very early during the query.

References

[1] C. Wilson, S. Rogers, and S. Weisenburger, "The potential of precision maps in intelligent vehicles," in *Proceedings of the 1998 IEEE International Conference on Intelligent Vehicles*, 1998, pp. 419–422.

[2] S. Edelkamp and S. Schrödl, "Route planning and map inference with global positioning traces," *Computer Science in Perspective: Essays Dedicated to Thomas Ottmann*, pp. 128–151, 2003.

[3] S. Schroedl, K. Wagstaff, S. Rogers, P. Langley, and C. Wilson, "Mining gps traces for map refinement," *Transactions on Data Mining and Knowledge Discovery*, vol. 9, no. 1, pp. 59–87, 2004.

[4] R. Bruntrup, S. Edelkamp, S. Jabbar, and B. Scholz, "Incremental map generation with gps traces," in *Proceedings of the 2005 IEEE International Conference on Intelligent Transportation Systems*, Sept. 2005, pp. 574–579.

[5] J. Davies, A. Beresford, and A. Hopper, "Scalable, distributed, real-time map generation," *IEEE Transactions on Pervasive Computing*, vol. 5, no. 4, pp. 47–54, Oct.-Dec. 2006.

[6] S. Worrall and E. Nebot, "Using non-parametric filters and sparse observations to localise a fleet of mining vehicles," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 509–516.

[7] ——, "A probabilistic method for detecting impending vehicle interactions," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 1787–1791.

[8] S. Worrall, "Providing situation awareness in complex multi-vehicle operations," Ph.D. dissertation, University of Sydney, July 2009.

[9] M. Amo, F. Martinez, and M. Torre, "Road extraction from aerial images using a region competition algorithm," *IEEE Transactions on Image Processing*, vol. 15, no. 5, pp. 1192–1201, May 2006.

[10] R. Stoica, X. Descombes, and J. Zerubia, "A gibbs point process for road extraction from remotely sensed images," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 121–136, 2004.

[11] T. Hastie and W. Stuetzle, "Principal curves," *Journal of the American Statistical Association*, vol. 84, pp. 502–516, 1989.

[12] M. Pavan and M. Pelillo, "Dominant sets and pairwise clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 167–172, 2007.

[13] D. Ashbrook and T. Starner, "Using gps to learn significant locations and predict movement across multiple users," *Personal and Ubiquitous Computing*, vol. 7, no. 5, pp. 275–286, 2003.

[14] L. Liao, D. Fox, and H. Kautz, "Extracting places and activities from gps traces using hierarchical conditional random fields," *International Journal of Robotics Research*, vol. 26, no. 1, pp. 119–134, 2007.

[15] G. Agamennoni, J. Nieto, and E. Nebot, "Mining gps data for extracting significant places," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, 2008.