Robust Inference of Principal Road Paths for Intelligent Transportation Systems

Gabriel Agamennoni, *Member, IEEE,* Juan I. Nieto, *Member, IEEE,* and Eduardo M. Nebot, *Member, IEEE*

***Abstract*—Over the last few years, electronic vehicle guidance systems have become increasingly more popular. However, despite their ubiquity, performance will always be subject to the availability of detailed digital road maps. Most current digital maps are still inadequate for advanced applications in unstructured environments. Lack of up-to-date information and insufficient 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 amounts of data combining both modalities will be available in a near future. The approach presented here draws on machine learning techniques and processes logs of position traces to consistently build a detailed and fine-grained representation of the road network by extracting the principal paths followed by the vehicles. Although this work addresses the road building problem in dynamic environments such as open-pit mines, it is also applicable to urban environments. New contributions include a fully unsupervised segmentation method for sampling roads and inferring the network topology, a general technique for extracting detailed information about road splits, merges and intersections, and a robust algorithm that articulates these two. Experimental results with data from large mining operations are presented to validate the new algorithm.**

***Index Terms*—Digital road maps, data mining, machine learning, GPS, road safety.**

I. Introduction

In the last few years, the wide availability of commercial digital road maps has enabled a large number of vehicle navigation and guidance applications. Commercial vector maps have managed to cover to a great extent the major road networks in urban areas, achieving a level of accuracy in the order of a few to a few tens of meters. The combination of these digital maps with precise positioning systems has allowed the development of numerous in-vehicle applications that provide navigation aid and driving assistance. However, there exists a much wider range of potential applications beyond those aimed at enhancing driver convenience. As shown in [1], [2], [3], detailed road maps could also be used to improve road safety by means of lane keeping, rollover warning, obstacle detection and collision avoidance systems.

A fundamental limitation of commercially available road maps is their lack of fine-grained information. Standard high-end road safety applications require not only accuracy but also a superior level of detail, and currently this can only be achieved by the use of probe vehicles and manual surveying methods. The resulting map is therefore very expensive and time-consuming to build and update. Many authors [4], [5], [6], [7] propose that, instead of using a single dedicated probe to generate such an enhanced map, a large number of inexpensive low-precision information sources could yield similar or even better results. With the emergence of low-cost positioning devices and the accelerated development of wireless communication systems [8], integrated data gathering and processing becomes feasible at a very large scale. This allows large volumes of position data to be recorded and used for learning or updating the road map. Even though the data is polluted with noise and outliers, its abundance compensates for its lower quality. The resulting road map is not only more precise and more refined, but can also be updated continuously as new information becomes available.

An important distinction is to be made between the usual notion of a road map and the enhanced maps which are the focus of this article. Commercially available GPS devices usually contain a digital map which they use for localization and for providing directions to the user. However, their lack of detail renders them ineffective for road safety applications. In contrast, the class of refined maps that are the focus of the present work do have a lot potential when it comes to safety. These enhanced maps are much richer in detail and contain far more information. They not only describe the road geometry and the network topology but also contain statistics about the way agents move.

In order to distinguish between the two classes of maps, the term *Principal Road Path* map is introduced here. Principal road path (PRP) maps will be used hereafter to refer to the class of enhanced maps previously described. The name emphasizes the importance of typical driving behavior in road safety applications. Its meaning as well as its motivation will be explained in more detail in the following sections.

A. Principal road paths and road safety

PRP maps enable numerous road safety applications. For instance, an adaptive cruise control system could operate basing its decisions on curvature and elevation in order to minimize rollover risk. Also, a high degree of map detail would enable precise lane-level tracking for a lane departure warning mechanism and even fault detection for inferring obstructed sections in a road. Several degrees of driver interaction are possible, ranging from unobtrusive driving assistance to full automatic control. Also, wireless communication systems could allow information to be shared across multiple nearby agents. This would enable vehicles to incorporate context-aware systems, and eventually to develop cooperative decision-making schemes.

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

Manuscript received April 19, 2009; revised January 11, 2010.

An important area of application of PRP maps is mining safety. Every year there are a large number of accidents involving collisions between haul trucks and other mine resources such as light vehicles, graders and loaders. Many of them result in fatalities and significant loss of equipment and production.

Previous work has shown the importance of situational awareness in mining safety [9], [10]. To predict and prevent potential collisions, a safety system requires more than just a road map. A comprehensive approach to risk assessment must incorporate information about the typical behavior of the vehicles within the map, including the paths they usually take at each intersection. This information is very important and can be represented in a PRP map.

B. Related work

There has been a significant amount of work on autonomous road map construction using image processing techniques. For the problem of road extraction from aerial and SAR images, the interested reader should turn to [11], [12], [13] and the references therein. A few of these approaches [14], [15] define a fully probabilistic model of the road network. The problem of road generation is then cast as performing inference on the model, which the authors accomplish using either numerical optimization or statistical sampling techniques. Also, the work in [7] proposes a method where image processing tools are applied to vector data. After constructing a two-dimensional histogram from GPS data logs, the authors recover the skeleton of the network by means of simple linear filtering and morphological operations.

Approaches based purely on vector data also abound in the literature. Some of these [16], [4] assume the existence of an initial base map, consisting of a commercially available road map or constructed from a grey scale image [17]. From this initial estimate, the authors progressively refine the map by fusing it with GPS information. Other approaches build the map from the start assuming no prior knowledge. For instance, the authors of [4], [18] draw on the field of computational geometry and use the sweep-line algorithm to efficiently build a travel graph from vector data. Others [19] construct the graph by defining a clustering scheme to link GPS traces together. The approach introduced in [6] defines a graph-matching technique that is incremental and specifically designed for real-time applications. One of the approaches that is most relevant to the one presented here is the work of [5]. In this case, the strategy consists of placing road seeds and linking them together according to transition counts.

This article is organized as follows. Section II presents the road model utilized throughout the rest of the paper. The fundamental components of the map building algorithm, Sampling and Linking, are presented in sections III and IV respectively. Results are presented in Section V along with a thorough discussion and a brief comparison to other mapping algorithms. Section VI draws conclusions and discusses future research directions.

II. Road Model

This section explains the road model adopted throughout this work. Before explaining the algorithm itself, it is necessary to lay the foundations by setting forth in words the way a road network is conceived. First, the structure that is chosen for representing the map is described. Then, each of its constituent elements is explained in turn. This discussion serves as a starting point for developing the algorithm in full depth and leads seamlessly into Section III, where all the details are given.

A. Representing the network

Commercial digital road maps usually consist of an attributed undirected graph. Junctions or intersections are represented as vertices with degree[1] greater than two, whereas roads are depicted as a sequence of edges. Since the graph is undirected, a path possesses 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 [5], [6] instead. Specifically, roads will be regarded as a sequence of directed edges, essentially splitting each bidirectional road into a pair of unidirectional road paths. Doing so will yield much better discrimination of the dominant directions in cluttered areas, as will be shown later on.

Additional attributes can be defined to complement the graph. Geometric features such as width and curvature can be present as part of the map description. These features are essential for road safety applications because they provide detailed information about the local geometry. Likewise, label attributes such as “junction” and “bidirectional” can also prove useful since they give a qualitative high-level characterization of the network. There is a wide variety of attributes that can be defined. Many of these encode important information regarding the way vehicles move, or are supposed to move, as they traverse the network. They impose restrictions on vehicle behavior, and hence can be extremely valuable for assessing risk. The end result is a structure that comprises not only the skeleton of the roads and their interconnections, but also a rich combination of edge- and vertex-level attributes containing a wealth of fine-grained information.

The core of the map structure lies in its directed graph. The graph constitutes the skeleton of the network. It also induces a representation of the road paths. Specifically, a finite sequence of connected nodes on the graph can be seen as a sequence of samples of a particular road path. The underlying road path would then be a curve that interpolates the nodes on the sequence. Conversely, from a generative point of view, the principal road path may be regarded as a hidden generating curve that maps distances along the road to positions in three-dimensional space.

It is important to distinguish between the graph itself and its corresponding node set. A node as such is a real-valued point that marks a physical location on the map, whereas a graph is the mathematical object that links all the nodes together. As already mentioned, direction information becomes extremely

[1]The degree of a vertex is the number of edges incident on it.

valuable when dealing with cluttered regions. For this reason, it is already incorporated as part of the map representation by directly working in joint position-velocity space. That is, each node is considered as being composed of six real numbers; the (x, y, z) position triple and the (u, v, w) velocity triple. As a consequence of this construction, all of the nodes, and hence all road traces, not only lie on the principal road path but are also tangent to it.

B. Principal curve

The concept of a principal road path can be defined concisely in mathematical terms. Specifically, it can be seen as the *principal curve* of a set of position data traces. Thus the motivation behind the term PRP becomes clear. The remainder of this section is set apart for the purpose of defining the principal curve. The intuition behind this concept and its relationship to the principal road path will be explained in detail in the next section.

Definition 1. Let $[a, b] \subset \mathbf{R}$ be a compact connected subset of the real line. A real, d-dimensional parametric curve c is a continuous mapping $c : [a, b] \to \mathbf{R}^d$.

Definition 2. Let C^d be the set of all real, d-dimensional curves. A reparameterization of a curve $c \in C^d$ is a surjection $r : D(c) \to I$. Notice that, in general, the domain $D(c)$ of c and the image I of r need not be equal.

The following is a definition of a principal curve. It generalizes the one in [20] to the case where the data contains temporal information. In this case, instead of a set of points in Euclidean space, the data is taken to be a set of curves. It can be easily verified that this definition reduces to the original definition in the limit where all curves have zero length.

Definition 3. Let p be a probability measure on C^d. The curve $\gamma \in C^d$ is a principal curve of p if it is self-consistent in all of its domain, that is if $\forall\, t \in D(\gamma)$,

$$\gamma(t) = E\left[c(u) | r_c(u) = t\right], \tag{1}$$

where $r_c : D(c) \mapsto D(\gamma)$ is a reparameterization of $c \sim p$.

The mapping r_c projects any given curve drawn from p onto the principal curve. It yields a matching of c to γ by associating every point $c(u)$ with another point $\gamma(t)$, where $t = r_c(u)$. Notice that r_c must be a surjective function and therefore every point on the principal curve must originate from a corresponding point on c.

The reparameterization plays a key role in Definition 3. The self-consistency property in Equation 1 states that every point on γ must equal the statistical mean of all points on any given curve that project to it. Projection is achieved by application of the mapping r_c and can be chosen according to an optimality criterion. Namely, it can be chosen to minimize

$$r_c = \arg\min_r \delta\left(c \circ r, \gamma\right),$$

where $\delta : C^d \times C^d \mapsto \mathbf{R}^+ \cup \{0\}$ is a distance metric, such as the Fréchet distance [21] or the longest common subsequence [22] distance, that measures the dissimilarity between two given curves.

In practice, the distribution p is seldom known. Instead, only a finite data set is available. Each element of the data set usually comprises a sequence of data points of the same dimension. For example, when reconstructing road maps from vector data, the data set usually consists of a set of position traces. After linear interpolation, these traces become polygonal line curves. Evaluating the Fréchet distance for polygonal line curves can be done efficiently, as shown in [21], and yields the matching function r_c as a byproduct. Therefore, estimating the principal curve of the data set could be accomplished efficiently via a straightforward extension of the polygonal line algorithm of [23] or the k-segment version of [24] to curvilinear data. However, as will be shown in Section III, this method is only adequate for the case of a single isolated road. When multiple roads coexist, the method is no longer applicable due to the data association problem that arises.

III. Sampling

This section and the following one develop the map-building algorithm step by step in two separate parts. Each part focuses on a particular processing stage. The diagram in Figure 1 shows an elementary description of the algorithm. First, the road network is sampled at a number of nodes along the principal road paths. Afterwards, these nodes are linked together to produce the final graph. The present section describes the first stage of the algorithm, called the sampling stage.

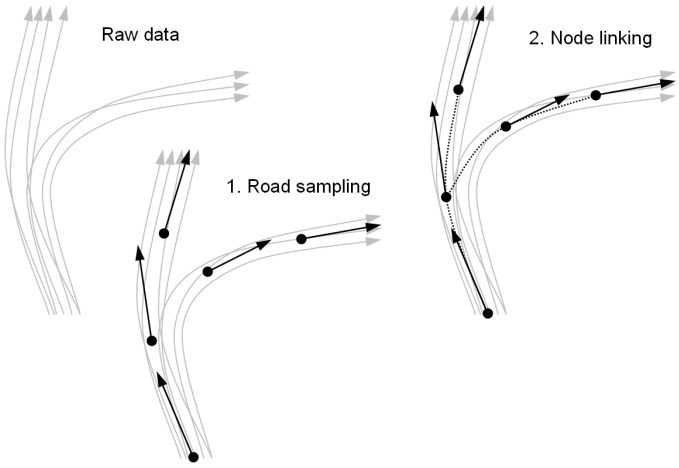

Fig. 1. Steps in the map building process. The algorithm takes raw position data as input and proceeds in two consecutive steps. During the first step, a set of nodes is constructed by sampling the road network at a series of points along the principal road paths. During the second step, the nodes are linked together to yield a directed graph.

A. Sampling the principal road path

The first step consists of drawing a finite number of samples from the set of principal road paths that constitute the network. In order to accomplish this, an appropriate sampling scheme must be devised based on Definition 3. This definition states that the principal road path is composed by the set of all points that run through the middle of the road path. Any given point on this set must be self-consistent in the sense that it must be

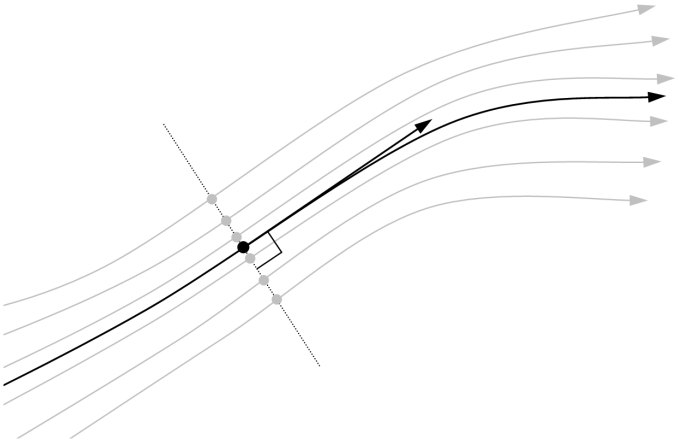

Fig. 2. Self-consistency property of the PRP. Light grey arrow-tipped lines denote position traces along a given road, whereas the black line represents the principal road path. Light grey dots are the points along the traces that project onto the black dot, which is a sample from the principal path. By definition, the black dot is equal to the statistical mean of the grey dots.

equal to the mean of all points on the road projecting onto it. Figure 2 illustrates this idea.

A road network is composed of many roads. If it was somehow known which vehicle follows which road, then sampling the principal road paths would be straightforward. In this case a simple method, based on the same principle as the polygonal line algorithm for principal curves [23], would suffice. Such method would proceed by taking one road at a time and iteratively estimate the underlying principal curve via a series of smoothed projections. However, this knowledge is rarely available. In general, the correspondences between vehicle paths and road paths are not given. As a matter of fact, deriving them turns out to be one of the most critical aspects of map building. A robust mapping algorithm must be able to account for the ambiguity that arises when attempting to associate position traces to roads.

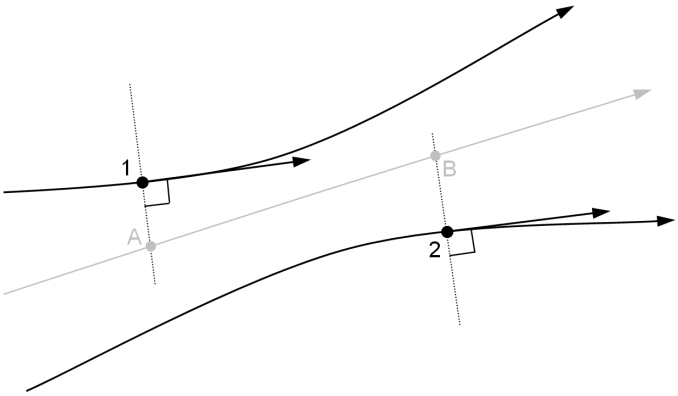

Fig. 3. Ambiguity when assigning position traces to road paths. Symbol convention is the same as in the previous figure. Here, points A and B must be assigned to either path 1 or 2. Because at this point it is uncertain which association is the correct one, soft assignments are computed instead and the final decision is deferred until enough information is gathered.

Ambiguity can be handled by making soft assignments of traces to roads. To exemplify this, imagine the situation depicted in Figure 3 is presented. This figure shows a single position trace that must be attributed to one of two different road paths. At this stage it is still not clear which assignment is true one. So, instead of making a hard decision, soft probabilistic assignments are computed and the final judgement is delayed until more evidence is collected. Elaborating a model capable of propagating this uncertainty is one of the main contributions in this article.

B. Initial estimate

Suppose for a moment that an estimate is provided. Namely, let $\mathbf{x}_m, \mathbf{y}_m$ be a node that approximates the principal road path at a given point. Recall that each individual node comprises both a position and a direction vector, as discussed previously. Hence the pair $(\mathbf{x}_m, \mathbf{y}_m)$ uniquely defines an oriented plane π_m that passes through $\mathbf{x}_m$ and is normal to $\mathbf{y}_m$. Direction vector $\mathbf{y}_m$ defines the positive orientation for this plane.

A set of position data traces is also given. Each trace is composed of a finite sequence of position samples ordered according to time. Time stamps need not be given explicitly but must agree with sample indices. That is, they must be monotonically increasing in such a way that $m < n$ implies that sample m was taken before sample n. Note that the associated time stamps need not be uniformly spaced and may contain gaps due to missing data. In order to ensure a minimum degree of spatial continuity, data traces containing gaps larger than a few seconds must be split.

The sequence of points on each data trace is interpolated by a curve. There are many possible criteria for interpolating a sequence of points in Euclidean space, being nearest neighbor the simplest one. However, by Definition 1 curves are required to be continuous. This is why linear interpolation is chosen instead, since it is the simplest method that yields a continuous mapping. Instant velocity can be estimated via numerical differentiation or Kalman filtering. Section V will demonstrate that even a very crude approximation such as finite differences yields surprisingly good results.

Curves and road samples relate in the following way. Let z_n be the curve that interpolates the n-th position data trace. Let $T_n^{(m)}$ be the set of all parameter values whose corresponding image under z_n intersect π_m with strictly positive orientation,

$$T_n^{(m)} = \left\{ t : (\mathbf{z}_n(t) - \mathbf{x}_m)^T \mathbf{y}_m = 0, \frac{d\mathbf{z}_n}{dt}(t)^T \mathbf{y}_m > 0 \right\} \quad (2)$$

Since each trace contains a finite number of points, and from construction of $\mathbf{z}_n$, $T_n^{(m)}$ must be a finite set. Therefore, the union of all the images under z_n and dz_n/dt

$$X^{(m)} = \bigcup_n \mathbf{z}_n \left(T_n^{(m)} \right) = \{\mathbf{x}_k^{(m)}\},$$
$$Y^{(m)} = \bigcup_n \frac{d\mathbf{z}_n}{dt} \left(T_n \right) = \{\mathbf{y}_k^{(m)}\}.$$

each contain at most finitely many points. An illustration of Notice that both of them comprise column vectors of the same dimension and hence can be regarded as matrices. Therefore, $\mathbf{X}^{(m)}$ and $\mathbf{Y}^{(m)}$ are defined as the rectangular matrices formed by vertically concatenating the transposes of the elements in $X^{(m)}$ and $Y^{(m)}$ respectively.

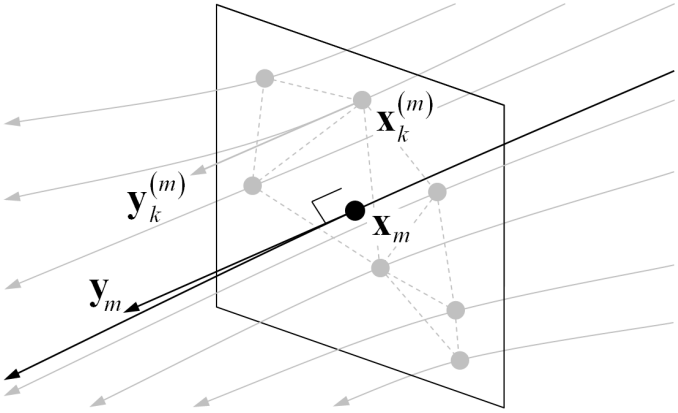

Fig. 4. Cross-section of the PRP. Curves intersect the normal plane at a series of points, depicted as grey dots. Pairwise similarities are drawn between them to derive a weighted graph, represented as grey dashed edges. This graph is then used to cluster the points on the plane and construct a partition.

The set $X^{(m)}$ resembles the cross-section profile of the road at $\mathbf{x}_m$. As already stated earlier, it is not known beforehand which elements in these sets correspond to which road. 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. This idea leads the way to the next step, which consists of performing clustering on π_m. The clustering procedure will yield a partition of $X^{(m)}$ and $Y^{(m)}$ into points that belong to $(\mathbf{x}_m, \mathbf{y}_m)$ and points that do not. Conveniently, pairwise clustering via dominant sets turns out to be ideal for this matter.

C. Clustering on the normal hyperplane

Clustering by searching for dominant sets on a weighted graph is described in Appendix B in [25]. The dominant set framework is particularly effective at finding compact groups of points, and hence it is especially well suited for the purpose of finding principal paths. In order 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. Specifically, let

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

denote the Euclidean distance between points $\mathbf{x}_i^{(m)}$ and $\mathbf{x}_j^{(m)}$ in $X^{(m)}$, and the angle between their corresponding derivative vectors in $Y^{(m)}$, respectively. After thorough experimentation, it was found by trial and error that the expression

$$w(i,j) \propto e^{-d_{ij}^2/\epsilon^2 + \kappa\,(1 - \cos a_{ij})} \tag{3}$$

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

The reader should take a moment to examine Equation 3. Constants ϵ and κ are two parameters that account for uncertainty in the model. The first one corresponds to the standard deviation of a Gaussian distribution and has units of distance, while the second corresponds to the angular concentration parameter of a spherical density function 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.

In practice, an equivalent and more intuitive parameter is used instead of angular concentration. Namely, the standard deviation δ is a more natural way of quantifying the degree of spread. For the Von Mises-Fisher distribution there exists a bijection between concentration and standard deviation, both relating via the following equation

$$\left(\frac{\delta}{2\pi}\right)^2 = 1 - \left(\frac{I_{d/2}(\kappa)}{I_{d/2-1}(\kappa)}\right)^2,$$

where I_j denotes the modified Bessel function of order j and d is the dimensionality of the data. Therefore, given a value for δ, the corresponding concentration can be determined by numerically solving the equation above. Standard deviation is given in radians and varies from 0 to 2π.

The graph, represented as a similarity matrix, is constructed by evaluating Equation 3. Then, a cluster is found by applying the procedure outlined in Appendix B in [25]. This yields a a weight vector $\mathbf{w}^{(m)}$ having as many elements as $X^{(m)}$ and $Y^{(m)}$ and dwelling on the closed standard simplex. Its support $\sigma(\mathbf{w}^{(m)})$ is dominant with respect to the index set of $X^{(m)}$ and $Y^{(m)}$. As explained in the appendix, the k-th element of $\mathbf{w}^{(m)}$ reflects the extent to which the k-th point belongs to the dominant set and hence can be interpreted as the degree or membership of $\mathbf{x}_k^{(m)}$ and $\mathbf{y}_k^{(m)}$ to the current road. Thus $\mathbf{w}^{(m)}$ provides a measure of uncertainty in association.

D. Iterating

The pair $(\mathbf{x}_m, \mathbf{y}_m)$ is an initial estimate of the principal road path at a given position. This estimate is only an approximation to the true path. It constitutes an initial guess that serves as a starting point for iteratively improving it. It is approximate because it does not necessarily satisfy the self-consistency property mentioned in Section II. Ideally, for an infinitely large number of data traces, $\mathbf{x}_m$ and $\mathbf{y}_m$ should be self-consistent with respect to the sets $X^{(m)}$ and $Y^{(m)}$ respectively. Given a finite data set, the best possible strategy is to set

$$\mathbf{x}_m \leftarrow \mathbf{w}^{(m)T}\mathbf{X}^{(m)}, \quad \mathbf{y}_m \leftarrow \mathbf{w}^{(m)T}\mathbf{Y}^{(m)}, \tag{4}$$

that is to assign $\mathbf{x}_m$ and $\mathbf{y}_m$ their corresponding average. Notice how each element in the sum is weighted by its corresponding belief, meaning that the estimate should match the weighted average of all points that project to it.

Each iteration proceeds in a coordinate-wise fashion. It starts with a position $\mathbf{x}_m$ and a direction $\mathbf{y}_m$ and yields two arrays $\mathbf{X}^{(m)}$ and $\mathbf{Y}^{(m)}$ and a vector $\mathbf{w}^{(m)}$. For the next iteration, the estimate is updated according to Equation 4. A

summary of the sampling routine can be seen in the pseudocode in Algorithm 1. This routine takes as input an initial estimate of the principal road path at a given point and iteratively improves it until reaching a tolerance threshold τ. Function SIMILARITY() evaluates the similarity in Equation 3, and the routine CLUSTER() is described in Appendix B in [25]. The final estimate is approximately self-consistent.

Algorithm 1 Sampling routine.

function SAMPLE($\{\mathbf{z}_n\}, \mathbf{x}, \mathbf{y}, \tau$)
  repeat
    $\hat{\mathbf{x}} \leftarrow \mathbf{x}, \quad \hat{\mathbf{y}} \leftarrow \mathbf{y}$
    $T_n \leftarrow \left\{ t : (\mathbf{z}_n(t) - \hat{\mathbf{x}})^T \hat{\mathbf{y}} = 0, \frac{d\mathbf{z}_n}{dt}(t)^T \hat{\mathbf{y}} > 0 \right\}$
    $X \leftarrow \bigcup_n \mathbf{z}_n(T_n), \quad Y \leftarrow \bigcup_n \frac{d\mathbf{z}_n}{dt}(T_n)$
    $W_{ij} \leftarrow$ SIMILARITY$(\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}_j, \mathbf{y}_j), \quad i \neq j$
    $\mathbf{w} \leftarrow$ CLUSTER$(\mathbf{W})$
    $\mathbf{x} \leftarrow \mathbf{w}^T \mathbf{X}, \quad \mathbf{y} \leftarrow \mathbf{w}^T \mathbf{Y}$
  until SIMILARITY$(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \mathbf{x}, \mathbf{y}) < \frac{\tau}{2}$
end function

Deriving a convergence bound for this algorithm is nontrivial, the greatest difficulty being the fact that the data set is not finite[2]. Moreover, in some cases it may not converge. This happens for certain geometric configurations of the data traces that generate empty $X^{(m)}$ and $Y^{(m)}$ 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.

E. Placing the initial seeds

So far, this section has focused on a single node of the principal road path. It has shown how an initial estimate can be refined to yield a self-consistent node. However, the problem of how to efficiently place these initial estimates, or seeds, still remains. In order to capture the full extent of the road network, a number of seeds must be deployed along the traces. Ideally, they should be evenly spaced in such a way that they cover the entire area of interest. However, too large a node set could pose a serious computational burden. This is why care must be taken in order to limit their number.

A simple yet effective seeding method is as follows. It proceeds by taking each data trace in turn and placing seeds along its corresponding trajectory. At the same time, a nearest-neighbor tree of the node set is built incrementally. The tree calculates and stores similarities between nodes the same way as in Equation 3. Construction of the tree proceeds in an incremental fashion. At each step, a new candidate seed is drawn and, depending on its similarity to the rest of the seeds, it is either discarded or incorporated into the tree. Let m index the set of seeds already present in the tree. A new candidate seed $\mathbf{x}_n$, $\mathbf{y}_n$ is added as a leaf node if and only if

$$\max_m w(m, n) < \alpha, \tag{5}$$

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

where $0 < \alpha < 1$. This decision rule ensures that the tree is grown only if the new node lies far enough from all the other nodes that are already present. It effectively imposes a minimum distance between them, thus limiting their number and making the algorithm scalable to large data sets.

A method for selecting nodes would be to simply place them at regular intervals along each trajectory. However, this would create a compromise between resolution and computational load. If the interval is too large then consecutive seeds would contain large gaps in between, resulting in an undersampled road network. On the other hand, a very small interval would create a large amount of candidate nodes and hence a large number of nearest neighbor queries. Both extremes are undesirable and should be avoided.

To bypass this compromise, a very simple strategy can be employed. It involves little overhead and minimizes the gaps between candidates, yielding a dense group of seeds. Given a trajectory, the first step consists of drawing evenly spaced nodes at a fixed distance $\alpha\epsilon$ from each other. This yields a set of candidate seeds, indexed by n. Let k be the index of the first candidate that exceeds the decision threshold in Equation 5,

$$k = \min \left\{ n : \max_m w(m, n) < \alpha \right\}.$$

If k equals either the first or last index, then the corresponding seed is immediately added to the tree. In contrast, for other values of k a one-dimensional search is conducted to find a candidate that lies exactly at a similarity value of α from the tree. Because the trajectory is a curve and the similarity metric is a continuous function, such candidate must lie between seed k and its predecessor. Therefore, it can be found approximately using binary search or a standard line search routine.

Once all the seeds are in place, they are used as initial estimates of the principal road path. The sampling routine in Algorithm 1 is called for each individual seed in turn. Upon completion, a set of self-consistent road samples and a corresponding set of weight vectors is obtained. The set of weight vectors, holding values that quantify the uncertainty in the association of roads to traces, are of central importance in the second stage of the algorithm. This stage deals with how to link nodes together to generate the network topology and is thoroughly described in the following section.

IV. LINKING

The last section described the first stage of the mapping algorithm. During this stage, the road network was sampled by covering it with a number of seeds lying on the principal road paths. The seeds will now serve as anchor points upon which the rest of the map will be built. The next step consists of linking nodes together so as to capture the topology of the network. The way connections are drawn is explained in this section. Basically, linking is formulated as a constrained optimization problem. Because connections are often ambiguous, not all links are actually needed and some must be discarded. In order to decide which ones to discard, the links are first ranked using a scoring criterion and then pruned incrementally according to their score. Details are given next.

A. Transition count matrix

Linking is performed by connecting nodes to one another via directed edges. The nodes and edges form a directed graph that comprises the skeleton of the road map. The decision of whether two given nodes 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 nodes are weakly coupled and should not be connected. Each pair of nodes has an edge and hence a corresponding likelihood value associated to it. Therefore, the set of all these values can be grouped together to form the transition matrix.

Recall the intersection set defined in Equation 2. This set is composed of all parameter values in the n-th trajectory that map to points on the m-th hyperplane with strictly positive orientation. Consequently, the set $\bigcup_m T_n^{(m)}$ comprises all the intersections of trajectory n with every hyperplane. All elements in this set also possess a corresponding entry in one of the weight vectors $\mathbf{w}^{(m)}$ associated to the road nodes. Therefore, each element implicitly maps to the index of the node that was intersected, the parameter value for which the intersection occurred and the weight. Let

$$\mu = \{\mu_n^{(k)}\}, \quad \tau = \{\tau_n^{(k)}\}, \quad \omega = \{\omega_n^{(k)}\},$$

be the set of node indices, parameter values and weights respectively for all of the elements in the union. For convenience, all three sets can be represented in a compact form by grouping them together in vectors. Specifically, define the set $\{\mathbf{w}_n^{(k)}\}$ using an m-ary coding scheme, where the m-th element of vector $\mathbf{w}_n^{(k)}$ is equal to 1 if $\omega_n^{(k)} > 0$ and $m = \mu_n^{(k)}$, and 0 otherwise. Assume the sets are sorted according to time, that is $i \leq j \Rightarrow \tau_n^{(i)} \leq \tau_n^{(j)}$, and define the weight matrix as the sum of outer products of pairs of consecutive vectors

$$\mathbf{W} = \sum_{n,k} \mathbf{w}_n^{(k-1)} \mathbf{w}_n^{(k)T}. \tag{6}$$

With these definitions, the transition matrix $\mathbf{A}$ can be expressed as the row-normalized version of the weight matrix,

$$\mathbf{A} = \operatorname{diag}(\mathbf{We})^{-1} \mathbf{W}, \tag{7}$$

where $\mathbf{e}$ is the column vector with unit elements. In this equation it is assumed that every row of $\mathbf{W}$ contains at least one nonzero element, so that the inverse is well defined.

The transition matrix in Equation 7 is a normalized version of the matrix of weighted transition counts between road nodes. Its expression is very similar to the maximum likelihood estimate of the transition matrix in a hidden Markov model (HMM). In this case, A_{ij} is proportional to the sum of the posterior joint probabilities of any two consecutive hidden states being i and j. An analogy can be drawn between the HMM and the model presented here, since the identity of the road samples act as latent variables. The difference is that for the road model there is no ambiguity about hidden states because a trajectory almost never intersects two or more planes simultaneously. Or in other words, it is extremely rare to find two elements in the set τ_n that are identical. Therefore, the joint posterior probability is always equal to one. This is reflected in Equation 6, where the weight matrix is formed by accumulating products of binary vectors.

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.

An edge is said to be redundant if its removal does not affect the local edge connectivity of the graph. That is, an edge is redundant if, after removing it from the graph, every pair of vertices remains connected. A simple illustration of edge redundancy that arises frequently in practice is given in Figure 5. This example shows a typical situation involving a set of edges, one of which is spurious and should be removed. In this case it is clear which of the edges is to be pruned because the choice is intuitive. However, it is not clear how this intuition can be extended to more complex scenarios. Arriving at a general criterion that applies to every possible setting is the focus of the following paragraphs.

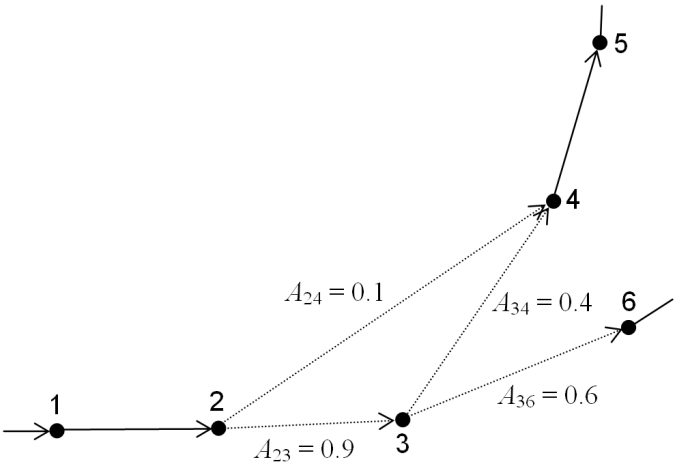

Fig. 5. Edge redundancy arising from the transition matrix. Numbered black dots represent road samples in the vicinity of a junction, where a road splits into two. Arrows denote edges present in $\sigma(\mathbf{A})$. Solid edges have likelihood A_{ij} equal to unity and hence are bound to belong to the graph. Dotted edges, on the other hand, have likelihood strictly smaller than one. In this case, edges $(2,4)$ and $(3,4)$ are both candidates for removal because removing either one of them does not destroy the local edge connectivity. However, edge $(2,4)$ has the smallest likelihood of the two and hence is the redundant one.

Let $\{G_E\}$ be a family of graphs parameterized by their edge set E. Each member is a directed weighted graph $G_E = (V, E, w)$. All members share a common vertex set $V = \{1, \ldots, v\}$, where v is the size of the transition matrix $\mathbf{A}$ in Equation 7, and a weight function $w : E \to [0,1]$, defined as $w : (i,j) \mapsto A_{ij}$. Also, every edge set is constrained to range over the support of $\mathbf{A}$, that is

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

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 can be cast a maximization problem. Specifically, the problem is formulated as that of finding the graph with the largest overall weight,

$$\max_{E \subseteq \sigma(\mathbf{A})} \prod_{(i,j) \in E} A_{ij}, \tag{8}$$

amongst all possible subgraphs of $G_{\sigma(\mathbf{A})}$. As it is, this minimization is ill-posed. Because all elements of $\mathbf{A}$ lie on the unit interval, the objective function is monotonically decreasing in $|E|$, the cardinality of the edge set. Therefore, the total weight can be trivially maximized by setting $E = \emptyset$. To avoid this vacuous solution, a set of constraints is added. These constraints will ensure that the optimum graph does not become disconnected as edges are pruned.

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. Formally, this constraint can be expressed in terms of the transitive closure of G_E,

$$\lambda_E(i,j) \geq \lambda_{\sigma(\mathbf{A})}(i,j) \quad \forall\, i,j \in V, \tag{9}$$

where λ_E is the local edge connectivity defined in Appendix A in [25]. The constraint set ensures that the optimum graph G_E contains no redundant edges. This can be verified via the following reasoning. Every edge $e = (i,j) \in E$ in the optimum graph is also an edge-independent path between vertices i and j. In fact, Equation 9 implies that it must be the path with largest weight connecting them. Should there exist more than one edge-independent path, they must all have weight lower than A_{ij}. Otherwise, the objective function in Equation 8 could be increased by removing e from E. For this reason, e cannot be redundant.

The complete minimization problem consists of solving Equation 8 with respect to E, subject to Equation 9. 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 complexity becomes linear in the number of vertices and quadratic in the number of edges. Although complexity is still quadratic in the worst case, the routine scales well in practice since graphs tend to be very sparse. Furthermore, computation time can be minimized by taking into account certain properties of the edge set. A full description of the routine can be found next.

C. Suboptimal linkage

Now that the linking problem is well defined, an efficient method for solving it can be derived. Recall 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 Equation 9 and hence yields a feasible graph. Therefore, removing it increases the overall weight in Equation 8, bringing the graph closer to the optimum.

Pseudo-code for the linking routine can be found in Algorithm 2. The 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 is modified. If it decreases, the edge is incorporated back again into the graph. Testing whether the weight of the shortest path between two vertices is larger than a given threshold can be achieved via breadth-first search. The time complexity of the search is bounded by $|E|+|V|\log|V|$, since in the worst case all vertices and edges must be traversed. However, the complexity is always much smaller in practice since the threshold is often exceeded very early during the search. This makes identification of redundant edges fairly inexpensive, allowing the test to be performed intensively without incurring a large computational cost. Also, notice that all edges corresponding to vertices with out-degree[3] equal to one are never redundant and hence need not be tested.

Algorithm 2 Linking routine.

```
1:  function LINK(A)
2:      E ← σ(A)
3:      E ← SORT(E)                    ▷ Sort in descending order.
4:      for (i,j) ∈ E do
5:          E ← E − {(i,j)}
6:          if λ_E(i,j) < λ_σ(A) then  ▷ Test connectivity.
7:              E ← E ∪ {(i,j)}
8:          end if
9:      end for
10: end function
```

Last of all, observe from Equation 8 that the objective function may have multiple global maxima. In practice this may occur due to a large number of edges having unit weight. In these cases, the edge set with the highest cardinality is selected. Doing so substantially reduces the size of the problem, since only those edges with $A_{ij} < 1$ are considered for removal. Hence a considerable amount of computation is saved if all edges with unit weight are fixed from the start. An edge (i,j) has unit weight if and only if $A_{ij} = 1$, meaning that every trace starting at node i transitions to node j. Therefore, unit-weight edges cannot be redundant.

A skeleton of the map is obtained once the linking routine has converged. This skeleton consists of a set of road sample nodes linked together by the directed edge set. Already this representation holds valuable information about network topology and geodesic distance between nodes. It also contains transition counts between nodes and measures of uncertainty that quantify the association of traces to roads. There is still a large number of additional attributes that can be computed to augment the map. Due to lack of space, it is not possible to engage in a thorough discussion here. A few of these features will be briefly mentioned in the next section, leaving an in-depth discussion for future work.

[3]The out-degree of a vertex is the number of edges stemming from it.

V. 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.

A. Experimental data

Over 10 Mb of raw double-precision data was used to benchmark the algorithm. Position information was collected at an opencast mine in Western Australia with standard GPS equipment built around a SiRF III chipset. The data corresponds to 5 days of operation of more than 15 mobile resources. Since all vehicles are retrofitted with low-cost GPS sensors, the algorithm presented in this work becomes very appealing for dynamic road mapping. Around 400 position traces were constructed with almost no preliminary processing. The only two considerations that were taken are as follows. First, a simple outlier rejection method was implemented based on a distance threshold. Specifically, consecutive position samples that lay more than 100 mts apart were considered as outliers and removed from the data set. And second, the size of time gaps between samples was examined. Sequences containing gaps more than 3 sec long were split into several segments so as to ensure a minimum degree of spatial coherence.

Figure 6 shows the final road map overlaid on an aerial photograph of the mine. The map spans a total of 3.5 by 10.5 km. Only two critical parameters, ϵ and δ, are factors that strongly affect the outcome of the algorithm. The standard deviation for position was set equal to 15 mts, roughly comparable to the width of the haul trucks driving on these roads. The circular dispersion parameter was selected as $1/8$ (corresponding to a standard deviation of $\pi/4$ rad) after a small number of tuning instances. The algorithm remained fairly robust to variations in δ, yielding almost identical road maps when doubling its original value. The seeding threshold α in Equation 5 was fixed at $1/2$.

The algorithm was implemented in MatLab®, release 2008 beta, using class definitions. It was tested on an Intel® Core™ Duo CPU with a 2.33 GHz processor and 2 Gb of RAM. Execution took 83 min, with the sampling routine accounting for more than 99% of the total running time. Over 4100 seeds were placed during the sampling stage, giving an average of roughly 1.2 sec per sample. This amount is not excessively large given the dimensions of the map and taking into account that no specialized storage structure was used for implementing the sampling algorithm (see Algorithm 1).

B. Inferred road map

Figure 7 shows a typical road junction extracted from the data set. Data in this area is relatively noise-free due to satellite availability and because trucks tend to travel at high speeds[4]. However, position traces exhibit a considerable amount of dispersion along the cross section. Notice how drivers traveling from west to south tend to take either very sharp or very wide turns. Also, vehicles driving westbound on the left side often cross to the opposite lane. A similar scenario is depicted in Figure 8, which shows what is possibly the most difficult intersection area found in the data set. Here, the high density of traces cluttering the center of the region makes mapping a very challenging task. Even so, the algorithm is able to reconstruct the road network. In both cases, the underlying topology is captured and the principal road paths are correctly inferred.

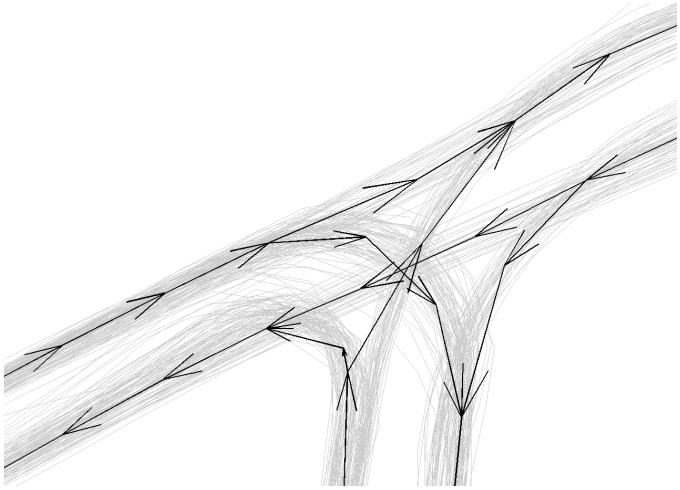

Fig. 7. Typical road junction inferred from the data set. The road map is represented as a directed graph. It is depicted in the figure as a set of black arrows, each one connecting two road sample nodes together. Notice that, even though trucks take sharp or wide turns and drive on the wrong lane, the algorithm still manages to faithfully recover the structure of the junction.

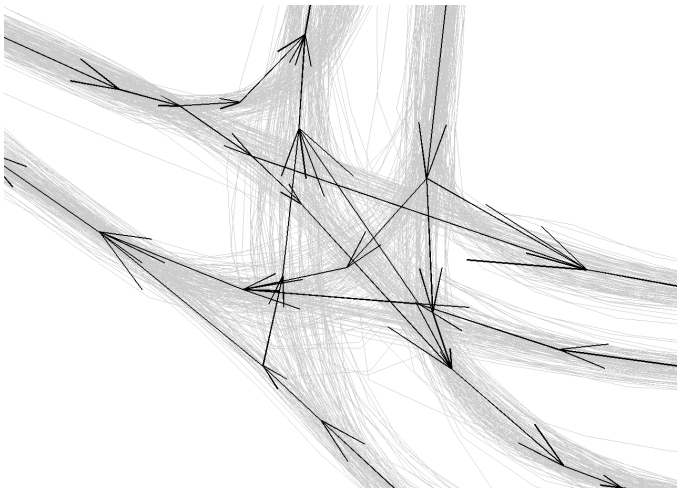

Fig. 8. Complex intersection area from the data set. This part of the network is particularly difficult to infer because of the large amount of clutter in the middle. In spite of this, the algorithm performs well and is able to extract an accurate road map.

A rather different situation is presented in Figure 9. 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 turn aside from a straight course. Vehicles traveling northwest often needed to swerve to the right lane in order to avoid a collision. Once again, the road map was still correctly inferred from this data. The road found by 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.

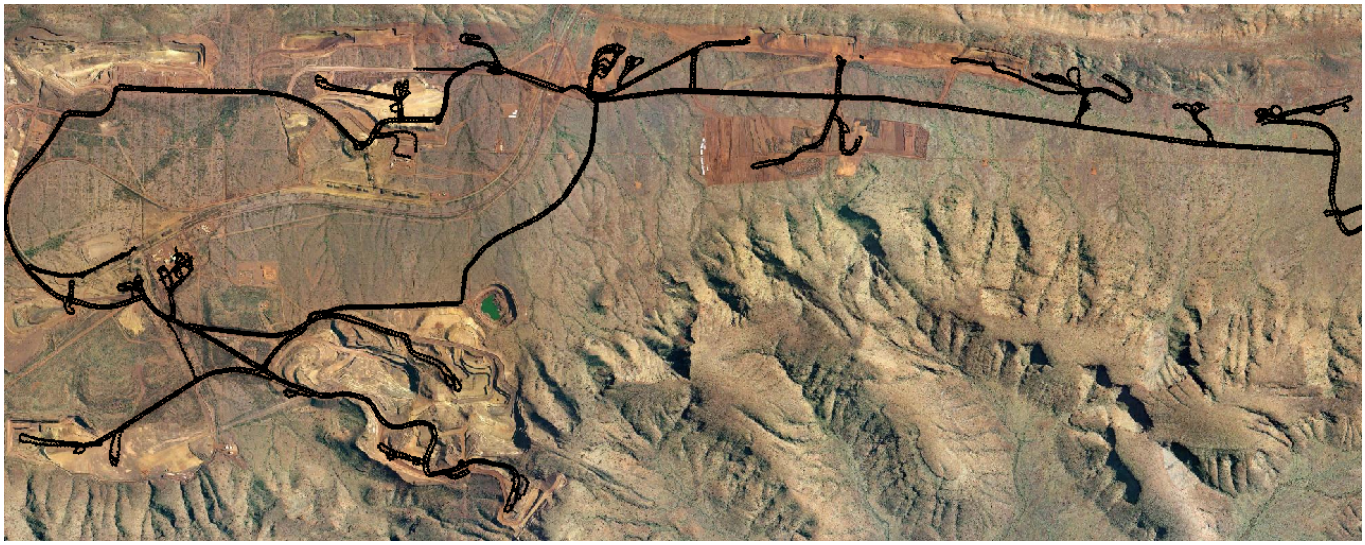

Fig. 6. Final road map superimposed on an aerial photograph of the mine. The map is drawn in black. Although details are not appreciable at this scale, it still serves to contemplate the magnitude of the problem. The photograph spans an area of 10.5 by 3.5 kilometers.

algorithm follows the most intuitive path and does not appear to be biased towards the middle of the road. This robustness stems from the update equation in the road sampling scheme. It is due to the fact that Equation 4 weights positions and directions according to their corresponding responsibilities.

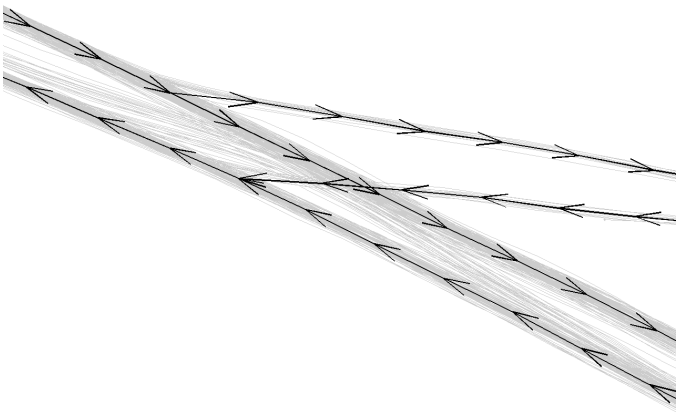

Fig. 9. Heavily traveled junction with an obstruction. A two-way road splits into two at a junction, one of the two emerging roads being much more transited than the other. Data traces show that an obstruction was present at the intersection, temporarily blocking the heavily transited road. This forced truck drivers to swerve in order to avoid the obstacle.

The approach remains robust to noise and outliers even when data does not abound. The crossing shown in Figure 10 is not visited as frequently as the junctions in the previous figures. Hence only a small amount of data is available for extracting roads. Also, some of this data contains a relatively high level of noise. When trucks approach the crossing from the north-east, they reduce their speed almost to a complete halt. Because standard GPS tracking filters are less accurate at low speeds, this data is often noisier than normal. However, in spite of the sparsity and the noise, results are still correct. The algorithm still manages to robustly trace the principal road paths and capture the underlying structure of the network.

Due to lack of space, a comparison with other approaches

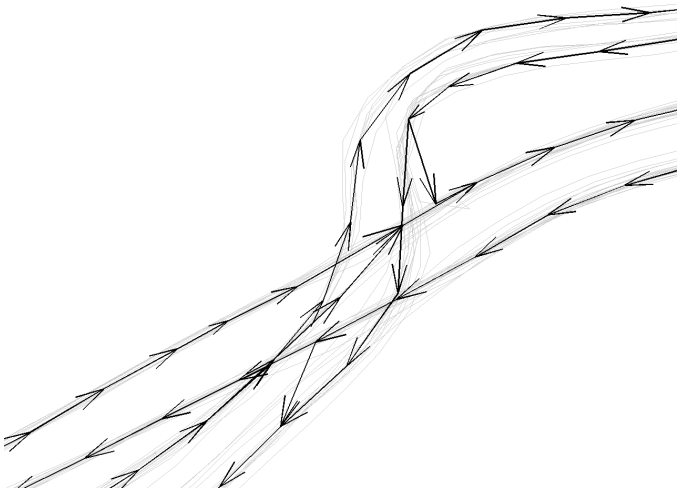

Fig. 10. Road crossing with sparse data. Two double-lane roads intersect each other, creating a narrow crossing. Because this area of the mine is not visited as frequently as the previous two, the amount of data is much smaller. Vehicles approaching from the north-eastern corner tend to slow down, almost to a full stop. Hence, due to the low speeds, these data traces are often noisier than usual.

is not presented here. Please refer to the extended technical report [25] for detailed comparisons with the work of [7] and [5] and a quantitative numerical analysis of the results.

VI. Conclusion and Future Work

This article described an approach for automatically inferring high-precision road paths from position data. The approach draws on machine learning concepts and graph theory to define a consistent model of the road network. It proceeds in two stages, first by sampling the PRPs at a series of nodes and then by linking these nodes together to form a directed graph. Sampling is performed in an iterative fashion by repeatedly improving an initial estimate. Node linkage is expressed as a constrained optimization problem and is approximately solved via a greedy search.

The algorithm was tested with data from an operating mine. Experimental results clearly show that the algorithm achieves the desired performance and in all cases determines the center of the path followed by the resources. These are remarkable results considering the complexity of the task. The inferred roads agree with the data, showing that the PRP are indeed the standard path of the vehicles. The algorithm is also successful at capturing the shape and topology of the network. In addition, it compares favorably to other existing methods in the literature. The algorithm exhibits superior robustness when tracing the principal road paths and produces a much more accurate graph representation of the network especially at the intersections, where it matters most for automation and safety applications.

The mapping algorithm is scalable to large data sets. Experiments have shown that the algorithm can obtain high-quality maps for a medium-sized mine in an hour and a half without any computer optimization. Furthermore, it is not limited by the size of the data. So far, no experiments were conducted with data spanning more than five days. However, this is not due to memory constraints. The reason is that roads tend to change relatively frequently, and these 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 the efficiency of the algorithm and its linking performance, as well as studying the topic of place recognition. With respect to place recognition, previous approaches have used complementary information from a separate routine that is able to recognize these areas. It will be very useful to tailor the algorithm so as to account for these kinds of areas. Some work has already been undertaken towards place recognition from position information [26], [27], [28]. These are valuable tools that could be used to complement the mapping approach. For example, the output from the place-detection routine could be used to selectively prune the road map. And also, image processing tools could prove useful when aerial photographs or satellite imagery is available. All these possibilities pose promising directions for future research.

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] A. Eidehall, J. Pohl, F. Gustafsson, and J. Ekmark, "Toward autonomous collision avoidance by steering," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 84–94, March 2007.

[3] R. Toledo-Moreo and M. Zamora-Izquierdo, "IMM-based lane-change prediction in highways with low-cost GPS/INS," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 180–185, March 2009.

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

[5] 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.

[6] 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.

[7] 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.

[8] I. Sen and D. Matolak, "Vehicle-to-vehicle channel models for the 5 GHz band," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 2, pp. 235–245, June 2008.

[9] S. Worrall and E. Nebot, "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.

[10] S. Worrall, "Providing situation awareness in complex multi-vehicle operations," Ph.D. dissertation, School of Aerospace, Mechanic and Mechatronic Engineering, July 2009.

[11] M. Negri, P. Gamba, G. Lisini, and F. Tupin, "Junction-aware extraction and regularization of urban road networks in high-resolution sar images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 10, pp. 2962–2971, October 2006.

[12] P. Gamba, F. Dell'Acqua, and G. Lisini, "Improving urban road extraction in high-resolution images exploiting directional filtering, perceptual grouping, and simple topological concepts," *IEEE Geoscience and Remote Sensing Letters*, vol. 3, no. 3, pp. 387–391, July 2006.

[13] J. Hu, A. Razdan, J. Femiani, M. Cui, and P. Wonka, "Road network extraction and intersection detection from aerial images by tracking road footprints," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 4144–4157, December 2007.

[14] 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.

[15] 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.

[16] S. Rogers, P. Langley, and C. Wilson, "Mining GPS data to augment road models," in *Proceedings of the Fifth ACM International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 1999, pp. 104–113.

[17] M. Drozdzynski, S. Edelkamp, A. Gaubatz, S. Jabbar, and M. Liebe, "On constructing a base map for collaborative map generation and its application in urban mobility planning," in *Proceedings of the 2007 IEEE International Conference on Intelligent Transportation Systems*, 30 2007-Oct. 3 2007, pp. 678–683.

[18] S. Edelkamp, S. Jabbar, and T. Willhalm, "Geometric travel planning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 5–16, March 2005.

[19] J. Roth, "Extracting line string features from gps logs," in *5. GI/ITG KuVS Fachgespräch "Ortsbezogene Anwendungen und Dienste"*. University of Applied Sciences, Nürnberg: Scientific Series Offprint, 2008, pp. 41–46.

[20] T. Hastie and W. Stuetzle, "Principal curves," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 502–516, June 1989.

[21] H. Alt and M. Godau, "Computing the fréchet distance between two polygonal curves," *International Journal of Computational Geometry and Applications*, vol. 5, no. 1–2, pp. 75–91, 1995.

[22] S. Atev, G. Miller, and N. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE Transactions on Intelligent Transportation Systems*, 2010.

[23] B. Kégl, A. Krzyżak, T. Linder, and K. Zeger, "A polygonal line algorithm for constructing principal curves," in *Proceedings of the 1998 International Conference on Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1999, pp. 501–507.

[24] J. Zhang, D. Chen, and U. Kruger, "Adaptive constrained k-segment principal curves for intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 4, pp. 666–677, 2008.

[25] G. Agamennoni, J. Nieto, and E. Nebot, "Inference of principal road paths using GPS data," http://www-personal.acfr.usyd.edu.au/jnieto/Research_files/PRPInferenceReport2010.pdf, June 2010.

[26] 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.

[27] 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.

[28] 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.