

# Mining GPS Data to Augment Road Models

**Seth Rogers**

**Pat Langley**

**Christopher Wilson**

(rogers, langley, wilson)@rtna.daimlerchrysler.com

DaimlerChrysler Research and Technology Center

1510 Page Mill Road

Palo Alto, CA 94304-1135

650-845-2533

## **Abstract**

Many advanced safety and navigation applications in vehicles require accurate, detailed digital maps, but manual lane measurements are expensive and time-consuming, making automated techniques desirable. This paper describes a data-mining approach to map refinement, using position traces that come from Global Positioning System receivers with differential corrections. The computed lane models enable safety applications, such as lanekeeping, and convenience applications, such as lane-changing advice. Experiments show that, starting from a baseline map that is commercially available, our lane models predict a vehicle's lane with high accuracy from a small number of passes over a particular road segment. Multiple position traces are a powerful new source of data that enables cheap, automated methods of inducing lane models, as well as other geographic knowledge, like traffic signals and elevations, and potentially impacts any geographic information system with a need to relate to actual behavior.

Keywords: Background knowledge, noisy data, incremental algorithms, implementation and use of KDD systems, case studies, evaluating knowledge and potential discoveries.

# 1 Introduction

There are many potential uses for an in-car system that can determine position on the road relative to lane markings. These can be roughly grouped into safety applications, such as lane departure warnings (Pomerleau, 1995), and convenience applications, such as lane-changing advice to improve traffic flow (Moriarty & Langley, 1998). Although Differential Global Positioning System (DGPS) receivers are often *accurate* enough to locate a vehicle to within a lane,<sup>1</sup> there is no reliable source of lane *locations*.

In this paper, we present and evaluate the first system, to our knowledge, that enhances digital road maps with descriptions of lane structure, including number of lanes and their locations. Our approach involves mining massive amounts of DGPS traces from floating probe vehicles to augment the digital maps with lane information, creating a resource usable by any lane-related automotive application. Our system does not require special vehicles or expensive hardware to collect data, unlike some GPS mapping methods (Grejner-Brzezinska, 1995). Our approach to data collection is to unobtrusively and indiscriminately gather as much data as possible from multiple drivers going about their ordinary business and to mine the resultant traces for knowledge about the road network.

Previous work on lane boundary finding has focused on directly performing tasks, such as lanekeeping, using machine vision to find lane markings related to the vehicle position (Chen, Jochem, & Pomerleau, 1995). This approach is limited in several ways. First, the vision system must be correctly calibrated to the lane markings it will sense. Our reliance on DGPS traces effectively lets us use the driver's lanekeeping ability to identify the center of the lane. The absolute nature of the data also provides information on upcoming terrain not directly sensible from the vehicle. Second, it is difficult, if not impossible, to build an accurate database of lane models with machine vision, or any other relative sensing method, alone. This is because the straightforward approach to building such a database is to store the lane structures in a spatially absolute reference system, and vehicles without an absolute sensing method, such as GPS, have no way to register the data spatially. An advantage to using machine vision techniques is that GPS accuracy suffers

---

<sup>1</sup>The GPS receivers used in this study are generally accurate to between 1 and 2 meters, whereas road lanes are about 3 to 4 meters wide.

when the satellite view is partially or totally obstructed. In fact, deployed systems will probably use a combination of both technologies. We are currently developing a positioning system that combines GPS and local sensors to compensate for satellite visibility problems (Weisenburger & Wilson, 1999).

The next section motivates this work in more detail by describing some uses of a lane-sensitive vehicle. We then discuss some related work on problems similar in spirit to our own and some possible approaches to the problem. After this, we describe a system that creates an accurate description of road centerlines from a commercially-available map with relatively low accuracy and induces lane models by unsupervised learning. We evaluate the lane models against manual lane labels on highways. Finally, we describe some plans for future extensions to the work.

## 2 Lane-sensitive applications

The combination of a digital road map with accurate lane models and an in-car positioning system enables several novel applications (Wilson, Rogers, & Weisenburger, 1998). Some require additional parameters and data fields, but all are easily derived from current position traces. These include:

**Lane departure warning/lanekeeping** This safety application tracks a car's current offset from the road/lane centerline. If it deviates more than a certain amount, a warning signal could activate or the car could assume control to avoid an accident. The threshold amount could be related to the standard deviation of offsets during typical driving. This application requires very high position accuracy to avoid false positives and negatives.

**Lane-level navigation** This enhancement to standard road-level navigation advises the driver as to which specific lane he should choose to reach his destination without excess and last-minute lane changing. Besides the lane models, this application requires a per-lane model of intersection behavior. For example, position traces may indicate that 100% of drivers in the left lane at a particular intersection turn left, 50% of drivers in the right lane go straight, and 50% turn right.

**Dynamic lane closures** If aggregate data on lane occupancy is available dynamically through wireless communications, a lane closure application can compare current occupancies for a road segment with past lane occupancies. If that lane is particularly under-represented, the application may infer that it is closed due to an accident or construction. Navigation applications can then take this into account when calculating routes.

We believe these, and other, safety and convenience applications will provide benefits to drivers, and that unsupervised learning from position traces will make them possible without the high cost typically required to build the supporting database by hand.

### **3 Related Work**

At an abstract level, this paper addresses the problem of taking an existing knowledge structure, the digital map, and augmenting it with additional information, the lane models. If we view the digital map as a theory describing the actual roadways, then adding lane models refines the theory, making it more accurate and complete. Our approach combines the strengths of theory revision and automated mapping research to take advantage of existing knowledge while processing large amounts of unlabelled real-world data.

#### **3.1 Theory revision**

Much work in theory revision is framed as a companion to explanation-based learning. Since the latter typically requires a complete and correct theory, theory revision techniques rework an invalid theory into a form such algorithms accept. If an explanation module fails to generate an explanation of some examples, the theory revision module could inductively guess at a refinement or correction, allowing valid explanations and use of the theory on similar examples in the future.

There are several research projects in the literature that apply this framework of “learning by failing to explain” to particular theory representations and tasks. Ourston and Mooney’s (1994) EITHER uses theory revision on supervised learning problems. The system accepts a theory expressed in propositional Horn-clause notation and a number of labelled training examples. The theory evaluates the features and predicts a label. If the label is wrong, EITHER computes the

minimal change to the theory that corrects the prediction, then continues making changes to the theory until all examples are correctly classified. Our problem differs in that it involves unsupervised learning, so the training procedure cannot estimate its own performance.

A common knowledge representation for the diagnosis of complex machines is the fault hierarchy, which lets technicians proceed from a high-level description of symptoms to the identification of likely causes and malfunctions through a series of tests. Langley, Drastal, Rao, and Greiner (1994) describe the  $\Delta$  theory revision algorithm for correcting the fault hierarchy in case of diagnostic errors. Like EITHER, their system detects training cases that are mislabelled by the existing knowledge base. The revision procedure generates all possible transformations of the fault hierarchy and chooses the one that reaches the most correct diagnoses, continuing until there is no transformation that improves on the current fault hierarchy. Besides depending on labelled training examples,  $\Delta$  has little relevance to our problem because it exhaustively generates theory transformations, which is not practical in continuous domains.

Some planning-based systems that interact with an environment also demonstrate the use of theory revision to complete their tasks successfully. If Gil's (1992) EXPO, or Pearson's (1996) IMPROV fail to achieve a goal expected from planning, they attempt to correct their plan knowledge through interaction with the environment. Both agents take a variety of actions in the world, then analyze the effects to determine how to perform better in the future. Our problem is fundamentally different because our system cannot to perform experiments to test hypotheses. Instead, it is forced to passively observe the environment and build knowledge structures. Although Wang's (1996) OBSERVER also passively observes a series of expert execution traces, it also requires sensors to record the effects of the expert's actions on goal conditions. A final distinction from all these planning systems is that, rather than accomplish any specific goal, our system attempts to augment current knowledge about the driving environment.

### **3.2 Automated mapping**

Researchers have reported some progress in automatically building maps from rich sensor data, but they have paid little attention to taking advantage of existing knowledge structures. Teller (1998) reports on Argus, a system that also infers knowledge from unlabelled data, but that does not have

the advantage of preexisting background knowledge. Argus constructs a 3D model of a scene from a series of digital images taken by a mobile camera platform. As in our effort, he acknowledges the need for an absolute reference system to build the database, and, like our system, Argus uses GPS traces. The system employs a GPS receiver on the camera platform to establish the absolute coordinate system. In this case, however, the positions themselves only provide a reference point for processing, and the principal algorithms operate on the images.

Automated mapping approaches have also focused on special-purpose, labor-intensive efforts to exhaustively map a target area. For example, the GPSVan (Grejner-Brzezinska, 1995) combines many sensors, including multiple GPS receivers, laser cameras, and stereo vision, to capture detailed information about the roadways it travels. However, the system is prohibitively expensive and requires dedicated personnel to encode features as the vehicle drives. The fields of machine learning and data mining have examined techniques to extract useful knowledge from large data sets not specifically designed to support modeling a particular phenomenon, making up in volume what is lacking in focus and detail. This approach has the potential to reduce mapping costs, covering a target area roughly at first, then with higher precision as more data become available.

## **4 Problem analysis**

The lane-sensitive applications described in Section 2 require a comprehensive, detailed database of roads for the targeted area. Since there are hundreds of thousands of road miles, it is prohibitively expensive and logistically challenging for cartographers to measure the entire road network. Our approach is to track probe vehicles as they sample the road network and invoke unsupervised learning techniques to induce the lane structure without error-correcting feedback.

### **4.1 Observable data characteristics**

To sample the road network, we equip a fleet of cars with absolute position sensors, and they record traces of their trips. Each record in the trace includes the latitude and longitude of the vehicle, as well as the estimated standard deviations on the latitude and longitude. The probe cars record positions at regular intervals. These probe cars require two main components: accurate position sensing, usually built around a GPS receiver, and communications with a centralized aggregator.

The cost of GPS devices is rapidly decreasing, to the point where most new cars sold will have at least one GPS receiver in the next few years. Wireless technology is also advancing rapidly, and position communications may be “piggybacked” on other content, such as route update requests. In the near future, cars with these capabilities will become commonplace, making it possible to build a database of raw position traces with little cost.

Figure 1 plots two sample position traces in the San Francisco Bay Area. Some parts of one trace coincide with the other trace while other parts are solitary. The plot overlays the traces on a rough digital map available from Navigation Technologies, Inc. These maps divide the road network into portions of road between two intersections, called *segments*. For example, at a standard highway interchange, the segments are the part of the highway before the exit, the part between the exit and the entrance, and the part after the entrance. Each segment has a unique identifier and associated attributes, including the segments to which each end connects and a rough approximation of its shape.

The problem with such a database is that it provides no direct data regarding the information of interest: the lane a car occupies at a given time. Another problem is that the positioning systems will not be perfectly accurate. Generating lane models from such data requires the use of background knowledge about the domain to structure the input and statistical techniques to accommodate the noise.

## **4.2 Possible approaches to finding lane models**

We implemented a map-matching module that takes position traces and a digital map, then finds the most likely sequence of segments taken by the vehicle, along with the points in the trace where the vehicle transitioned from one segment to another. The module uses a modified shortest path algorithm to find a minimum-error path from the nearest starting intersection to the nearest ending intersection. The error at each intersection in the path is the closest distance between the intersection and the position trace. Since the intersection locations and the position traces are inaccurate, the map matcher will not necessarily give correct results, but in practice it met our needs.

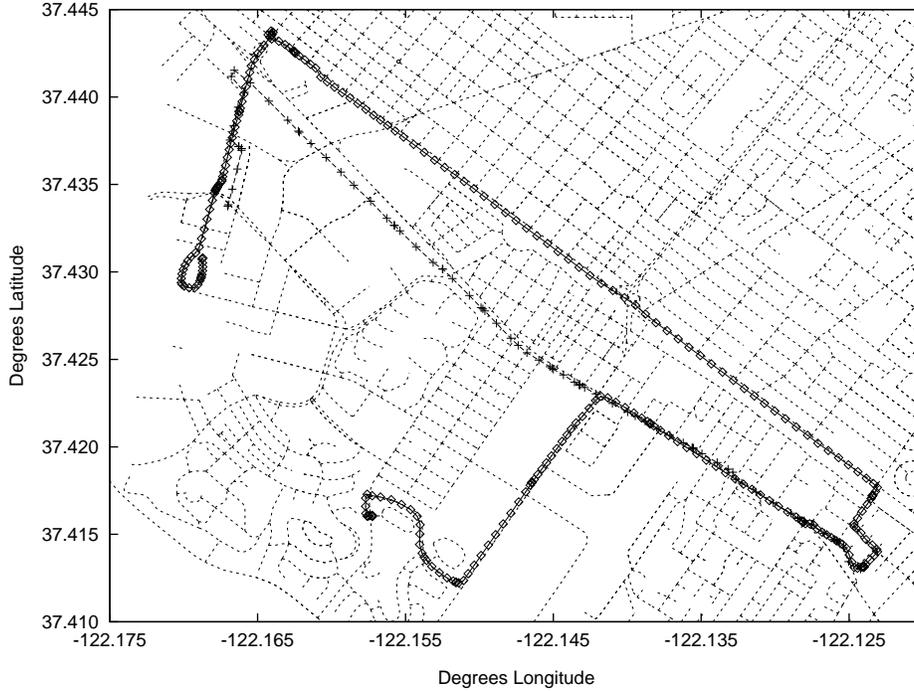


Figure 1: Two sample traces of raw GPS positions in the San Francisco Bay Area with a digital map superimposed.

The map matcher lets the system focus on a single segment at a time, but the problem of modeling the lane structure for that segment remains. If we assume that vehicles in the center of different lanes will always be a certain distance apart, we can use a clustering technique to separate positions into lanes. Spatially clustering the positions using an algorithm such as  $k$ -means will not work, because two points in the same lane may be spatially distant. However, if two points are less than half a lane apart, they are probably in the same lane. Similarly, if a point is within half a lane of any point in a lane cluster, it is probably in that lane.

We could use this intuition to “grow” lanes by initializing each point to be its own lane, then merging lanes where a point from one lane is within half a lane of a point from the other. This algorithm is similar to hierarchical agglomerative clustering (Hartigan, 1975), but it represents the clusters by their constituent points instead of a statistical average. However, this algorithm does not take advantage of the knowledge that lanes are constrained to be parallel to each other and the segment centerline. Figure 2 shows a representative road segment and its parallel lanes. If we had an accurate representation of the segment centerline, the perpendicular distance, or offset, between a lane and the centerline should be constant. This allows us to represent a lane with a single value,

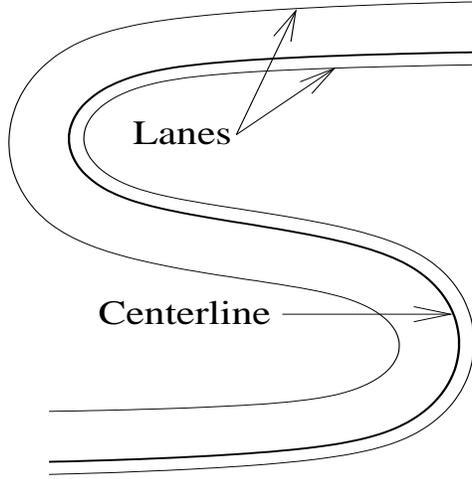


Figure 2: A generalized roadway model that, instead of modeling each lane independently, specifies the center of the road and describes each lane as a constant offset from the center.

substantially reducing the dimensionality of the space of lane models. We can state an algorithm that builds a lane structure for a segment using this approach as:

1. Let a lane be an offset from the segment centerline.
2. Let  $D$  be half the minimum distance between lanes, about 1.5 meters.
3. Initialize the segment with  $n$  lanes, one for each point's distance from the segment centerline.
4. While there is more than one lane,
  - (a) Let  $d$  be the difference in offset between two lanes  $(P, Q)$ .
  - (b) Find the pair of lanes  $(P_{min}, Q_{min})$  with smallest  $d$ ,  $d_{min}$ .
  - (c) If  $d_{min} < D$ , merge  $P_{min}$  and  $Q_{min}$ .
  - (d) If  $d_{min} \geq D$ , exit while loop.

The challenge in this approach lies in finding a sufficiently accurate segment centerline. The Navigation Technologies digital map includes shape information on segments, represented as a sequence of two or more points consisting of latitude/longitude pairs. If we use the piecewise linear curve connecting these points as the road centerline, we can compute offsets and cluster them. Unfortunately, experimental evidence shows that lanes are far from parallel to this curve,

because the digital map is not sufficiently accurate. For example, an analysis of a sample trace with no lane-changing shows offsets from -20 to 10 meters from the digital map centerline.

It may also be possible to use one of the position traces itself as a sufficiently accurate approximation of the segment centerline. For our purposes, the centerline does not need to follow the center of the pavement. Instead, any curve parallel to all the lanes suffices. To select the best trace, we could try all available traces and select the one that scores best on an evaluation metric of the resulting clusters. The standard deviation of each cluster is a plausible metric. The standard deviation for a correct clustering should be near the position accuracy, because all the points, except for points during lane changing, center around a lane. Clustering on uniformly distributed points produces the highest possible standard deviation, because points are just as likely to be far from as cluster as near. However, even the best trace for clustering according to this metric may not be suitable, because all traces may change lanes at some point, or all traces may be noisy. Clearly, a superior approach will take advantage of the parallel structure of lanes without relying exclusively on an inaccurate digital map or a single inaccurate position trace.

## **5 A task-decomposition approach to map refinement**

Although the centerline in the digital map is not accurate enough to compute constant lane offsets, any line parallel to the true lanes is, by definition, a constant distance from the lanes. We have devised a procedure to bring the centerline from the original digital map into alignment with the traces. The procedure computes the “average” between the current centerline and a new trace, weighted by the confidence in the centerline and in the trace. As the system incrementally incorporates more traces, it averages out errors in the traces to find a centerline more accurate than any of the traces that went into it. With this centerline refinement procedure, our approach to finding a lane model for a target segment  $S$ , covered by a set of position traces  $P$ , is to decompose the task into two subtasks:

1. Find an accurate centerline for  $S$  using  $P$ .
2. Cluster  $P$ 's offsets from the centerline into lanes.

Next we describe how the system combines new traces into segment centerlines and clusters offsets into lanes.

## 5.1 Finding an accurate road centerline

Existing digital maps represent the centerline geometry of a road segment as a widely spaced sequence of latitude and longitude points, with an advertised accuracy of 15 meters, connected by line segments. We also represent geometry as a sequence of points, but at a much higher density of 10 meters to allow finer control. We also add estimated standard deviations for longitude and latitude to represent confidence in the point. Connecting the points by linear interpolation is sufficient for low-curvature highways, but for roads with higher curvature, for better accuracy, or to reduce storage requirements, higher-order interpolation is possible. We have not addressed the issue of space efficiency of map representation.

The geometry refinement procedure iteratively improves the road geometry of a segment by performing a weighted average on the digital map with each trace. The map improvement process takes the current description of a map segment and a position trace corresponding to that segment, and produces a new and improved segment. Figure 3 illustrates the map improvement process for a short segment of map points. For each map point  $m$  with standard deviation  $m_\sigma$ , the procedure first finds the nearest point  $n$  on the trace by linearly interpolating between the GPS trace points. The standard deviation  $n_\sigma$  is the weighted average of the standard deviations of the surrounding GPS points. The new map point  $p$  is the average of  $m$  and  $n$  weighted by  $m_\sigma$  and  $n_\sigma$ , and the new standard deviation is

$$p_\sigma = \sqrt{\frac{m_\sigma^2 \cdot n_\sigma^2}{m_\sigma^2 + n_\sigma^2}}.$$

The net effect of these calculations for each point in the digital map is a weighted “averaging” of the map with a position trace. If the mean of the error distribution for the probe vehicle positions is zero, as assumed, then the weighted average will become more accurate as the number of traces increases. An interesting property of this procedure is that it does not compute the centerline of the road *pavement*, but instead weights the centerline toward the most-traveled lane. For example, if most vehicles travel along a segment in lane two and some in lane one, the centerline will be closer to lane two. Since the centerline is still parallel to the lanes, this property is not a serious issue.

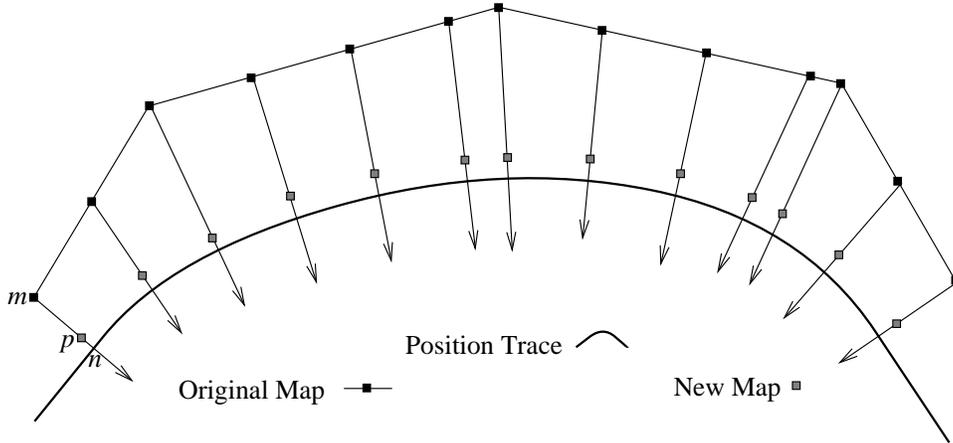


Figure 3: “Averaging” a map and a position trace. In this case, the map is the baseline digital map with points interpolated every 10 meters. In general, the map line segments may not be collinear. The new map point is the average of the current map point and the closest trace point, weighted by the confidence in the map and the trace.

## 5.2 Clustering offsets into lanes

To induce a lane model of a particular segment, we assume the system has an accurate geometrical representation of the centerline of the road, and that all lane centerlines are parallel to the centerline.<sup>2</sup> Since the lanes are parallel, the only parameter for each lane is the perpendicular distance to the centerline, which we call the *offset*.

We assume drivers generally are in a lane, so the perpendicular distance from most positions to the road centerline is an estimate of the offset for the lane. Since GPS is noisy and drivers are not always in the center of the lane, the mean of many samples should give a more accurate estimate of the true offset. Once the system has calculated an offset for each position in the position trace, the problem is to group these offsets into lanes and average them to find the lane centerline. Since the centerline is now accurate, we expect the hierarchical agglomerative clustering on offsets described in Section 4.2 to behave correctly.

Although our agglomerative clustering method is slow ( $O(n^3)$  in the number of offsets), it has two important properties:

---

<sup>2</sup>If the parallel assumption is not correct, this will become clear in subsequent analysis and the segment can be subdivided until the assumption is valid or the model can be changed.

1. It will never merge two lanes into one, because the procedure will terminate if the closest clusters are farther apart than the minimum lane width.
2. It makes no prior assumptions on the number of lanes.

If the system incrementally builds the lane models from the same centerline, we can dramatically improve the speed of the algorithm for each iteration with the results of the previous traces. If the previous iteration processed  $m$  offsets and found  $N$  lanes, with  $N \ll m$ , and the current iteration is processing  $n$  offsets, the original algorithm creates one lane for all  $m + n$  offsets. Instead, the incremental algorithm creates one lane for the  $N$  known lanes and the  $n$  new points. This version essentially integrates new offsets into the lane structure, instead of recomputing the lane structure from scratch. To avoid spurious lanes (e.g., from very noisy position data), we require that each cluster represent a certain percentage of the data. For our experiments, the threshold was one percent.

## 6 Evaluation of the approach

Evaluating our algorithms is a challenge, because ground truth is difficult or impossible to find. We decided to carry out a number of complementary evaluations. Since the system consists of two independent procedures, centerline refinement and lane clustering, we can determine the weaknesses and strengths of our overall system by examining the two components separately. The overall performance of the system is important for evaluating its commercial viability and the interaction of its components.

If our approach to learning lane models is viable as a cost-saving alternative to manually encoding lane structure, it must achieve “acceptable” performance without needing “excessive” training data. Although these terms depend on particular business assumptions regarding cost and profit, we can use learning curves to estimate how performance improves as more training data becomes available. Since our training data are fairly accurate and our algorithms are based on plausible geometric assumptions, we expect the learning curves to show that the system approaches its best performance on a segment after only processing a few traces that pass over the segment.

To test our algorithms and empirically investigate their behavior, we collected 44 position traces along a 15 kilometer section of Interstate Highway 280 between Redwood City and Palo Alto, California. The positions were calculated twice a second from a differential GPS system using a Novatel DGPS receiver and a CSI differential corrections unit obtaining corrections from the U. S. Coast Guard beacon network. The data were then matched to the commercially-available digital map to determine what segments each trace traversed. Since the traces did not follow the same path, different segments received different numbers of passes. Each of the 42 total segments of Interstate 280 in the target region received between 9 and 35 passes. We did not consider the difference in coverage to have a significant impact, so the results are averaged over all segments. All segments had four lanes, but all four lanes were not covered by any trace for a few segments. The traces generally stayed in one lane for the entire duration, and each point was tagged with the current lane, an integer from one to four.

## **6.1 Centerline refinement alone**

Since lane prediction involves two concurrent processes, we first tested each process in isolation. Centerline refinement is the most difficult algorithm to evaluate. The only objective evaluation is a comparison with the true centerline, but there are no means of measuring this centerline. Traditional surveying is impractical for busy public highways. Geo-referenced aerial or satellite photographs are alternative sources of raw data, but the data may be noisy, and the vision processing algorithms may not be reliable. Construction blueprints are available, but there is no guarantee that the road is actually built according to the plan. Additionally, all of these alternatives measure the center of the pavement, whereas our technique produces a centerline “weighted” toward the most common lane sampled. So even if the independent centerline measurement is very different from our own centerline, it is not clear if that makes a difference in the performance of the overall task.

Besides the final accuracy of the centerline, the rate of convergence is also of interest. Since the system is incremental, it can measure the difference between the centerline at each iteration and a reference centerline. If we plot the average difference between the current and reference centerlines for each iteration, the learning curve describes how quickly the centerline approaches the reference.

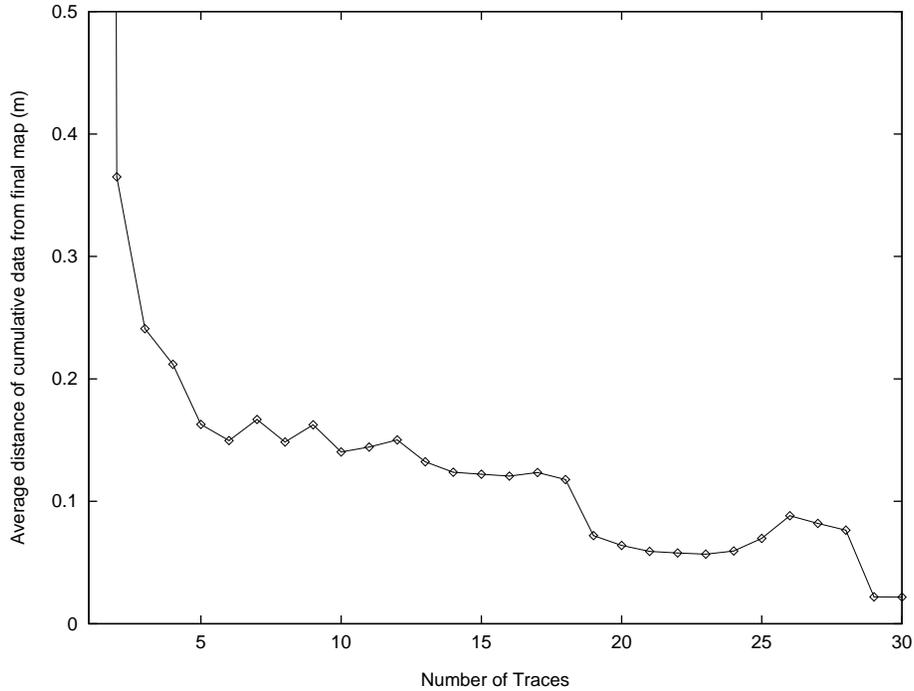


Figure 4: Centerline convergence. After an initial average error of 7 meters, the road centerline slowly converges to the final estimate.

This is of interest because it lets us estimate how many passes are necessary over a given segment before the centerline stops changing significantly. Ideally the reference centerline would be the true road centerline, but since the true centerline is unavailable we need an approximation. The best approximation available is the final centerline after the system has processed all traces. The rate of convergence in this case describes how quickly the centerline approaches the final result.

Figure 4 plots the difference between the centerline before each trace and the final centerline for a representative highway segment. The original map database was provided by NavTech, Inc., and has a stated accuracy of 15 meters for the road centerline. The plot shows that the major adjustment occurs on the first pass, where the baseline estimate is corrected by a GPS estimate with 1 to 2 meter accuracy. Processing the successive traces slowly improves accuracy by averaging out the noise in the GPS readings. Although there is no ground truth to measure the final accuracy, later processing critically relies on an accurate centerline, so good results in those evaluations imply a sufficiently accurate centerline.

## 6.2 Offset clustering alone

A detailed evaluation of the lane models is also problematic, since it is difficult to measure a vehicle’s true position within a lane, but a rough evaluation of the lane models is possible. Although regular training data from the field will be unlabelled, we labelled our data for testing purposes only. The label indicates the lane that the vehicle occupied for the given position. The system can find which cluster is closest to the position, and test if the cluster matches the label. For example, if a position’s perpendicular distance to the centerline is 2.1 meters, and the closest cluster is centered at 2.0 meters, the system predicts that the vehicle is in the lane corresponding to that cluster. Although overall accuracy is important, the learning curve is also important here, because we want to know the minimum number of passes over a segment that yields acceptable results. For this experiment, instead of a fixed testing set evaluated against increasing amounts of training data, we incrementally treat each position trace first as testing data against the current lane models, then as training data to refine the lane models.

The remaining issue is matching clusters with labels. In our tests, we used integer labels starting at one for the rightmost lane. In our coordinate system, offsets increase as they move left, so smaller offsets correspond to lower lane numbers. Therefore, the evaluation matches the cluster with the smallest offset to the smallest lane label seen so far in training. It matches the cluster with the second-smallest offset to the second-smallest lane label seen so far, and so on. For example, if all the training data have come from lanes two and four, the system maps the smallest cluster to lane two and the second-smallest to lane four. If an offset is closest to any other cluster, it is automatically wrong. This means that if there is a spurious cluster with a very small offset, all other clusters will be “bumped” to the next lane label, probably making them all incorrect. Fortunately, this is not likely to happen, because the clustering algorithm deletes all clusters representing less than one percent of the data.

We evaluated the lane clustering process by assuming the best centerline model we have—the result of centerline refinement on all 44 traces. With this centerline for all highway segments, we tested the cumulative lane prediction accuracy. For each trace, the system calculated the offsets from the centerline, then integrated the offsets into the current lane clusters using the incremental clustering algorithm presented in Section 5.2. For example, if all the offsets in the current trace

were between 5.0 meters and 5.5 meters, and clustering previous traces had produced a lane at 5.1 meters, the system would predict that the vehicle was in this lane for all positions in the trace. The system would then update the lane by agglomerating the offsets into the lane cluster.

The accuracy of the trace is the percentage of positions in the trace whose nearest cluster matches its lane label. As the lanes get more data, the lane centers become more accurate and lane prediction accuracy improves. Figure 5 plots the average accuracy of the clustering algorithm over 50 random orderings of the traces. Surprisingly, the results are initially quite good, then drop slightly for a few traces. By the 44th trace, the performance is at or slightly higher than the initial level. We believe the initial good performance is due to our procedure for matching clusters to lanes. Since there are often samples of only one lane early in the experiment, the clustering algorithm will probably create only one cluster, and the mapping guarantees that the only cluster maps to the only tag, giving 100% accuracy. As more data become available, there are more clusters and more possibility of error. Overall accuracy probably never reaches 100% because of noisy GPS data and mislabelled points. These results encourage us to believe that, given an accurate centerline for a segment traversed by several traces, our system can confidently predict a vehicle's lane.

### **6.3 Combined performance**

The final study combined the centerline refinement and the lane clustering processes. This experiment is most similar to how we expect to actually deploy the system, because the system initially generates lane models with no information beyond the baseline digital map. The procedure was similar to lane clustering alone, except the system computed the offsets of the first trace from the NavTech baseline. After computing the offsets and evaluating the predictions for each trace, the system refined the digital map centerline with the trace. We expected the results to be poor at first because of the inaccurate centerlines, but quickly approach levels in the previous experiment as the centerline improved. Figure 6 plots the average accuracy of the interleaved processes over 50 random orderings of the traces. As expected, the early results were poor, although somewhat above chance.<sup>3</sup> Starting at the fifth trace, the combined algorithm performed comparably to clustering on the most accurate centerline.

---

<sup>3</sup>Guessing one of four lanes is correct 25% of the time, although the clustering algorithm was not constrained in the number of clusters it could generate.

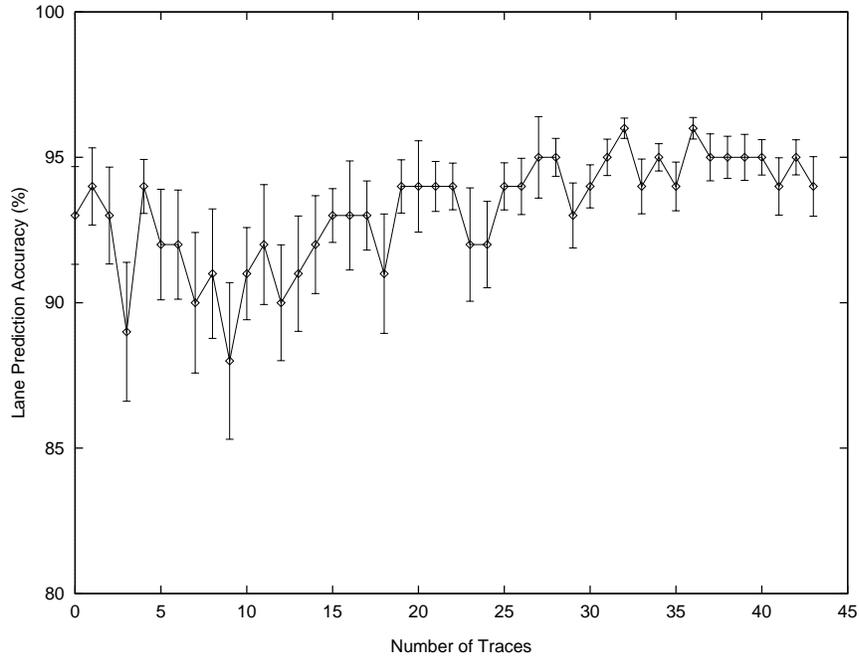


Figure 5: A learning curve for clustering offsets from an accurate centerline. Accuracy is high at the beginning stages (since there is generally only one cluster), drops as more clusters are seen, and rises again as the clusters become more accurate.

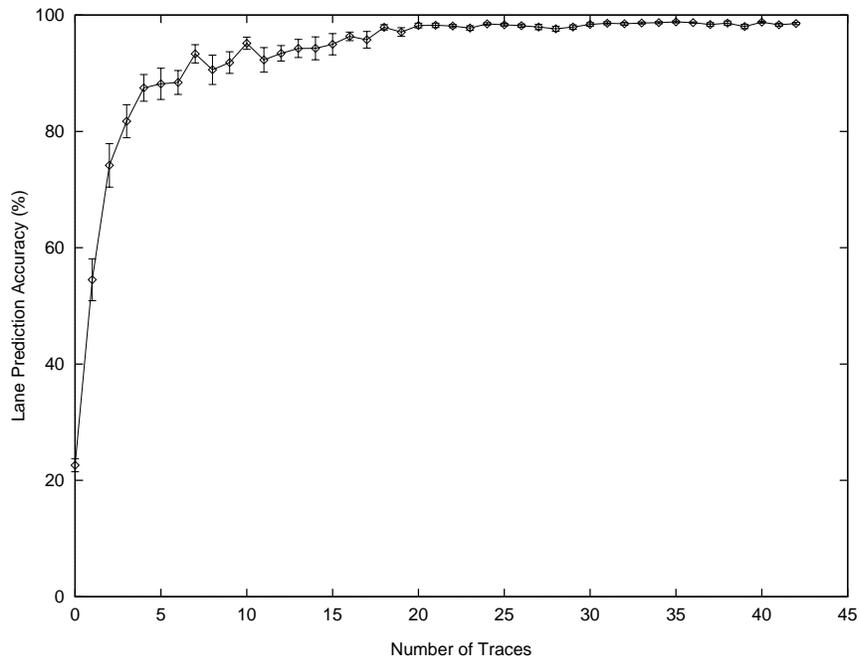


Figure 6: A learning curve for interleaved clustering and centerline refinement. Accuracy is low for the first few traces, then the centerline becomes accurate enough to provide correct offsets for the clustering algorithm.

The results of this experiment show that, starting with baseline geometry that is commercially available, it is possible to generate an accurate road centerline and lane models after a few high-precision GPS passes. Our position recorder is compact and robust enough to operate unattended in any car with a cigarette lighter socket, and our algorithms make no assumption regarding particular route or lane changing characteristics, so an entire city highway network could be modeled by distributing a number of recording units to vehicles. The vehicles, acting as probes during their normal driving patterns, accumulate information about the highway network. Once a vehicle has sufficiently sampled its driving patterns, it can relay its data to a centralized mapping service, either through a wireless transmission or by physically bringing the recorder to the center. The speed at which coverage and accuracy of the digital map improves is proportional to the number of recording units in operation. This is a low-cost technique for automatically mapping highways with high geometric accuracy.

## **7 Directions for future work**

This paper presents a realization of our methodology: to provide support for safety and convenience applications via unsupervised learning algorithms operating on anonymous probe vehicle traces. Our results are good for the limited data sample we collected, but we need to study and improve the robustness and autonomy of our algorithms. Our final goal is to let the centerline refinement and lane clustering processes operate unattended, receiving GPS data from probe vehicles and refining digital maps. However, we need more analysis of different types of driving conditions, such as curved roads, and tools to measure data quality in the absence of labels to have sufficient confidence in our models.

We also plan to make more comprehensive evaluations of our lane models using prototype vehicular applications. These applications will indicate the commercial viability and effectiveness of our approach. Our initial application is a simple lane position task that recognizes the position of the vehicle relative to the lane structure of its current road segment. This application is simple enough for rapid development, but relevant to complete deployable applications such as lanekeeping and lane departure warning.

This study focused on road centerlines and lane models, but data mining over position traces can yield many more types of geospatially specific knowledge, particularly when paired with a geographic information system. Virtually any database with a geographic component, such as records about how often a vehicle comes near different types of locations, can benefit from a suitably large set of position traces. Elsewhere we have reported work on predicting traffic controls (Pribe & Rogers, 1999) and travel times (Handley, Langley, & Rauscher, 1998). Since position traces are inherently individual, we are also developing methods to construct personal digital maps, with features such as preferred routes and typical speeds. It is now possible to quickly and cheaply accumulate volumes of position traces that let one annotate objects in a geographic database with real-world behavior. This capability has the potential for impact on many applications areas, from safety to navigation to marketing.

## Acknowledgments

We thank Huy Ton and Michael Jahr, who assisted in initial algorithm development and coding.

## References

- Chen, M., Jochem, T. M., & Pomerleau, D. A. (1995). AURORA: A vision-based roadway departure warning system. In *Proceedings of the IEEE Conference on Intelligent Robots and Systems*. Pittsburgh, PA.
- Gil, Y. (1992). *Acquiring domain knowledge for planning by experimentation*. Doctoral dissertation, School of Computer Science, Carnegie Mellon University.
- Grejner-Brzezinska, D. A. (1995). Positioning accuracy of the GPSVan. In *Proceedings of the 52nd Annual National Technical Meeting of the Institute of Navigation* (pp. 657–665). Palm Springs, CA.
- Handley, S., Langley, P., & Rauscher, F. (1998). Learning to predict the duration of an automobile trip. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 219–223). New York: AAAI Press.
- Hartigan, J. A. (1975). *Clustering algorithms*. New York: Wiley.

- Langley, P., Drastal, G., Rao, R. B., & Greiner, R. (1994). Theory revision in fault hierarchies. In *Proceedings of the Fifth International Workshop on Principles of Diagnosis* (pp. 166–173). New Paltz, NY.
- Moriarty, D. E., & Langley, P. (1998). Learning cooperative lane selection strategies for highways. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 684–691). Madison, WI.
- Ourston, D., & Mooney, R. J. (1994). Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66, 311–344.
- Pearson, D. J. (1996). *Learning procedural planning knowledge in complex environments*. Doctoral dissertation, University of Michigan, Ann Arbor, MI.
- Pomerleau, D. A. (1995). RALPH: Rapidly adapting lateral position handler. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*. Detroit, Michigan.
- Pribe, C. A., & Rogers, S. O. (1999). Learning to associate driver behavior with traffic controls. In *Proceedings of the 78th Annual Meeting of the Transportation Review Board*. Washington, DC.
- Teller, S. (1998). Automated urban model acquisition: Project rationale and status. In *Proceedings of the Image Understanding Workshop* (pp. 455–462). Monterey, CA.
- Wang, X. (1996). *Learning planning operators by observation and practice*. Doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.
- Weisenburger, S., & Wilson, C. (1999). An integrated vehicle positioning system for safety applications. In *Proceedings of the 56th Annual National Technical Meeting of the Institute of Navigation*. San Diego, CA.
- Wilson, C. K. H., Rogers, S., & Weisenburger, S. (1998). The potential of precision maps in intelligent vehicles. In *Proceedings of the IEEE International Conference on Intelligent Vehicles* (pp. 419–422). Stuttgart, Germany.