Real-Time Trajectory Refinement with Probablistic Rules

Taylor Lloyd
University of Alberta
tjlloyd@ualberta.ca

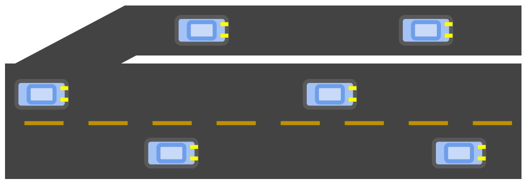

Figure 1: Motivating Example: Differentiating vehicles on nearby parallel roads

1. Introduction

Many systems today keep track of objects in real-time through positioning technologies such as the satelite Global Positioning System (GPS)[1]. However, positioning technologies have inherent error, which can range from centimeters to tens of meters. Consider the issue of tracking the number of vehicles exiting a highway to an adjacent, parallel frontage road (Figure 1). All vehicles are still going in the same direction, at similar speeds. The difference in relative position could easily be within the error of GPS under poor conditions.

This problem can be handled in a number of different ways. If additional sensors can be used, then disparate information can be combined to produce more accurate results, known as *sensor fusion* [6]. Alternatively, historical sensor information can be used to refine new sensor inputs. Finally, rules can be introduced to help clean up random information. Returning to our example, if the vehicle velocity drops below highway speeds over a period of time, it could be assumed to have left the faster roadway.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, contact the Owner/Author(s). Request permissions from permissions@acm.org or Publications Dept., ACM, Inc., fax +1 (212) 869-0481.

PLDI '17 Month d–d, 20yy, City, ST, Country
Copyright © 2017 held by owner/author(s). Publication rights licensed to ACM.
ACM 978-1-nnnn-nnnn-n/yy/mm. . . $15.00
DOI: http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

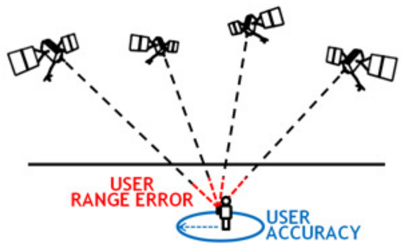

Figure 2: From [1]: GPS Technology Overview

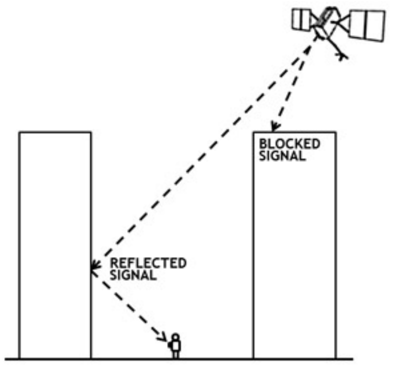

Figure 3: From [1]: GPS sources of error - reflected signals.

This work combines historical information with probablistic rules, using techniques similar to sensor fusion to combine continuous probabilities. Continuous probability spaces of position and velocity are approximated with 4-dimensional meshes, which can be efficiently combined to refine location estimates.

2. Background

2.1 GPS Positioning

GPS Positioning technology makes use of satelites in known orbits around the earth, broadcasting the current time. By consulting the orbits and the time taken for a signal to arrive from a number of satelites, GPS recievers can determine their own location on Earth, as shown in Figure 2.

However, GPS cannot produce perfectly precise locations. Error is introduced by a variety of factors: Orbital information can be out-of-date or subtly incorrect, or too few satelites may be within reach. Figure 3 shows a more dramatic problem, in which the signal recieved is bounced off a nearby object before being received.

2.2 Path Refinement

GPS positions inherently include measurement error, as mentioned above, but naive attempts to interpolate between positions can cause dramatically more error if done poorly. Lonergan *et al.* demonstrated that point-connecting strategies such as linear or quadratic interpolation have error that scales linearly with the time between GPS positions when tracking wild animals [5].

Different strategies can be applied to refine positions within a *path*, depending on the goal. If attempting to determine the length of the path, map-matching techniques allow a path to be overlaid on a known (and presumably accurate) network. Conversely, if the goal is to extract a more accurate path, sensor readings can be combined with additional information and rules.

Path refinement is extremely difficult to evaluate, as the underlying premise is that the sensor data provided is inaccurate. As a result, evaluation of such techniques tends to focus around path *recoverability*, in which the sensor data is randomly permuted across trials, and the similarity of the refined paths is compared.

3. Related Work

Related work tends to fall into distinct categories. Kalman filters allow for sensor integration with historical data, but rely on all inputs to be expressible as gaussian probabilities, or as absolute certainties. 2-dimensional probability grids have been used in robotics to assist with self-location, but existing techniques use the existing model as history only, rather than as a tool to enable the use of additional rules. Finally, map-matching techniques attempt to attach all sensor positions to a known network, typically to enable routing applications.

3.1 Kalman Filters

Kalman Filters [4] allow for the efficient combination of sensor data and historical state by modelling all inputs as gaussian uncertainties. In addition, it can accommodate for unknown or random inputs by reducing certainty as forward propagation continues. However, because many rules cannot be expressed with gaussian uncertainty, Kalman filters are unable to be used for our proposed application.

3.2 Probability Grids

Probability grids [3] have been used in robotics to enable memory of environments, both for position estimation and for environment recognition. However, such grids are historically used to learn an environment, and tend to track probabilities other than the current location. This work uses probability meshes to track the object's current position in real-time.

3.3 Map Matching

Map-Matching [2] is the process of mapping arbitrary locations onto a network. As a typical example, GPS positions are mapped onto the road network, to allow for routing and turn-by-turn directions. Most map-matching algorithms attempt to map current positions to networks, though some are designed to match an entire path. When an entire path must be matched to the network, greedy strategies are typically employed, occasionally with lookahead to reduce error. This work makes use of probability meshes to approximate a 4-dimensional (position,velocity) space. The meshes are then used to implement efficient forward estimation, integration of new sensor data, and integration of probabilistic rules. Finally, a pipeline is constructed from these elements to produce refined paths from raw positional input.

4. Probability Meshes

Each trajectory of interest t has an associated 4-dimensional $n*m*i*j$ probability mesh M_t, with each cell $M_t(n, m, i, j)$ in the mesh representing the likelihood that the most recent position and velocity in the trajectory lie in that cell. However n, m, i, and j are integer values while locations and velocities are continuous. In order to account for continuous values, an interpolation scheme can approximate points between cell centers.

Interpolation works as follows: For each dimension i in a query point $q = (q_n, q_m, q_i, q_j)$, calculate $\mathit{floor}(i)$ and $\mathit{ceil}(i)$. Then collect the set of adjacent points P by generating all combinations of $floor$ and $ceil$ in each dimension. Interpolation is calculated over P as shown in Equation 1.

$$\frac{\sum_{p\in P} 1/\mathit{dist}(p,q) * M_t(p_n, p_m, p_i, p_j)}{\sum_{p\in P}(1/\mathit{dist}(p,q)} \tag{1}$$

With interpolation between cells, probability meshes can approximate a bounded continuous probability. Note that the division by *dist* is undefined when dist is zero. In practice, that can only occur when all indices are integers, and in that case the exact point found is returned. This interpolation algorithm has a property that will be heavily utilized in the remainder of the paper: Because all probabilities are weighted by *dist*, probabilities generated from a set of points are bounded by the minimum and maximum values stored in those points. Therefore, the point of maximum probability *must be an exact grid cell.*

However, the grid indices have no mapping to real-world positions and velocities. To provide mappings, boundaries and a mapping function must be specified. Given an input position and velocity (p_x, p_y, v_x, v_y) and boundaries $x_{min}, x_{max}, y_{min}, y_{max}, v_{min}, v_{max}$, the appropriate grid point is calculated by Equation 2.

$$map(p_x, p_y, v_x, v_y) = \left(\frac{(p_x - x_{min}) * n}{x_{max} - x_{min}}, \frac{(p_y - y_{min}) * m}{y_{max} - y_{min}}, \frac{(v_x - v_{min}) * i}{v_{max} - v_{min}}, \frac{(v_y - v_{min}) * j}{v_{max} - v_{min}}\right) \tag{2}$$

With the inclusion of the mapping function and boundaries, a probability mesh can approximate a continuous probability over arbitrary boundaries. In addition, the mapping function can be reversed to $map_{reverse}$, calculating the corresponding point in the probability space for a particular cell in the mesh.

4.1 Forward Propagation

Because these probability meshes capture both position and velocity, future probability spaces can be estimated. Equation 3 shows how a zeroed mesh M'_t can be calculated from a past mesh M_t.

$$M'_t(n+(t'-t)i, m+(t'-t)j, i, j) += M_t(n, m, i, j) \tag{3}$$

M'_t represents an accurate estimate, assuming unchanging velocities. However in the real world objects can change velocity over time. To accomodate this, a blur factor can be added by averaging all adjacent points using a stencil algorithm.

5. Probabilistic Rules

Often in addition to sensor input, we have domain knowledge about objects whose positions and velocities are being tracked. Such rules can often be colloquially expressed as "Cars tend to drive on roads", or "Trucks and busses rarely make U-turns". Under a traditional path refinement system such as a Kalman Filter, these rules cannot be expressed, as they represent non-gaussian probabilities. However, using probability meshes, these rules can be approximated and used to refine any sensor inputs or estimates. In order for a rule to help refine paths, it need not always be true. The examples above discuss things that usually occur, which can be formally defined as a probability. By phrasing rules as probability functions in the same 4-dimensional position, velocity space as the meshes defined above, such probabilistic rules can be expressed.

A rule R can be intersected with a mesh M by applying the pointwise update equation – Equation 4 – to all cells in the mesh, producing a new mesh M'. Multiplication allows the probabilities to be combined to generate the probability of both the original mesh and the rule being true for each cell.

$$M'(n, m, i, j) = M(n, m, i, j) * R(map_{reverse}(n, m, i, j)) \tag{4}$$

5.1 Road Matching

As an example of probabilistic rules, this work implements the rule *"vehicles usually drive on roads"*. The exact equation for this rule is in Equation 5.

$$R_{road}(p_x, p_y, v_x, v_y) = max(\alpha, CNDF(distanceToRoad(p_x, p_y)/\beta)) \tag{5}$$

α allows for this rule being incorrect, by setting a lower bound on the probability of any cell, regardless of roadways. β allows the allowed distance from the roadway to be scaled. Here CNDF refers to the cumulative normal distribution function, which returns a value between 0.5 and 0 for positive inputs.

The *distanceToRoad* function requires additional explanation. Road map data for the city of Edmonton was aquired from OpenStreetMap, and roadways were extracted. Then, to reduce the number of roadways inspected for a particular calculation of this rule, a uniform grid index is overlaid on the area of interest, and roads are placed into each grid cell overlapped by their minimum bounding rectangle (MBR). Grid cells should be of a width substantially larger than β above, so that a minimum of cells need to be inspected.

When calculating *distanceToRoad* for a given point, all roads in the corresponding index cell are collected, as well as all roads in adjacent cells to account for query points near the edge of a cell. Then, the distance from each segment of each road to the query point is calculated. Finally, the minimum distance found is returned. An example of the probability space rendered from this work is shown in Figure 4.

6. Grid-Based Trajectory Refinement

From probability meshes and probabilistic rules, a pipeline can be assembled to refine trajectories in real-time. Consider a set of objects being tracked, O. Each object $o \in O$ has an associated probability mesh $M(o)$, valid for a particular timestamp in the past $updateTime(o)$. All probability meshes are initialized to an equal uniform distribution. As each sensor update $(o, time, p_x, p_y)$ comes in, meshes are updated according to the following pipeline:

1. The object mesh is propagated forward in time according to Equation 3, using the time difference *time* − *updateTime*(o).
2. A gaussian probability *GPS* is constructed around p_x, p_y, constructed to match the error distribution that 95% of sensor readings are within 2 meters of accurate. The GPS probability is integrated into $M(o)$ according to Equation 4.

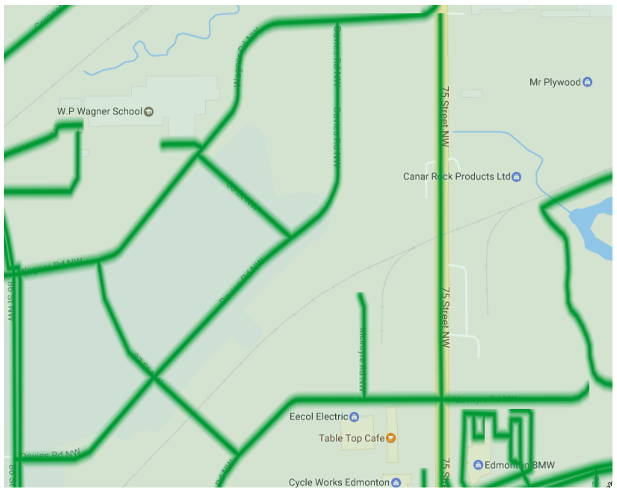

Figure 4: Example rule "Vehicles usually drive on roads" overlaid on Google Maps

3. Any applicable rules can be integrated, again using Equation 4.
4. The new most likely position is found by finding the maximum cell in mesh $M(o)$ and reverse mapping the cell into the real-world position.

7. Evaluation

In order to evaluate this technique, 1 hour of location data was captured for 4 busses in the Edmonton Public Transit System. This data consists of 120 sensor updates per bus, or 1 every 30 seconds. All results were captured on a machine with 32 GB of RAM, and a Core i5 2700K processor.

Unless otherwise specified, all probability meshes are 256*256*9*9, and are mapped to the Edmonton City Limits, with velocities in the range (-100km/h,100km/h).

7.1 Optimal Grid Size

Using a finer spatial grid (n and m) allows for smaller changes in position to be represented, and for finer points to be selected as most-likely positions. However, in order to be useful for real-time refinement, it must be possible to update meshes at least as quickly as new sensor updates come in. This means it must be possible to update all meshes in under 1 hour.

By calculating recoverability between the raw GPS positions and the refined positions for a series of grid sizes, we can determine at what grid size the fidelity no longer contributes a substantial error. Because in this case refinement itself will appear to contribute error, we are looking not for a low absolute error, but a "levelling off" in which increasing size does not further reduce error. Error is presented here as the mean distance between points at the same timestamp, in meters.

Shown in Figure 5, substantial error is contributed by the 64x64 and 128x128 grids, but beyond that there appears to

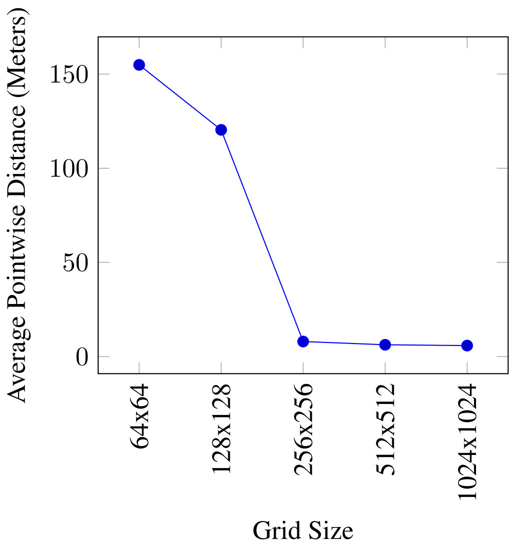

Figure 5: Refined Position vs Raw Position by Grid Size

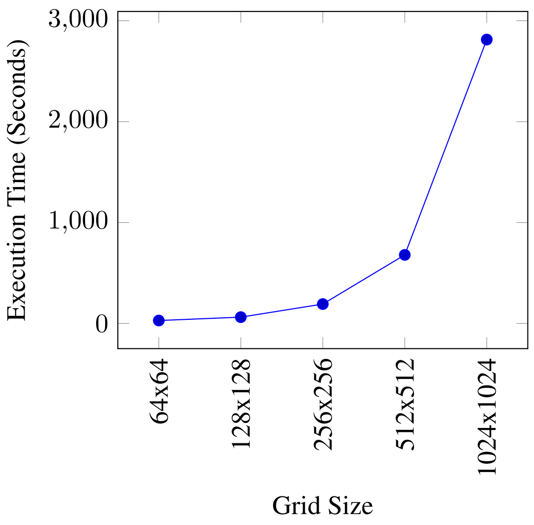

Figure 6: Execution Time by Grid Size

be negligible gain from further size increases. It may still be desirable to use larger grids, as the error continues to reduce, but execution time may prevent it. Figure 6 shows the time required to compute all 120 sensor updates for all 4 busses. While sizes up to 256x256 are relatively efficient to compute, it takes nearly the full hour to process the 1024x1024 grid.

7.2 Route Recoverability

Because GPS sensors are assumed in this work to be unreliable, and the resulting techniques operate on that assumption, it makes little sense to compare the refined paths to the original sensor data. Instead, we perform multiple runs, randomly permuting GPS positions within specified radii. By

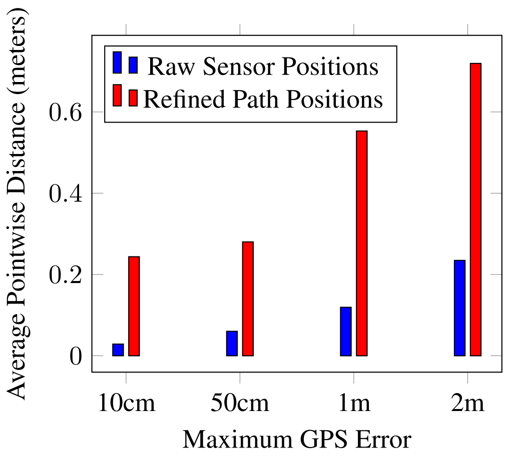

Figure 7: Refined Error vs Raw Error by GPS Error Radius

comparing the error from raw sensor input to the error in the refined path, we can evaluate the effectiveness of these techniques in accomodating for errors. Figure 7 shows the resulting information.

Clearly visible in the figure, path refinement fails at all error levels to produce a more recoverable path than just using the original sensor readings. However, as GPS error increases, the gap begins to close in relative terms. At 10cm of GPS error, the refined error is 8.4x greater, while at 2m of GPS error, the refined error is only 3x greater. At further inspection, this error occurs for two main reasons:

1. **Error Magnification:** The probability meshes can express maximum likelihood points only at exact grid cells. Thus, a small change in raw position can cause the refined path to move the position an entire cell. Since cells correspond to positions multiple meters apart, this magnifies small changes into large changes. Though most cells are unchanged, a few errors skew the result.
2. **Off-the-grid Error:** One bus being tracked actually leaves the City of Edmonton geographic limits for a portion of its route. During this portion, the probability mesh is forced to approximate its location as at the nearest edge.

These issues, along with the reducing gap in recoverability as the error grows more dramatic, seem to indicate that a more precise probability mesh would solve much of the issue in refined recoverability.

8. Conclusion

As presented, probability meshes for sensor refinement are not suitable for managing error in GPS positions at the scale of a full city. At sizes that can be processed in real time, the fidelity of the mesh is too poor to meaningfully improve over raw sensor positions, and at sizes that can produce high-quality results processing is too slow to keep up with incoming data.

9. Future Work

While this work has so far been unsuccessful, there are a variety of promising avenues to explore. First, the major limiting factor has been the uniform grid overlaid on the city. Adaptive grids, either remaining uniform but moving with the current best estimate, or with differing fidelity depending on the current estimate, would allow for dramatically faster computation with improved resolution.

Alternatively, larger meshes could be made feasible through the integration of GPU computing. Updating large meshes in parallel is a key GPU feature, and could increase the resolution of real-time meshes by orders of magnitude.

If resolution can be improved to the point where this technology can outperform raw sensor input, there is a vast array of rules that can be implemented and evaluated. In the context of public transit, busses typically stay on their routes, which are publicly available. Speed limitations for roads could be applied, and one-way roads could be encoded based on velocities. By integrating additional rules, more sensor error can be eliminated.

Finally, while applying rules provides more semantically likely routes, it also accurately identifies discrepant behaviour. In all the work here, only the most-likely point is considered. However, the relative likelihood of that point encodes valuable information about the behaviour of the object. A low likelihood identifies an object that is diverging from expected behaviour, which could be useful in additional applications.

References

[1] GPS.gov. `http://www.gps.gov`. Accessed: 2017-04-08.

[2] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Very Large Data bases (VLDB)*, pages 853–864, 2005.

[3] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the national conference on artificial intelligence*, pages 896–901. Citeseer, 1996.

[4] R. E. Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic engineering*, 82(1):35–45, 1960.

[5] M. Lonergan, M. Fedak, and B. McConnell. The effects of interpolation error and location quality on animal track reconstruction. *Marine Mammal Science*, 25(2):275–282, 2009.

[6] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9(2):61, 1988.