Automated Track Projection Bias Removal Using Frechet Distance and Road Networks

Lingji Chen and Ravi Ravichandran
BAE Systems
Burlington, MA, USA

***Abstract*—When target tracks produced by a video tracker are projected to the Earth's surface, they often become a (slightly) rotated, stretched and translated version of the true tracks due to sensor calibration errors. If the correspondence between the projected tracks and the roads they are on can be established, a homography can then be computed and the projection bias can be removed. This correspondence is typically easy to establish by a human operator; however, our objective is to seek an automated solution to reduce an operator's work load, and the problem is challenging due to the lack of fiduciary targets. Building upon the body of research in the GPS community, the computer vision community as well as the tracking community, we have developed a new algorithm to compute a discrete Frechet distance from a polygonal curve to a planar map, and use it to automatically establish the above correspondence and remove the projection bias. Simulations with synthetic data show the efficacy of this innovative approach.**

I. Introduction

Target tracks produced by EO/IR sensors in the image plane often need to be projected onto the surface of the Earth for subsequent display or for fusion with tracks produced by other sensors. A bias in the parameters used for this projection, for example, heading or pitch of the platform that carries the sensors, leads to a projective bias in the ground tracks: They are typically a rotated, stretched and translated version of the true tracks. Figure 1 shows a simulated example using the road networks of the Greater Boston area, where true tracks are shown in green and biased tracks are shown in red; the details of data generation will be presented in Section V. It can be seen that on the upper left (near MIT), the red tracks are to the west of the green tracks, while on the lower right (in the downtown area), they are to the east. How to automatically remove such a bias in a batched processing step is the focus of this paper.

It is straightforward to show that if the tracks lie on an approximately flat surface, then the bias can be described and corrected by a homography that transforms the biased track positions to their true positions. Our aim is to automatically calculate this homography and make the correction.

The contribution of this paper is twofold. First, we have developed a new algorithm that extends the map matching algorithm in [1] to the discrete case, and the discrete distance in [2] to the case of a planar map. Second, we have integrated various algorithm components to solve the particular problem of automated bias removal without fiduciary tracks.

Track bias removal has been studied extensively and many methods have been proposed in literature. To provide some context to our particular problem formulation and approach, we mention a few published results here, which are by no means comprehensive. The problem of removing biases by the fusion center from state estimates produced by local, bias-ignorant trackers, when track-to-track association is known, is considered in [3]. The problem of calculating the probability of track-to-track association given data from biased sensors is considered in [4]. In [5], the problem of jointly obtaining the optimal track-to-track association and the estimation of the (relative, additive) sensor bias is formulated as a nonlinear mixed integer programming problem and solved using a multistart local search heuristic. The work reported in [6] is closely related to ours; there they obtain matchings of biased tracks to roads by treating both the road network and the tracks as binary images, and using feature finding methods in image processing. It can be seen from Figure 1 that, when roads are dense and similar, the most prominent features are often the turns (intersections). Our method can be thought of as a natural extension to [6] that considers not only the turns, but all points on a track/road. It is also interesting to note the work in [7], where measurements are filtered into tracks with road map assistance, and intersecting roads are handled with an interacting multiple model (IMM) scheme. More recent work that deals with bias estimation/removal includes [8], [9] and [10].

Our work also borrows from the large body of research in map matching from the GPS community, where a main problem is, qualitatively speaking, to put GPS dots (recorded by a device) back on the road. Interested readers are referred to [11], [12], [13], [14], [15], and the references therein. It is worth emphasizing that our problem formulation is a simpler one: There is a homography that relates the biased tracks to truth tracks, and map matching is used as an intermediate step for finding the homography, not as an end result.

The paper is organized as follows. Section II provides a summary of formulas used to find a homography from matched lines (as opposed to the canonical form of matched points). Section III presents a discrete map matching algorithm based on discrete Frechet distance, which to the best of our knowledge has not been reported in literature. Automated bias removal is summarized in Section IV, and the efficacy of the approach is illustrated in Section V using simulated tracks. Section VI draws some conclusions and also speculates on the future direction of this work.

Approved for Public Release; Distribution Unlimited. Cleared for Open Publication on 3/20/2014.

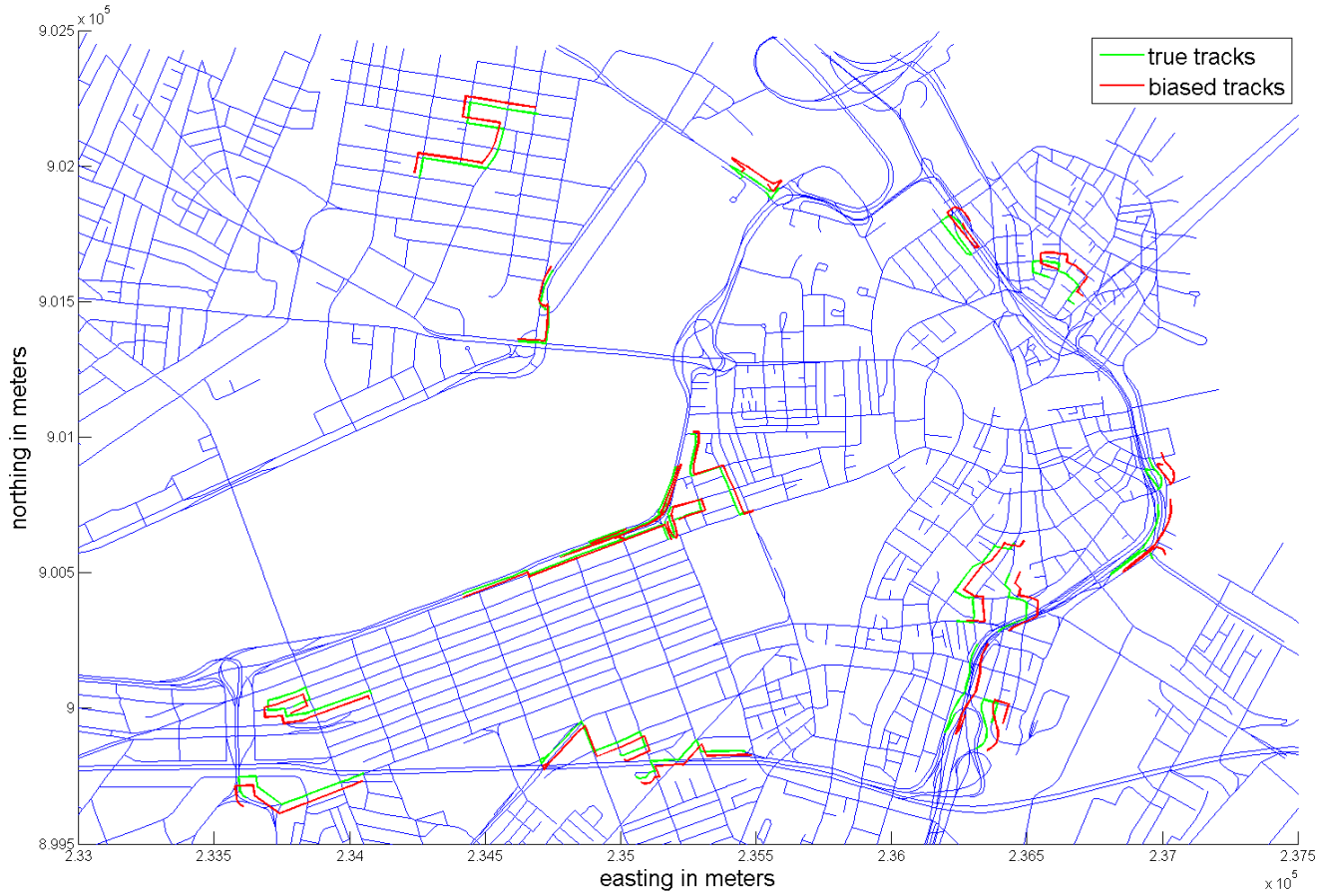

Fig. 1. Boston road network with simulated true tracks and biased tracks.

II. Finding homography from matched lines

Before we present an automated procedure, we first describe how the bias correction task can be performed manually. We can visually inspect a biased track and compare it with the road segments "nearby" on the given road network, and if we believe we have found a match, we mark the correspondence between the line segments in the tracks and the line segments on the road. Repeat this for many biased tracks, and we can obtain many pairs of matchings and solve for the homography through the matched lines.

We summarize here formulas from [16] that are relevant for the above task. Let $[x_1, x_2]^T$ be a point in a plane (where a superscript T denotes transpose). Since a line in the plane is represented by an equation such as $ax_1 + bx_2 + c = 0$, the line is identified with the homogeneous point $[a, b, c]^T$. With the homogeneous representation $\mathbf{x} \triangleq [x_1, x_2, 1]^T$ of the point $[x_1, x_2]^T$, we see that the point $\mathbf{x}$ lies on the line $\mathbf{l} \triangleq [a, b, c]^T$ if and only if their inner product is zero, i.e., $\mathbf{x}^T\mathbf{l} = 0$. In the following we will drop the bold face for vectors when the meaning is clear from context.

Let H be the 3×3 homography matrix that transforms a homogeneous point x into x', i.e.,

$$x' = Hx.$$

If l is a line passing through x, then

$$l' = H^{-T} l$$

is a line passing through x', because

$$x'^T l' = (x^T H^T)(H^{-T} l) = x^T l = 0.$$

Thus, if we treat l' and l as homogeneous points and find the homography that transforms the former to the latter, then the transpose of the found homography is H since

$$H^T l' = l.$$

We note in passing that when we identify a line segment l_{ab} using its two end points x_a and x_b, the homogeneous representation of the line is given by the cross product of the homogeneous representations of the end points, i.e.,

$$l_{ab} = x_a \times x_b.$$

It can be readily verified that x_a is on the line because $x_a^T l_{ab} = 0$, and so is x_b.

How to compute homography, especially in a robust fashion, can be found in [16]. Because of the availability of open source libraries such as OpenCV (Open Source Computer Vision Library) [17], we treat this as a "library calling" subproblem and will not go into its details here.

It is now clear that one way to automate the batched process of removing the projective bias discussed earlier is to automate the matching of biased tracks to the road segments on the road network. In the following we present a new algorithm that maps a polygonal curve to segments on a planar map based on the discrete Frechet distance.

III. Frechet distance and map matching

A. Continuous Frechet distance

The Frechet distance is a measure of similarity between curves that takes into account the location and ordering of the points along the curves. Intuitively, the Frechet distance between two curves is the minimum length of a leash required to connect a dog and its owner, constrained on two separate paths, as they walk without backtracking along their respective curves from one endpoint to the other [18].

More specifically [19], let $f : [a, a'] \to \mathbb{R}^2$ and $g : [b, b'] \to \mathbb{R}^2$ be curves. Then their Frechet distance is defined as

$$\delta_F(f, g) \triangleq \inf_{\alpha,\beta} \max_{t\in[0,1]} d\left(f(\alpha(t)), g(\beta(t))\right),$$

where $d(\cdot, \cdot)$ is a chosen distance function, and $\alpha : [0, 1] \to [a, a']$ and $\beta : [0, 1] \to [b, b']$ range over continuous and non-decreasing functions only (and represent a choice of walking the dog in our analogy).

Alt and Godau presented an algorithm to compute the Frechet distance between two polygonal curves in [19], based on the idea of a *free space*. If P and Q are two line segments, then the free space F_ϵ describes all pairs of points, one on P and one on Q, whose distance is at most ϵ:

$$F_\epsilon \triangleq \{(s, t) \in [0, 1]^2 \mid d(P(s), Q(t)) \leq \epsilon\}.$$

This concept is illustrated in Figure 2. The same concept is generalized to polygonal curves and illustrated in Figure 3.

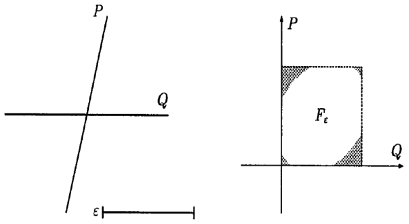

Fig. 2. From [19]. Left: Two line segments and a given distance; right: the corresponding free space in white.

It is proved that $\delta_F \leq \epsilon$ holds exactly if there exists a curve within the free space F_ϵ from the lower left corner to the upper right corner which is monotone in both coordinates. Thus the Frechet distance can be found through a search procedure, by checking at each iteration whether a connecting curve exists.

B. Discrete Frechet distance

The algorithm that Alt and Godau described to determine the existence of a connecting curve is quite elaborate and not straightforward to implement. A discrete counterpart has been proposed and studied in [2], where only distances between the end points of the polygonal curves are considered (and therefore the dog analogy has to be replaced by a "magical frog" analogy). More specifically, let P and Q be polygonal curves defined by the end points $(u_1, u_2, \ldots, u_p)$ and $(v_1, v_2, \ldots, v_q)$ respectively. We will also use the notation $P(i)$ to denote u_i. A non-backtracking discrete walk of P with length m is defined by a sequence of m indices $\alpha = \{\alpha_1 = 1, \alpha_2, \ldots, \alpha_m = p\}$ such that the walker either stops, i.e., $\alpha_{i+1} = \alpha_i$, or goes one step forward, i.e., $\alpha_{i+1} = \alpha_i + 1$. Then the discrete Frechet distance between P and Q is defined by

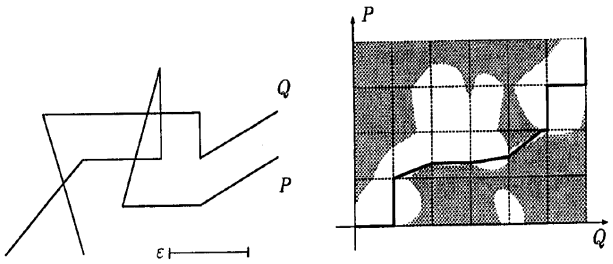

Fig. 3. From [19]. Left: two polygonal curves and a given distance; right: the corresponding free space in white.

$$\delta_{dF}(P, Q) \triangleq \inf_{m,\alpha,\beta} \max_{k=1,\ldots,m} d\left(P(\alpha_k), Q(\beta_k)\right),$$

where $d(\cdot, \cdot)$ is a chosen distance function, and α and β are non-backtracking discrete walks of P and Q with length m respectively.

It is easy to see that if we "subdivide" the two polygonal curves further and further, their discrete Frechet distance δ_{dF} should approach their continuous Frechet distance δ_F. In fact, the following bounds are established in [2]:

$$\delta_F(P, Q) \leq \delta_{dF}(P, Q) \leq \delta_F(P, Q) + \ell_{max},$$

where ℓ_{max} is the largest among the segment lengths of P and Q.

The discrete Frechet distance, and the corresponding walk (which is termed "coupling" in [2]), can be computed directly using dynamic programming; a MATLAB implementation can be found at [20].

C. Continuous map matching

Alt et al [1] also extended the concept of the free space to the case of a polygonal curve and a planar map, in order to find the matching line segments in the planar map that are closest to the polygonal curve according to the continuous Frechet distance. This idea is illustrated in Figure 4.

Once again, this algorithm which we call continuous map matching is quite elaborate and not straightforward to implement. It is desirable to use the discrete Frechet distance, but the dynamic programming approach for two curves does not seem to be readily extendable to the case of a curve and a map.

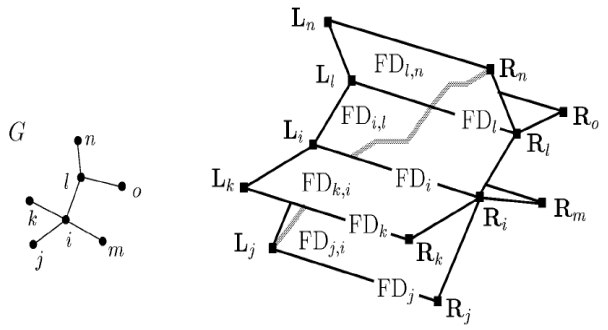

Fig. 4. From [1]. Left: The planar map G; right: The free space surface formed by the free space diagrams glued together. The graph G is "on the paper" and so are the "L" points. The "R" points are "sticking out of the paper." The gray curve illustrates a nondecreasing path through the free space.

D. Discrete map matching

We have developed a discrete version of the map matching algorithm by adopting the idea of the free space but restricting our attention to only the segment end points in the polygonal curves. This is achieved by first constructing a free space graph, which is illustrated in Figure 5 on Page 5. The construction is inspired by [21] for calculating discrete Frechet distance between two curves.

The construction of the free space graph can be qualitatively described as follows:

1) We are given a polygonal curve P and a planar graph (or *map*) G. Take the natural ordering of the n_P segment end points (or *nodes*) of P and mark the sequence on a horizontal line. Take any ordering of the n_G nodes of the map G and mark the sequence on a vertical line. We will create a free space graph with $n_G \times n_P$ nodes.
2) Each column of nodes in the free space graph is a replica of the nodes in G, and has a corresponding "node position" in P. See Figure 5.
3) Two nodes in the same column are connected if their "original" nodes are connected in the map G. Two nodes in adjacent columns are connected if they have the same "original" nodes, or if their "original" nodes are connected in G.
4) For a given distance parameter $\epsilon > 0$, mark each node in a column as
   - *free* (denoted by a solid rectangle in Figure 5) if its distance to the node on P that this column corresponds to is less than or equal to ϵ, or
   - *not-free* (denoted by a dashed ellipse) otherwise.

With the free space graph so constructed, it is straightforward to establish the following:

Proposition 1: There exists a path (consisting of line segments) in the map G with a discrete Frechet distance to the polygonal curve P smaller than ϵ, if and only if the free space graph constructed with ϵ as the distance parameter has a path consisting of only free nodes from the first column on the left to the last column on the right.

The existence of such a path can be decided by adding a source node to the left and a sink node to the right, and calling an algorithm such as Dijkstra's in a standard graph library. Optionally, edge weights can be added to the edges in the free space graph, so that among all valid paths that can establish the discrete Frechet distance upper bound, a "minimum length" path can be chosen that has the least total sum of distances between "coupling" nodes.

The discrete Frechet distance can then be found by a (binary) search on the distance parameter ϵ. More efficient parametric search methods are discussed in [22].

In Figure 5, the path is "1a $\rightarrow$ 2b $\rightarrow$ 3c," meaning that the matched line segments in the map is the polygonal curve specified by Nodes 1, 2 and 3. For this example, it is easy to see that the discrete Frechet distance from the track to the road network is equal to the distance between Node 3 and Node c.

IV. Automated bias removal

In this section we combine the results discussed earlier and present an outline of the automated bias removal algorithm:

1) Obtain road network data as a graph. Road network data is typically stored in a shape file that lists each road as a series of points. This has to be converted to a graph with unique points as nodes and with adjacency information between the nodes. Depending on the sampling interval of the tracks to be corrected, a long road may need to be further discretized to have more points on it in order to match the "resolution" of the tracks.
2) For each (biased) track, run the discrete map matching algorithm described in the last section to obtain its closest road segments. If the map is large, a "gating" procedure should be employed first to limit the candidate road segments to be within a certain distance to the track.
3) Extract matched lines from each matching. This can be done approximately: When there is a "simultaneous walk" on both P and Q, the two segments involved can be considered a pair of matched lines.
4) With all the matched lines, compute the homography as described in Section II using an OpenCV option that employs a robust regression method, either RANSAC (Random Sample Consensus) [23] or Least Median of Squares [24].
5) Using the homography computed, transform each point on the biased track, to obtain its corresponding point on the corrected track.

We emphasize once again that the map matching results serve only as an intermediate step; the corrected tracks are not the matched road segments but the result of a homography transformation. Because of the use of a robust homography finding method that tolerates outliers, the map matching results do not have to be perfect. Figure 6 illustrates the outlier rejection capability of the RANSAC algorithm [25].

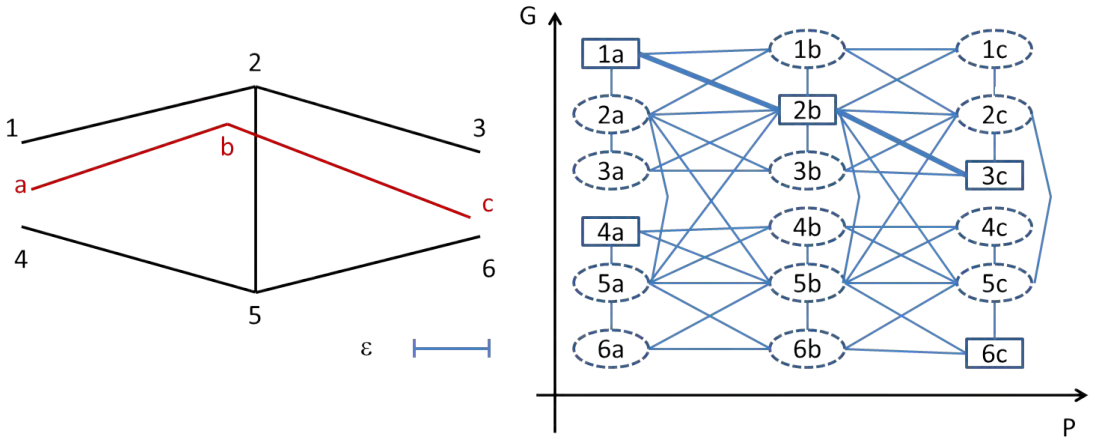

Fig. 5. Left: A map in black and a polygonal curve in red. Nodes for the former are denoted by "1" through "6," while for the latter, "a" through "c." Right: The free space graph constructed for a given distance ϵ where free nodes are denoted by a solid rectangle.

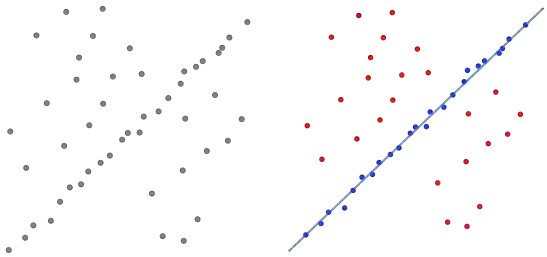

Fig. 6. From [25]. Left: A data set with a line and outliers; Right: The fitted line using RANSAC.

V. Simulations

To illustrate the efficacy of the proposed approach, we performed studies using simulated data.

We downloaded road network data from a government website [26] in the form of shape files. A shape file lists each road as a series of points. To extract topological structure from such data, we first determine all the points at intersections. Then for each unique point in the road network, we list its adjacent points on roads (and ignore the constraints of one-way roads in this study). This yields a graph such that we can apply the map matching algorithm. In our map structure we also include a k-d tree (see [27] and [28]) so that when a track is matched against the map, only the portion of the map that is close enough to the track is used.

To perform Monte Carlo simulations, we adopt the following method of generating tracks for each simulation. We start from a random point in the map and walk to one of its neighboring points that have not been visited yet. Repeat this until we have a track with a given number of points in it. Repeat to get the desired number of tracks.

To simulate the projective bias, we transform truth tracks as follows: We pick a point in the map to be the center. We rotate the tracks by 0.5 degree, expand them by 2 percent, and translate them by 5 meters to the east and 5 meters to the north. One set of simulated data is shown in Figure 1, where truth tracks are in green and biased tracks are in red.

We then perform map matching for each track using our proposed algorithm. After we get a matching such as "1a $\rightarrow$ 2b $\rightarrow$ 3c" as illustrated in Figure 5, we declare that the track segment "a $\rightarrow$ b" matches the road segment "1 $\rightarrow$ 2" and that the track segment "b $\rightarrow$ c" matches the road segment "2 $\rightarrow$ 3". Such correspondence is not always correct, but is correct most of the time, which gives us enough good data to calculate the underlying homography. This calculation is carried out by using a MATLAB interface to OpenCV called mexopencv [29], with the Least Median of Squares method.

After we find the homography, we apply it to the biased tracks and obtain corrected tracks, as shown in Figure 7 on Page 7, where corrected tracks are in red, and truth tracks are in green but are not visible under the red tracks.

The bias removal shown in this simulation is almost perfect because the relationship between the truth and biased tracks is indeed characterized by a homography, which we have successfully found through automated matching and robust fitting. In practice, there can be complications such as the following:

- The tracks may be noisy and may not be perfectly on the road even without any projection bias.
- The spacing of points on a road network may be too large compared to the spacing of points on a track, resulting in matchings that do not reflect the true relationship.
- The bias is large and the automated matching yields too many outliers.
- There may be other types of biases present.

Further studies are needed to assess the effects of these factors on automated bias removal.

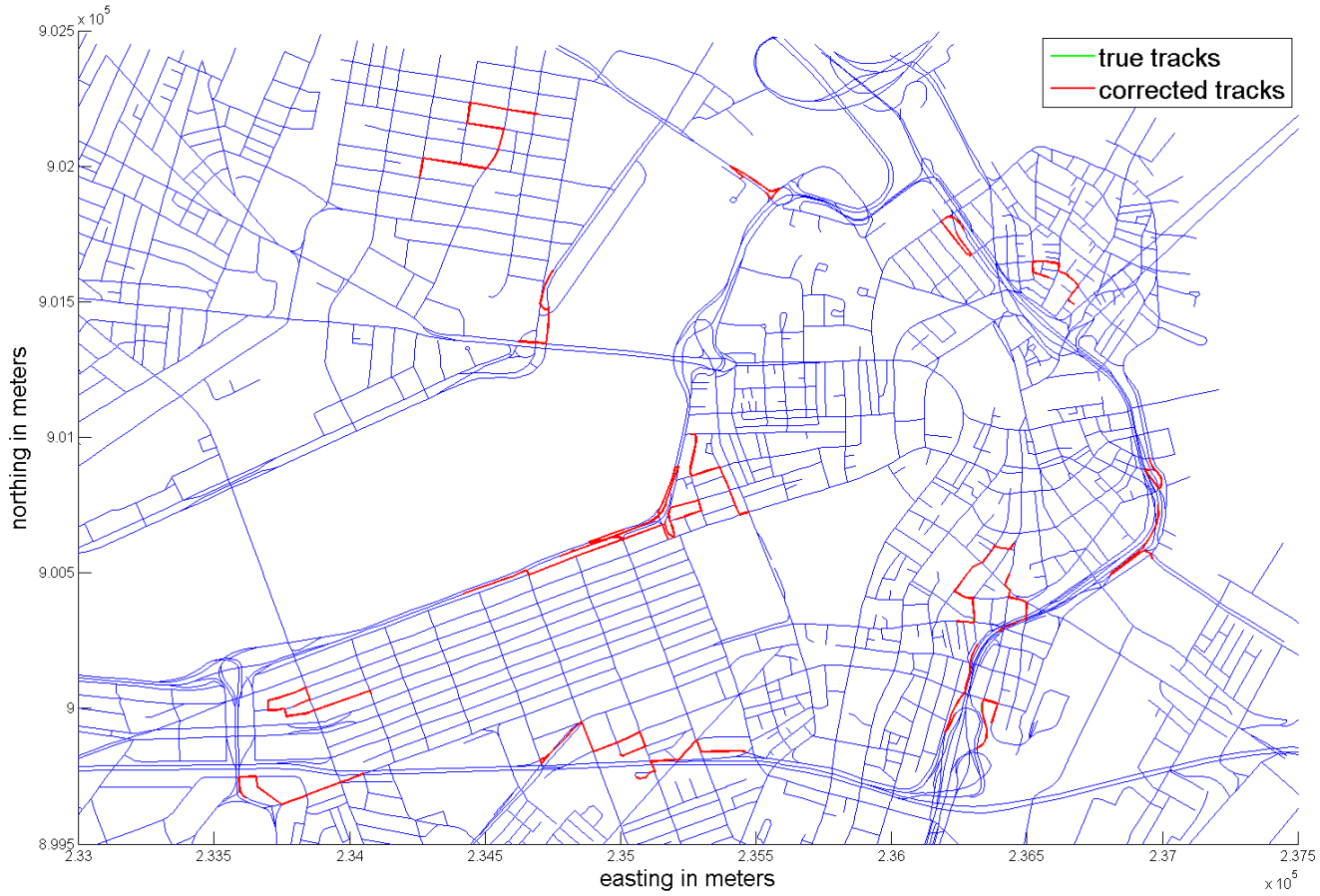

Fig. 7. Tracks with bias removed. Truth tracks in green are overwritten by corrected tracks in red.

VI. Conclusions and Future Work

A projection bias is often present when video tracks are projected to the surface of the Earth. To remove such a bias we need to determine the corresponding homography, which is conventionally calculated through matched points or line segments. To automate this process we have presented a method to match track segments to road segments based on the discrete Frechet distance. Simulations show that the method works well for a small projection bias.

The effect of the complicating factors listed in the previous section should be studied in future work. In particular, when the bias is large, Frechet distance alone, without considering shapes and other topological features, may not produce enough correct matchings to determine the homography accurately. One possible way to alleviate the problem is an iterative approach: After we obtain a matching between a set of tracks $\mathcal{T}_0$ and road segments $\mathcal{L}_0$, instead of calculating the homography that moves the tracks to align with the roads, we could conceivably adjust the starting homography H_0 along a direction that can reduce the discrepancy between the two, and apply the newly found H_1 to the tracks, to obtain a new set of (intermediate) tracks $\mathcal{T}_1$. Then we match this set of tracks to the roads to obtain a new set of road segments $\mathcal{L}_1$, and hopefully more matching results will be correct this time than last time. Then we further adjust H_1 to arrive at H_2 and so on and so forth, until the discrepancy cannot be reduced further.

Human eyes can perform the matching better than machines, but with data deluge and operator overload, automated methods have an important role to play.

Acknowledgment

The authors would like to thank Dr. Pablo Arambel for initiating the bias removal work and for many helpful technical discussions, and Drs. Chris Moss, Joao Cabrera and Lou Galup for their valuable feedback.

References

[1] H. Alt, A. Efrat, G. Rote, and C. Wenk, "Matching planar maps," *J. Algorithms*, vol. 49, no. 2, pp. 262–283, Nov. 2003.

[2] T. Eiter and H. Mannila, "Computing Discrete Frechet Distance," Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, Tech. Rep., 1994, technical Report CD-TR 94/64.

[3] X. Lin, Y. Bar-Shalom, and T. Kirubarajan, "Multisensor multitarget bias estimation for general asynchronous sensors," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 41, no. 3, pp. 899–921, Jul. 2005.

[4] J. P. Ferry, "Exact bias removal for the track-to-track association problem," *Information Fusion, 2009 12th International Conference on*. IEEE, Jul. 2009, pp. 1642–1649.

[5] D. J. Papageorgiou and J. D. Sergi, "Simultaneous Track-to-Track Association and Bias Removal Using Multistart Local Search," *Aerospace Conference, 2008 IEEE*. IEEE, Mar. 2008, pp. 1–14.

[6] C. Yang, E. P. Blasch, J. A. Patrick, and D. Qiu, "Ground target track bias estimation using opportunistic road information," *Aerospace and Electronics Conference (NAECON), Proceedings of the IEEE 2010 National*. IEEE, Jul. 2010, pp. 156–163.

[7] M. Ulmke and W. Koch, "Road-map assisted ground moving target tracking," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 42, no. 4, pp. 1264–1274, Oct. 2006.

[8] H. Zhu, S. Chen, C. Han, and Y. Lin, "Fusion of possible biased local estimates in sensor network based on sensor selection," *Information Fusion, 2013 16th International Conference on*. IEEE, Jul. 2013, pp. 357–364.

[9] D. Belfadel, R. W. Osborne, and Y. Bar-Shalom, "Bias estimation for optical sensor measurements with targets of opportunity," *Information Fusion (FUSION), 2013 16th International Conference on*. IEEE, Jul. 2013, pp. 1805–1812.

[10] E. Taghavi, R. Tharmarasa, T. Kirubarajan, and Y. Bar-Shalom, "Bias estimation for practical distributed multiradar-multitarget tracking systems," *Information Fusion, 2013 16th International Conference on*. IEEE, Jul. 2013, pp. 1304–1311.

[11] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," *Proceedings of the 31st international conference on Very large data bases*, ser. VLDB '05. VLDB Endowment, 2005, pp. 853–864.

[12] C. Wenk, R. Salas, and D. Pfoser, "Addressing the Need for Map-Matching Speed: Localizing Global Curve-Matching Algorithms," *Scientific and Statistical Database Management, 2006. 18th International Conference on*. IEEE, 2006, pp. 379–388.

[13] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 5, pp. 312–328, 2007.

[14] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk, "Approximate Map Matching with respect to the Frechet Distance," *ALENEX*, 2011, pp. 75–83.

[15] T. Hunter, P. Abbeel, and A. Bayen, "The Path Inference Filter: Model-Based Low-Latency Map Matching of Probe Vehicle Data," *Algorithmic Foundations of Robotics X*, ser. Springer Tracts in Advanced Robotics, E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, Eds. Springer Berlin Heidelberg, 2013, vol. 86, pp. 591–607.

[16] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, Apr. 2004.

[17] OpenCV User Site, http://opencv.org/, 2014.

[18] Wikipedia, "Frechet distance," http://en.wikipedia.org/wiki/Fr%C3%A9chet_distance, 2013.

[19] H. Alt and M. Godau, "Computing the Frechet distance between two polygonal curves," *International Journal of Computational Geometry & Applications*, vol. 5, no. 01n02, pp. 75–91, 1995.

[20] Z. Danziger, "Discrete Frechet Distance," http://www.mathworks.com/matlabcentral/fileexchange/31922-discrete-frechet-distance, 2013.

[21] B. Aronov, S. H. Peled, C. Knauer, Y. Wang, and C. Wenk, "Frechet distance for curves, revisited," *Proceedings of the 14th conference on Annual European Symposium - Volume 14*, ser. ESA'06. London, UK, UK: Springer-Verlag, 2006, pp. 52–63.

[22] R. van Oostrum and R. C. Veltkamp, "Parametric search made practical," *Computational Geometry*, vol. 28, no. 2-3, pp. 75–88, Jun. 2004.

[23] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[24] P. J. Rousseeuw, "Least Median of Squares Regression," *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 871–880, Dec. 1984.

[25] Wikipedia, "RANSAC," http://en.wikipedia.org/wiki/RANSAC, 2014.

[26] U.S. Census Bureau, "Maps & Data," http://www.census.gov/geo/maps-data/, 2014.

[27] Wikipedia, "k-d tree," http://en.wikipedia.org/wiki/K-d_tree, 2014.

[28] A. Tagliasacchi, "kd-tree for Matlab," http://www.mathworks.com/matlabcentral/fileexchange/21512-kd-tree-for-matlab, 2010.

[29] K. Yamaguchi, "mexopencv - matlab mex functions for opencv," http://www.cs.sunysb.edu/~kyamagu/mexopencv, 2013.