Dynamics-Aware Similarity of Moving Objects Trajectories

Goce Trajcevski[*] Hui Ding Peter Scheuermann[†]

Department of EECS Northwestern University
goce,hdi117,peters@eecs.northwestern.edu

Roberto Tamassia[‡]
Department of CS
Brown University
rt@cs.brown.edu

Dennis Vaccaro
Northrop Grumman Corp.
Defense Systems Division
dennis.vaccaro@ngc.com

ABSTRACT

This work addresses the problem of obtaining the degree of similarity between trajectories of moving objects. Typically, a Moving Objects Database (MOD) contains sequences of (location,time) points describing the motion of individual objects, however, they also have an implicit information about the velocity, which is an important attribute describing the dynamics of a particular object. Our main goal is to extend the MOD functionalities with the capability of reasoning about how similar are the trajectories of objects that, possibly, move along geographically different routes. Towards this, we use a distance function which balances the lack of temporal-awareness of the Hausdorff distance with the generality (and complexity of calculation) of the Fréchet distance. Based on the observation that, as a first-approximation in practice, the individual segments of trajectories are assumed to have constant speed, we provide efficient algorithms for: (1) optimal matching between trajectories; and (2) approximate matching between trajectories, both under translations and rotations, where the approximate algorithm guarantees a bounded error-quality with respect to the optimal one.

1. INTRODUCTION AND MOTIVATION

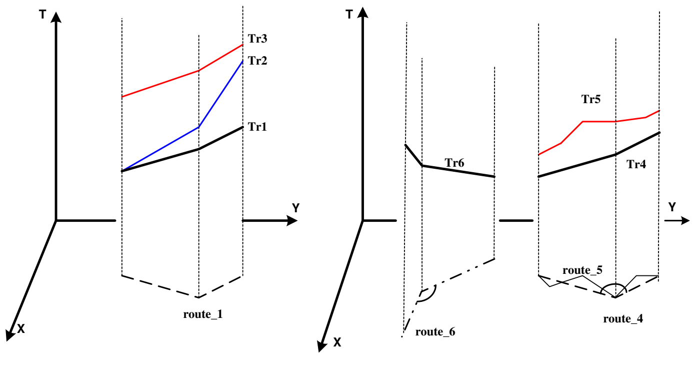
The management of the transient location-in-time information of mobile entities, is a fundamental ingredient for a variety of applications that involve Location Based Service (LBS) [25], and is a topic of the field known as Moving Object Databases (MOD) [19]. This work attempts to increase the MOD capabilities with providing algorithmic solutions that can be used to answer queries such as:

– **Q1:** *"Retrieve all the troops which had similar motion patterns in the geographic regions R1 and R2 in the last two months."*

– **Q2:** *Retrieve all the trajectories that had similar motion patterns while looking for parking in the weekends around the shopping malls in the Mid-West."*

– **Q3:** *"Retrieve all the Hurricanes with similar trajectories that occurred in any US state along the Mexican Gulf in the last 5 years."*

The commonality of all these queries is that the motion of interest need not occur in the same geo-locations (e.g., the hurricanes may occur in Texas and Florida) and, even more importantly, need not follow similar routes in terms of the geo-coordinates (e.g., a hurricane moving North and then changing its direction to North-West may be very similar to the one that has moved East and changed its direction to North-East).



a.) Dynamics over same route b.) Dynamics over different routes

Figure 1: Similarities of Motions

The main idea behind the *similarity of trajectories* that we tackle in this work is the ability to compare the *relative motion* of the moving objects along the consecutive segments of their corresponding trajectories. An illustration of our desiderata is provided in Figure 1. Namely, Figure 1.a) il-

[*]Research supported by the Northrop Grumman Corp., contract: P.O.8200082518

[†]Research supported by the NSF grant IIS-0325144/003

[‡]Research supported in part by NSF grant IIS-0324846

lustrates three trajectories Tr_1, Tr_2 and Tr_3, corresponding to the motion plans of three objects, say, o_1, o_2 and o_3, all of which were moving along a *same* 2D route (illustrated as *route_1*). However, although, o_1 and o_2 begin their trips at a same time, their respective trajectories are very different – namely, o_2 has been moving much slower than o_1. Throughout this work, we would like to perceive such trajectories as *not similar*. On the other hand, although o_3 had started its trip later than o_1 (and o_2), its trajectory, Tr_3 exhibits the same dynamics (velocity patterns over the corresponding route segments) as Tr_1. Clearly, the only difference is that Tr_3 is *translated* in the temporal domain with respect to Tr_1 and we would like to consider them *similar*. Figure 1.b) illustrates some other discrepancies among trajectories that we would like to consider *acceptable* for the purpose of similarity. Namely, although the route of Tr_5 does not exactly match the route of Tr_4 – e.g., several small detours were taken by the respective object o_5, the duration of both trips is approximately equal *and* the spatial deviations are not too large (plus we do need a translation in the temporal dimension). A careful observation of Tr_6 reveals that although it has a completely different 2D route from Tr_1, its corresponding segments are of equal length and are having equal corresponding angles between themselves. Moreover, the speed of the moving object, say o_6, along the route-segments of Tr_6 is exactly the same as the one of the corresponding object, say o_4, along the segments of Tr_4.

One of the main contributions of this work is precisely the introduction of the concept of *dynamical similarity of trajectories* under the *rigid motion* transformations, that is, translations and rotations [5, 17]. Translation is allowed both in the spatial and temporal dimension, while rotation is allowed only in the 2D – spatial dimension. We use an intuitive similarity-distance function that enables us to capture the relative dynamics of the trajectories, which can be qualified as a special case of the Fréchet distance [5, 6] for polygonal curves, and does not suffer from the time-unawareness of the Hausdorff distance [5, 11]. We present efficient algorithmic solutions to two aspects of the problem of similarity between trajectories: (1.) *optimization:* finding the rigid transformation (rotation/translation in the spatial domain, and translation in the temporal domain) which minimizes the similarity-distance between two given trajectories, and (2.) *approximation:* determining an *approximate* matching between two trajectories using a faster-than-optimal algorithm, with a guarantee that its output is within a fixed error-bound from the optimal similarity-distance. Additionally, our algorithms can also be used for determining whether one of the trajectories is similar to a portion of the other.

The rest of this paper is structured as follows. In Section 2 we recollect the necessary background. Section 3 presents our main results – the algorithms for determining the optimal and approximate similarity matching between trajectories. In Section 4 we present experimental observations which quantify the benefits of our proposed methods. Section 5 positions our work with respect to the related literature and Section 6 concludes the paper and presents directions for future work.

2. PRELIMINARY BACKGROUND

We now define the concepts that will be used in the rest of this paper. Firstly, we present a definition of a *trajectory*, which is often used in the MOD literature [19, 27, 28] to describe the motion of the moving objects, assuming that the objects are moving in a 2D space with respect to a given coordinate system:

DEFINITION 1. *A* trajectory *Tr is a function $F_t : T \rightarrow \mathcal{R}^2$ which maps a given (temporal) interval $[t_b, t_e]$ into a one-dimensional subset of $\mathcal{R}^2$. It is represented as a sequence of 3D points (2D geography + time) (x_1, y_1, t_1), (x_2, y_2, t_2), $\ldots$, (x_n, y_n, t_n), where $t_b = t_1$ and $t_e = t_n$ and $t_1 \leq t_2 \leq \ldots \leq t_n$.*

Each point (x_i, y_i, t_i) in the sequence represents the 2D location (x_i, y_i) of the object, at the time t_i. For every $t \in (t_i, t_{i+1})$, the *location* of the object is obtained by a *linear interpolation* between (x_i, y_i) and (x_{i+1}, y_{i+1}) with the ratio $(t - t_i)/(t_{i+1} - t_i)$, which is, in between two points the object is assumed to move along a straight line-segment and with a constant speed. The 2D projection of Tr is a polygonal chain with vertices (x_1, y_1), $(x_2, y_2) \ldots (x_n, y_n)$ and it is called a *route* of Tr.
Observe that a trajectory may represent both the *past* and the *future* motion, i.e. the *motion plan* of a given object (c.f. [19]). Typically, for future trajectories, the user provides the starting location, starting time and the destination (plus, possibly, a set of to-be-visited) points, and the MOD server uses these, along with the distribution of the speed-patterns on the road segments as inputs to a dynamic extension of the Dijkstra's algorithm [13, 27], to generate the shortest travel-time trajectory. As an example of the usage of this model, aside from the commercial-fleet vehicles (e.g., FedEx), the monthly requests for shortest travel-time individual future-trajectories posed to MapQuest is 46.4 million, Yahoo!Maps – 20 million, and Google Maps – 19.1 million [23].

We also need to recollect some definitions from the Computational Geometry (CG) literature [5, 6] regarding the distance(s) between curves. In what follows, we consider 2D planar curves, given as continuous mappings $C_i : [a_i, b_i] \rightarrow \mathcal{R}^2$ (i.e., via parametrization), where $a_i < b_i (\in \mathcal{R})$. Naturally, for our purposes, the domain of C_i will correspond to the temporal dimension.

DEFINITION 2. *Let $C_1 : [a_1, b_1] \rightarrow \mathcal{R}^2$ and $C_2 : [a_2, b_2] \rightarrow \mathcal{R}^2$ denote two curves. Let $\| \cdot \|$ denote the L_2 norm and let P_1 (resp. P_2) denote a point on C_1 (resp. C_2). The Hausdorff distance from between C_1 and C_2 is defined as:*

$$\delta_H(C_1, C_2) := \max(\tilde{\delta}_H(C_1, C_2), \tilde{\delta}_H(C_2, C_1)), \quad \textit{where}$$

$$\tilde{\delta}_H(C_1, C_2) = \sup_{P_1 \in C_1} \inf_{P_2 \in C_2} \|P_1 - P_2\|$$

$$(\textit{resp.} \quad \tilde{\delta}_H(C_2, C_1) = \sup_{P_2 \in C_2} \inf_{P_1 \in C_1} \|P_1 - P_2\|)$$

and $\tilde{\delta}_H(C_1, C_2)$ (resp. $\tilde{\delta}_H(C_2, C_1)$) denotes the directed Hausdorff distance from C_1 to C_2 (resp. from C_2 to C_1).

The Hausdorff distance between two curves simply looks for the smallest ε such that C_1 is completely contained in the ε-neighborhood of C_2 (i.e., C_1 is completely contained in the Minkowski Sum [12] of C_2 and a disk with radius ε) and vice versa. Although it is arguably a very natural distance measure between curves and/or compact sets, the Hausdorff distance is too "static", in the sense that it neither considers any direction nor any dynamics of the motion along the curves. A classical example of the inadequacy of the Hausdorff distance, often used in the CG literature [5, 6, 17] is the *"man walking the dog"*. Figure 2 illustrates the corresponding routes of the man *(M-route)* and the dog *(D-route)*, as

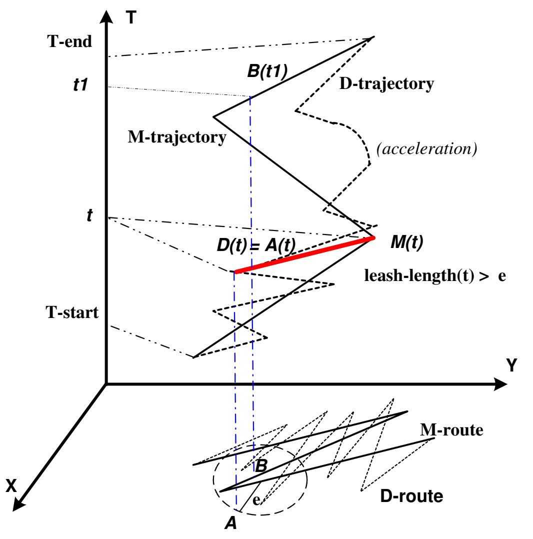


Figure 2: Hausdorff vs. Fréchet distance

well as their trajectories *M-trajectory* and *D-trajectory*. Observe that, ignoring the temporal aspect of their motions, the D-route and the M-route are within Hausdorff distance of e, as exemplified by the points A and B in the $X-Y$ plane. However, their actual (temporally-aware) distance corresponds to the minimal length of the leash that the man needs to hold. The 3D part of Figure 2 illustrates the discrepancy between the distances among the points along the *routes*, and their corresponding counterparts along *trajectories*: when the dog is at the point A, which is at time t, the man is actually at $M(t)$, and their distance is much greater then e (the man is at the geo-location B at the time $t_1 > t$). Now we have the following:

Definition 3. *Let $C_1 : [a_1, b_1] \rightarrow \mathcal{R}^2$ and $C_2 : [a_2, b_2] \rightarrow \mathcal{R}^2$ denote two curves, and let $\|\cdot\|$ denote the L_2 norm. The Fréchet distance of C_1 and C_2, denoted by $\delta_F(C_1, C_2)$ is defined as:* $\delta_F(C_1, C_2) := \inf_{\alpha,\beta} \max_{t\in[0,1]} \|C_1(\alpha(t)) - C_2(\beta(t))\|$ *where α (resp. β) ranges over all the possible continuous and monotonically increasing functions $[0, 1] \rightarrow [a_1, b_1]$ (resp. $[0, 1] \rightarrow [a_2, b_2]$).*

The Fréchet distance is more general than the Hausdorff one, in the sense that it allows for a variety of possible motion-patterns along the given route-segments, as formalized by the functions α and β in Definition 3. As an illustration, observe that on the portion of the *D-trajectory*, the dog may be moving non-uniformly (i.e., accelerating) along a route segment. Recall, however, that this is precluded in the Definition 1 for trajectories in MOD settings. It can be verified (c.f. [2]) that for any two curves C_1 and C_2 it holds that $\delta_H(C_1, C_2) \leq \delta_F(C_1, C_2)$. Observe that in Definition 3, the requirement is that both α and β are increasing functions. If this is relaxed, then the distance function is called *weak Fréchet* distance, denoted by $\tilde{\delta}_F(C_1, C_2)$, which intuitively corresponds to the ability of both the man and the dog moving backwards along the routes and: $\delta_H(C_1, C_2) \leq \tilde{\delta}_F(C_1, C_2) \leq \delta_F(C_1, C_2)$ [2].

3. DYNAMICS-AWARE SIMILARITY

Our main goal is to obtain and efficiently calculate a distance-measure which captures a *relativized space+time* difference of two given trajectories, for the purpose of measuring the similarity of their corresponding motions. Firstly, we present the necessary setup and introduce the distance function used in this work. Subsequently, we focus on the case where the trajectories have equal temporal durations and we analyze in details the optimization of matching two trajectories under translations and rotations in the spatial domain. We address separately the problem of optimal and approximate matchings. Lastly, we demonstrate how our techniques can be used for comparing the similarity-based containment of trajectories with different temporal-durations. Due to space limitations, we only give brief justification of our claims instead of formal proofs.

3.1 Similarity Distance of Trajectories

Let $Tr_1 = [(x_{11}, y_{11}, t_{11}), (x_{12}, y_{12}, t_{12}), \ldots, (x_{1n}, y_{1n}, t_{1n})]$ and $Tr_2 = [(x_{21}, y_{21}, t_{21}), (x_{22}, y_{22}, t_{22}), \ldots (x_{2m}, y_{2m}, t_{2m})]$ denote two trajectories with n and m segments, respectively. When measuring the distance between the two objects whose motions are represented by the Tr_1 and Tr_2, we would like to know its values *at given time-points*. The intuitive reason is that one cannot say "the distance between o_1 and o_2 is d, when o_1 is at location $L_1(x_1, y_1)$ and o_2 is at location $L_2(x_2, y_2)$, *except*, o_1 was at L_1 at time t_1, o_2 was at L_2 at time t_2, and t_1 and t_2 are far apart". This observation was actually used as a basis for introducing the, so called, *Eucledean time-Uniform* distance function in [11], a variant of which is given by the following:

Definition 4. *Given two trajectories Tr_1 and Tr_2, the* Eucledean time-Uniform distance function *$E_{ud} : (t, Tr_1, Tr_2) \rightarrow \mathcal{R}$ is defined as follows: $E_{ud}(t, Tr_1, Tr_2) = \|P(t) - Q(t)\|$ when $t \in ([t_{11}, t_{1n}] \cap [t_{21}, t_{2m}])$ and it is* undefined *otherwise. $P(t)$ (resp. $Q(t)$) is the 2D point corresponding to the spatial locations on the trajectory Tr_1 (resp. Tr_2) at time t.*

The Eucledean time-Uniform Distance *of two trajectories Tr_1 and Tr_2 is defined as*

$$TE_{ud}(Tr_1, Tr_2) = \max_t E_{ud}(t, Tr_1, Tr_2)$$

In other words, the TE_{ud} distance between trajectories maximizes the value of $\sqrt{(P.x - Q.x)^2 + (P.y - Q.y)^2}$, taken at same time-instances over their common temporal interval.

When two trajectories correspond to trips of *equal temporal duration*, $(t_{1n} - t_{11} = t_{2m} - t_{21})$, we can always "transform" one of them by translating it in the temporal domain. Hence, without loss of generality, we will assume in what follows that the two trajectories Tr_1 and Tr_2 pertain to same temporal intervals, i.e., $t_{11} = t_{21}$ and $t_{1n} = t_{2n}$. For the purpose of calculating $TE_{ud}(Tr_1, Tr_2)$ we rely on the following:

Lemma 1. *Given two trajectory segments over the same temporal interval: $[P_1P_2] = [(x_{11}, y_{11}, t_1), (x_{12}, y_{12}, t_2)]$ and $[Q_1Q_2] = [(x_{21}, y_{21}, t_1), (x_{22}, y_{22}, t_2)]$, the maximal value of $E_{ud}(t, Tr_1, Tr_2)$ is obtained at one of the corresponding endpoints, i.e., either $TE_{ud}(Tr_1, Tr_2) = \|\overline{P_1Q_1}\|$ or $TE_{ud}(Tr_1, TR_2) = \|\overline{P_2Q_2}\|$.*

The proof is based on the analysis of $\frac{dE_{ud}(t)}{dt}$ and $\frac{d^2E_{ud}(t)}{dt^2}$.

Recalling Definition 3 (Section 2), one may observe that the TE_{ud} distance function is actually a special case of the Fréchet distance when α and β are restricted to the natural (linear) mapping $[0,1] \rightarrow [t_{11}, t_{1n}]$. Now, one of the main questions of this work can be formulated as follows: when two trajectories Tr_1 and Tr_2 are ranging over the same temporal interval, what is the minimal value of $TE_{ud}(Tr_1, Tr_2')$, where TR_2' is obtained by applying only *translation* and *rotation* to Tr_2? Formally:

DEFINITION 5. *Given two trajectories Tr_1 and Tr_2, their* Rigid Transformations Similarity Distance $RTSD(Tr_1, Tr_2)$ *is defined as $RTSD(Tr_1, Tr_2) = (TE_{ud}(Tr_1, \Psi \circ \Phi(Tr_2)))$, where $\Psi \circ \Phi$ is a composition of translations (Ψ) and rotations (Φ) in a horizontal plane, applied to the route of Tr_2.*

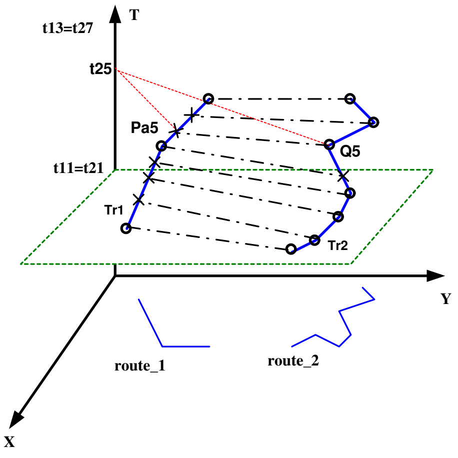


Figure 3: Artificial Vertices of Trajectories

3.1.1 Optimal Similarity Matching

In order to calculate the optimal $RTSD(Tr_1, Tr_2)$ we need to determine the particular Ψ_{opt} and Φ_{opt} that minimize the $TE_{ud}(Tr_1, \Psi_{opt} \circ \Phi_{opt}(Tr_2))$, for which we proceed as follows. First, in each of Tr_1 and Tr_2 we introduce extra *artificial vertices*, so that they both have the same number of points. Essentially, if a given vertex of Tr_1, say $(x_{1k_i}, y_{1k_i}, t_{1k_i})$ does not have a matching vertex on Tr_2 at the *same time-instance* t_{1k_i}, we introduce an artificial one $(x^a_{2k_i}, y^a_{2,k_i}, t_{1k_i})$, and similarly for the vertices of Tr_2. This is illustrated in Figure 3 where, originally, Tr_1 has three vertices and Tr_2 has seven, indicated with "$\circ$", however, after introducing the artificial ones, indicated with "x", each of them has eight vertices. For instance, Q_5 at t_{25} is a genuine vertex of Tr_2 and its corresponding artificial vertex on Tr_1 is P_{a5} at that time. According to Definition 1, the spatial coordinates of each artificial vertex $(x^a_{k_i}, y^a_{k_i})$ are obtained by a linear interpolation between the two original consecutive vertices (x_i, y_i, t_i) and $(x_{(i+1)}, y_{(i+1)}, t_{(i+1)})$, for which $t_{k_i} \in (t_i, t_{(i+1)})$. Hence, we obtain two *augmented trajectories* Tr^a_1 and Tr^a_2 with exactly the same number of points ($O(m+n)$), pairwise matching in the temporal dimension. Now, as a generalization of Lemma 1, we know that $RTSD(Tr_1, Tr_2)$ will be obtained at one of the temporally-corresponding pairs of vertices in Tr^a_1 and Tr^a_2. As a consequence, the problem of calculating $RTSD(Tr_1, Tr_2)$ is now reduced to calculating the corresponding L_2 distances between two point-sets corresponding to the (2D projections of the augmented trajectories) vertices along the routes. Moreover, if we view the two sets of 2D points $P = \{p_1, p_2, \ldots p_u\}$ and $Q = \{q_1, q_2, \ldots, q_u\}$, we have the restriction that the mapping $g : P \rightarrow Q$ is *fixed*, in the sense that $g(p_j) = q_j (j \in \{1, 2, \ldots u\})$ where $u = O(m+n)$. Now we are interested in a composition $\Psi \circ \Phi$ such that the largest distance between p_j and $\Psi \circ \Phi(q_j)$ is minimized over all j.

However, this particular problem, the optimal matching of two point-sets with a fixed mapping between them, was addressed in [22]. To explain the solution, observe firstly that applying a rotation ρ around the coordinate-center with an angle θ to the point $q_j(x_{qj}, y_{qj})$ will transform it to: $\rho_\theta(q_j) = (x_{qj} \cos\theta - y_{qj} \sin\theta, x_{qj} \sin\theta + y_{qj} \cos\theta)$. Subsequently, translating it by a vector $\vec{\tau} = (x_\tau, y_\tau)$, brings it to the point $\tau(\rho_\theta(q_j)) = ((x_{qj} \cos\theta - y_{qj} \sin\theta) + x_\tau, (x_{qj} \sin\theta + y_{qj} \cos\theta) + y_\tau)$ (c.f. [22]). Hence, our original problem of obtaining the $RTSD(Tr_1, Tr_2)$ is reduced to finding θ and $\vec{\tau}$ such that $\max_j \|p_j - \tau(\rho_\theta(q_j))\|)$ is minimized. Equivalently, we are seeking: $\min_{\theta,\tau} \max_j f_j(\theta, x_\tau, y_\tau) = \min_{\theta,\tau} \max_j$ $[(x_{pj} - ((x_{qj} \cos\theta - y_{qj} \sin\theta) + x_\tau))^2 + (y_{pj} - ((x_{qj} \sin\theta + y_{qj} \cos\theta) + y_\tau))^2]$.

For a fixed value of θ, say θ', we are essentially looking at $\min_{x_\tau, y_\tau} \max_j f_j(\theta', x_\tau, y_\tau)$, and the problem is now reduced to the one of minimum enclosing circle [24]. As θ "moves" between 0 and 2π, the problem of calculating the $RTSD(Tr_1, Tr_2)$ becomes the one of finding the minimum enclosing circle for "moving points" $\rho_\theta(q_j)$, subject to having a pair of points $\rho_\theta(q_s)$ and $\rho_\theta(q_l)$ on its boundary. The problem of the minimum enclosing circle can be reduced to finding the minimum value on the envelope of a family of functions $f_j(\theta', x_\tau, y_\tau)$. Its relationship with the *dynamic furthest Voronoi diagram* [8] of the points $(x_{qj} \cos\theta - y_{qj} \sin\theta) + x_\tau, (x_{qj} \sin\theta + y_{qj} \cos\theta) + y_\tau$ is presented in details in [22]. Essentially, the algorithm fixes a pair of indices s and l and solves the constrained problem of finding the θ which yields the minimum enclosing circle having $\rho_\theta(q_j)$, subject to having a pair of points $\rho_\theta(q_s)$ and $\rho_\theta(q_l)$ on its boundary. For a fixed pair l and s (c.f. Lemma 4.1 in [22]) the complexity of obtaining the maximum diagram of u functions of the kind $f_j(\theta', x_\tau, y_\tau)$ $(1 \leq j \leq u, s \neq j \neq l)$ is $O(\lambda_7(u) \log u)$. Repeating the search for the minimum enclosing circle for all the possible pairs s and l on its boundary (and picking the smallest one among them) yields a complexity of $O(u^2 \lambda_7(u) \log u)$. In the preceding, the $\lambda_7(u)$ denotes the maximum length of the $(u, 7)$ Davenport-Schinzel (DS) sequence, where the length of any (u, a) DS sequence for $a > 3 (a \leq u)$ is $\lambda_a(u) = u\alpha(u)^{O(\alpha(u)^{a-3})}$ (c.f. [26], where even sharper bounds are presented).$\alpha(u)$ denotes the extremely slow-growing inverse Ackermann function[1]. Hence, as a direct consequence of Theorem 4.1. in [22], the problem of calculating the optimal $RTSD(Tr_1, Tr_2)$ can be solved in $O((m+n)^2 \lambda_7(m+n) \log(m+n))$,
where $\lambda_7(m+n) = (m+n)\alpha(m+n)^{O(\alpha(m+n)^4)}$.

3.1.2 Approximate Similarity of Trajectories

Often, instead of actually finding the optimal matching (minimizing $RTSD$) between two trajectories, one is interested in an approximate matching, which can be obtained by a faster algorithm. However, it is desirable to have a bound on the error of the obtained approximation, in

[1]We note that $\lambda_1(n) = n$; $\lambda_2(n) = 2n - 1$; and $\lambda_3(n) = O(n \log n)$ [26].

the following sense. Let $\varepsilon_{opt} = RTSD(Tr_1, Tr_2)$, denote the distance obtained by finding the optimal transformation $\Psi_{opt} \circ \Phi_{opt}(Tr_2)$. For a given *approximate transformation* $\Psi_A \circ \Phi_A(Tr_2)$ – again, a composition of translation and rotation applied to Tr_2, we say that it has a *loss factor* d if $\varepsilon_A = E_{ud}(Tr_1, \Psi_A \circ \Phi_A(Tr_2))$ satisfies the inequality $\varepsilon_A \le d \cdot \varepsilon_{opt}$ [17]. Equivalently, an approximation with error bound ε_A is said to have a quality of "c", if it is within the bound of $(1+c)\varepsilon_{opt}$ [2, 6].

Definition 6. *Let $Tr = [(x_1, y_1, t_1), \ldots, (x_k, y_k, t_k)]$ denote an arbitrary (actual or augmented) trajectory with k points. The backbone of Tr is the line through the points (x_1, y_1, t_1) and (x_k, y_k, t_k). The backbone trajectory of Tr, denoted BTr, is defined as follows:*
– BTr begins at (x_1, y_1, t_1) and ends at (x_k, y_k, t_k).
– for each point $P_i = (x_i, y_i, t_i) \in Tr$, BTr has an anchor-point $AP_i = (x_i^a, y_i^a, t_i^a)$, given by the intersection of the backbone with the horizontal plane through P_i, i.e., $t_i^a = t_i$ and (x_i^a, y_i^a) is obtained by a linear interpolation between (x_1, y_1) and (x_k, y_k) at t_i. We call the segment $\overline{AP_iP_i}$ an anchor.

Let $Tr_1 = [P_1, \ldots, P_k]$ and $Tr_2 = [Q_1, \ldots, Q_k]$ denote two trajectories with the same number of points and spanning over identical time-intervals, i.e., $t_{P_1} = t_{Q_1}$ and $t_{P_k} = t_{Q_k}$. Let $BTr_1 = [AP_1, AP_2, \ldots, AP_k]$ and $BTr_2 = [AQ_1, \ldots, AQ_k]$ denote their respective backbone trajectories. We have:

Lemma 2. *Let Ψ and Φ denote a translation and rotation. If $TE_{ud}(Tr_1, \Psi \circ \Phi(Tr_2)) \le \varepsilon$, then $TE_{ud}(BTr_1, \Psi \circ \Phi(BTr_2)) \le \varepsilon$.*

Lemma 2 is a consequence of Definition 6 and the fact that the rotation and translation are rigid transformations, i.e., not only that they preserve the ratios, but also the lengths and angles. Since the first and the last anchor points coincide with the first and the last points on the actual trajectories, they are bound to be within distance ε after $\Psi \circ \Phi$ is applied to Tr_2, and the rest of the anchor points are linear combinations of the two end-points.

For a given backbone trajectory $BTr = [AP_1, \ldots, AP_k]$, we identify the direction from AP_1 towards AP_k as a *positive* one which, in turn, defines the positive orientation along the $X-Y$ projection (route) of BTr, denoted by BTr^{X-Y}. With respect to the positive orientation, the 2D projection of a given anchor $\overline{AP_iP_i}^{X-Y}$ can be either on the *left*-hand side (LHS) or to the *right*-hand side (RHS) of the BTr^{X-Y} and, for the special case when an anchor is collinear with BTr^{X-Y}, w.l.o.g we assume that it is on the RHS.

The concepts that we introduced so far are illustrated in Figure 4. Part a.) shows 2D projections of two trajectories which, after augmentation, i.e., introduction of artificial vertices, have 11 points each and, as one extreme case, all the points on each trajectory are on the RHS of their corresponding backbone-trajectories. Part b.), on the other hand, shows the 2D projections of (augmented) trajectories with 9 points each and, as another extreme case, they are completely on the opposite side of their backbone-trajectories. Before we proceed with the description of the algorithm for approximate matching between trajectories, we need the following:

Definition 7. *Given two trajectories, $Tr_1 = [P_1, \ldots, P_k]$ and $Tr_2 = [Q_1, \ldots, Q_k]$, their i-th anchor discrepancy is defined as:*
$\|\overline{AP_iP_i} - \overline{AQ_iQ_i}\|$, if P_i^{X-Y} and Q_i^{X-Y} are on the same side of the corresponding BTr_1^{X-Y} and BTr_2^{X-Y}.
$\|\overline{AP_iP_i} + \overline{AQ_iQ_i}\|$, if P_i and Q_i are on the opposite sides of the corresponding BTr_1 and BTr_2.

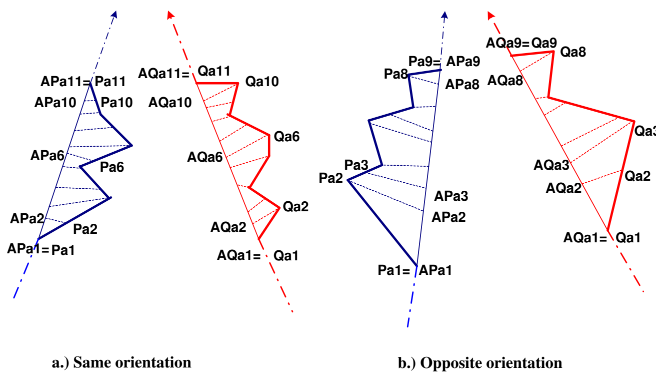


Figure 4: Approximate Matching of Trajectories

As an illustration, in Figure 4a.), the maximum anchor discrepancy is $\overline{AQa_6Qa_6} - \overline{APa_6Pa_6}$. On the other hand, the maximum anchor discrepancy in Figure 4b.) is $\overline{AQa_3Qa_3} + \overline{APa_3Pa_3}$. Actually, the goal of our heuristic (approximate matching algorithm) is to find a transformation, which is a composition of translation Ψ_a that will *contribute to minimizing the anchor discrepancy* between the two trajectories; and a rotation Φ_a that will minimize the rotation-distance between the points on Tr_1 and $\Psi_a(Tr_2)$. Assuming that the input trajectories are already augmented so that they have equal number of points, and that the temporal translation has already been applied so that their time-intervals coincide, we have the following algorithm for approximate similarity matching between two trajectories:

Algorithm MMAD
Input: *two trajectories* Tr_1 and Tr_2
Output: *translation* Ψ_a and *rotation* Φ_a

```
1.  For each pair of anchors $\overline{APa_iPa_i}$ and $\overline{AQa_iQa_i}$
2.  Let Translation(i): $\Psi_i = (((APa_i.x+Pa_i.x)/2-(AQa_i.x+Qa_i.x)/2), ((APa_i.y+Pa_i.y)/2-(AQa_i.x+Qa_i.y)/2))$ //translate $Tr_2^a$ so that the mid-points of the i-th anchors coincide
3.  For each $\Psi_i$
    3.1 Find the corresponding pair of points for which $\|P_j - \Psi_i(Q_j)\|$ is maximal
    3.2 Let rotation angle: $\theta_j^{(i)}$ = the angle which makes the i-th anchor midpoints, $P_j$ and $\Psi_i(Q_j)$ collinear.
4.  Return the pair $(\Psi_i, \Theta_j^{(i)})$ for which
    $TE_{ud}(Tr_1, \Psi_i \circ \Theta_j^{(i)}(Tr_2))$ is minimal
```

Clearly, the time-complexity of the Algorithm MMAD is $O(m+n)^2$, with an additional space requirement of $O(m+n)$. Let $\varepsilon_{MMAD} = TE_{ud}(Tr_1, \Psi_A \circ \Phi_A(Tr_2))$. We have the following:

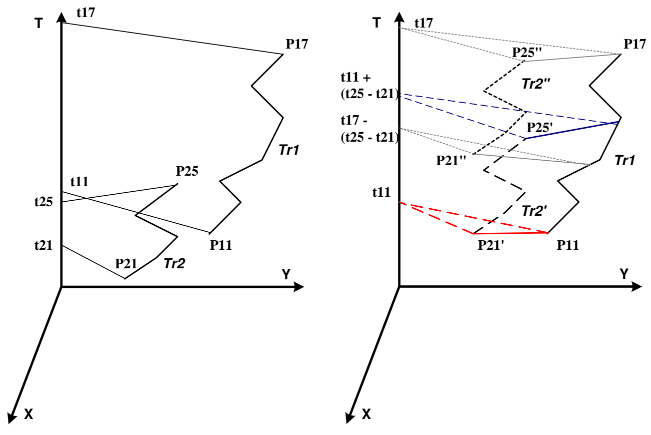
Lemma 3. *Algorithm MMAD generates an approximate matching with the loss-factor of 4, i.e., $\varepsilon_{MMAD} \le 4\varepsilon_{opt}$*

As a consequence of Lemma 2, we have that the midpoints of each corresponding anchors are at the distance $\le \varepsilon_{opt}$ when Tr_2 is transformed by $\Psi_{opt} \circ \Phi_{opt}$. Hence, making the midpoints of the anchors with the maximal discrepancy coincide in the Algorithm MMAD due to Ψ_{MMAD} will perturb the rest of the points with respect to the optimal transformation by an extra $\le \varepsilon_{opt}$. Due to the rotation Φ_{MMAD}

we have another loss of $\leq 2\varepsilon_{opt}$. Considering that the extra "initial" distance $\leq \varepsilon_{opt}$, we obtain the factor of 4.

3.2 Temporal Discrepancy of Trajectories

In the temporal dimension, the main source of a discrepancy between two trajectories is the difference of the duration of the corresponding objects' trips: $\delta_{TD}(Tr_1, Tr_2) = |(t_{1n} - t_{11}) - (t_{2m} - t_{21})|$. Without loss of generality, assume that $(t_{1n} - t_{11}) \geq (t_{2m} - t_{21})$ (i.e., Tr_1 has a longer temporal duration than Tr_2. Now, a variant of the problem of dynamics similarity of the objects' motions can be restated as: *what is the portion of Tr_1 that is most similar to Tr_2?*



a.) Discrepancy between begin times (t11-t21) and end-times (t17-t25)

b.) "Temporal sliding" of trajectories

Figure 5: Temporal Distance Between Trajectories

Figure (5.a) illustrates the discrepancy between the begin-times and the end-times of two trajectories. Again, we observe that the "absolute" measures of the difference of the corresponding begin-points ($|t_{11} - t_{21}|$) is not crucial for detecting their similarities, because a translation in the temporal domain can bring, say, Tr_2 into Tr_2' such that $t_{21}' = t_{11}$. To be able to reason about the similarity between Tr_2 and a *portion* of Tr_1, we need to consider all the possible *translations* in the temporal domain. As an example of two extreme cases:

(1) The begin-times of two trajectories coincide – we shift the begin-point of Tr_2 into t_{11}. Hence, $Tr_2' = [(x_{21}, y_{21}, (t_{21} + (t_{11} - t_{21}))), \ldots, (x_{2m}, y_{2m}, (t_{2m} + (t_{11} - t_{21})))]$;

(2) The end-times of two trajectories coincide – we shift the end-point of Tr_2 into t_{1n}. Hence, $Tr_2" = [(x_{21}, y_{21}, (t_{21} + (t_{1n} - t_{2m}))), \ldots, (x_{2m}, y_{2m}, (t_{2m} + (t_{1n} - t_{2m})))]$. Figure (5.b) illustrates the concept of "possible translations" of Tr_2 ("sliding") in the temporal dimension, and corresponding distances between points (e.g., $\overline{P_{21}' P_{11}}$; $\overline{P_{25}" P_{17}}$, etc.). Let $\tau_{t_v}(Tr_2)$ denote a temporal translation of Tr_2 for (one-dimensional "vector") t_v such that: $t_{21} + t_v \geq t_{11}$ and $t_{2m} + t_v \leq t_{1n}$. For each translation of τ_{t_v} in-between (and including) the two extreme cases, we are interested in the maximal E_{ud} between a point on $\tau_{t_v}(Tr_2)$ and a corresponding point on Tr_1 which has a *same time-value.* Clearly, there are infinitely many values for t_v, however, as a consequence of Lemma 1, we only need to look at the discrete set of temporal translations τ_{t_v} which map a time-value of a vertex from Tr_2 into a time-value of vertex of Tr_1.

Definition 8. *Let Tr_1 and Tr_2 denote two trajectories, where the temporal duration of Tr_2 is $\leq$ than the temporal duration of Tr_1. Their* Temporal-Containment Similarity Distance *$TCSD$ is defined as*

$$TCSD(Tr_1, Tr_2) = \min_{tv} RTSD(Tr_1, \tau_{t_v}(Tr_2))$$

Based on the results in Section 3.1, and observing that due to Lemma 1 there is a total of $O(mn)$ temporal translations (t_v) of interest, we have that:

(1.) The optimal $TCSD(Tr_1, Tr_2)$ can be obtained in $O(mn(m+n)^2 \lambda_7(m+n) \log(m+n))$;

(2.) An approximated $TCSD_A(Tr_1, Tr_2)$ with a loss-factor of 4 can be obtained in $O(mn(m+n)^2)$

4. EXPERIMENTAL EVALUATION

Our experiments were conducted on a PC with Pentium IV 3.0 GHz CPU, 1 Gigabytes of DDR2 memory and the Windows XP platform. We used both artificial and real data-sets. The artificial data set is generated using the network-based traffic generator [10] using the road network of Oldenburg. The trajectories in this data sets are temporally aligned and are of the same length. The realistic data set consists of 5000 trajectories of objects moving on the road segments of Cook County, Illinois. The length of these trajectories varies from 2 to 60 miles, and the average speed of the moving objects ranged between 10mph and 60mph. Additionally, we generated a "perturbed" set of trajectories as follows. For each trajectory in the original dataset, we randomly selected a value of ε and individual trajectory points were displaced by a randomly chosen value from the interval $[0.1\varepsilon, \varepsilon]$, and in a random $X - Y$ direction.

Our first group of experiments focus on the quality of Algorithm MMAD. We compared the values ε_{approx} obtained as an output of Algorithm MMAD with the values of ε_{opt}. More specifically, for each perturbed trajectory, we applied a randomly determined translation and rotation to obtain its variant of the perturbed trajectory. We then fed both the original trajectory and this variant into Algorithm MMAD to generate the approximate translation and rotation, and calculate the approximate distance. We measured the percentage of trajectories that yield an approximate distance within a given factor of ε_{opt}. The results are shown in the following table:

$\varepsilon_{approx}/\varepsilon_{opt}$	1.5	2.0	2.5	3.0	3.5
Trajectories	92.7%	98.5%	99.2%	99.68%	100%

As can be seen, for more than 95% of the trajectories used in the experiments, Algorithm MMAD had a loss factor ≤ 2, a loss factor ≤ 2.5 in more than 99% cases.

Our next group of experiments studied the selectivity of the TE_{ud} distance function for trajectories that have the same or similar routes. We picked a random trajectory Tr_{pick} from the Cook County data set and retrieved all the trajectories from the data set that have *routes* within a spatial (2-D Euclidian) distance of ε_S. For a filtering speed-up, the trajectories are indexed using an R-tree and we used the Linear Referencing System (LRS) of Oracle 9i [25] to measure the TE_{ud} distance during the refinement stage. Subsequently, from that subset, we obtained the *trajectories* whose dynamics are similar to Tr_{pick} and are within RTSD value of $\leq \varepsilon_S$, applying the proper translation/shifting in the temporal domain. For trajectories whose temporal duration are not the same, we measured the TCSD best-fit among all the possible translations of the shorter-duration trajectory into the one with a longer duration. In the experiments, we varied the value of ε_S from 0.5 to 2.5 miles and we averaged the results over 200 different choices of Tr_{pick}.

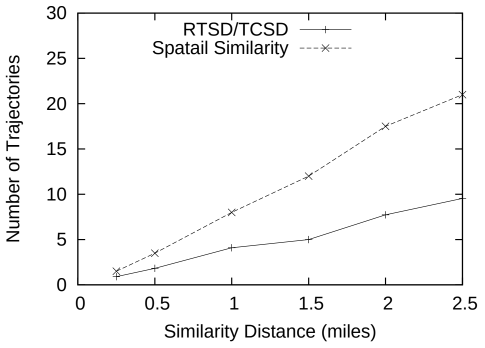


Figure 6: Trajectory Similarity *vs.* Route Similarity

As illustrated in Figure 6, the number of false-negatives can be quite large when the speed of the motion is not properly considered in the temporal dimension, and it increases in a linear manner with the value of ε_S. This illustrates the importance of our distance measure when considering the spatio-temporal similarities among moving object trajectories, as opposed to the pure spatial proximity of the moving objects' routes [9, 14].

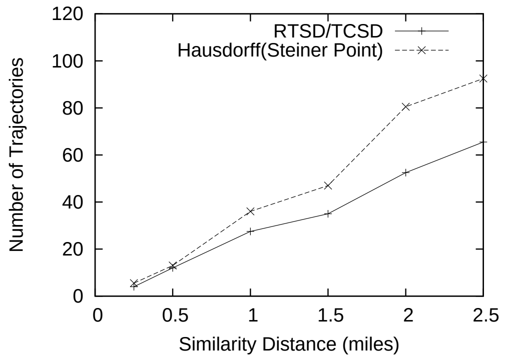


Figure 7: Dynamics-Aware *vs.* Steiner-point

Our last group of experiments measured the quality of our proposed techniques when, additionally, rigid transformations (translations + rotations) were applied in the spatial domain. A popular approach for matching 2-D polygonal shapes with respect to the Hausdorff distance is the one based on a *reference point* [3], relying on the *Steiner* point of compact 2-D sets. The work of [6] demonstrated that any point on a polygonal chain has a loss factor of 2 for the general Fréchet distance under *translations* only, however, there is no analogue of it for rotations. In the case of the 2-D routes of trajectories, we used the Steiner point calculated for the convex hull of the 2D projections of the points from the original trajectory and the anchor points of their corresponding backbones. As for rotation, we aligned every possible pair of respective points, and used the angle that would minimize the Hausdorff distance between the corresponding routes. Again, we randomly chose a trajectory Tr_{pick}, and found all the trajectories that are within ε_S distance to it. However, in this setting, we aim at finding similar trajectories by first allowing a pair of translation/rotation to minimize the distance. We used the trajectories from the Cook County data set, and varied the ε_S between 0.5 and 2.5 miles over 200 different choices of Tr_{pick} and observed the average selectivity. As shown in Figure 7, Steiner-point based routes-matching still introduces a large number of false negatives by up to 60%, since it fails to consider the speed-value, reflected in the temporal-dimension of the trajectories' points.

5. RELATED WORK

The problems of matching geometric shapes and point-sets are of interest for various applications areas: computer vision, pattern recognition, time-series-analysis, molecular biology, speech recognition and image processing [5, 17]. A survey of approaches for various geometric transformations and morphings including: (1.) rigid transformations (translations + rotations); (2.) homotheties (translations + scaling); (3.) general affine transformations that can occur in projections is presented in [5]. Initially, most of the efforts were focused on transformations that would minimize the Hausdorff distance between the two sets [3, 21, 17] for both cases of optimal and approximate matching. In particular, [3] introduced the concept of a reference point, which is the Steiner point for compact 2D polygons, and proposed an approximate matching algorithm with a loss factor of $1+4/\pi$. However (c.f. Section 2), for many applications, the Hausdorff distance is not an appropriate distance-measure, and efforts were geared towards the more dynamics-flexible *and* much harder to compute, Fréchet distance [6]. Specifically, [6] demonstrates that under Fréchet distance, a reference point of loss-factor 2 exists for matching polygonal curves under translations. The variations of the traditional Fréchet distance investigated in [4] pertain to the specific class of the κ-straight curves, in which the length of any arc is at most κ times their Euclidian distance. In such cases, Hausdorff-based algorithms can be used to approximate the Fréchet one. On the other hand, [7] considers κ-bounded and *backbone* curves, and introduces efficient approximation and pseudo-output-sensitive exact algorithms that calculate the discrete-Fréchet distance. A general discussion on various distance-measures for point sets and their computations is presented in [16]. Our work builds upon all these results, but its distinct nature is in the introduction of an intuitive distance-function for moving objects trajectories, which takes into consideration the practical assumption in MOD – the objects are moving with the constant speed along individual segment. As we demonstrated, the TE_{ud} a special case of the general Fréchet distance which enables minimization algorithms for point-sets fitting [22, 17] to be applied to efficiently obtain the trajectories' similarity under rigid transformations.

Recent efforts from the MOD and GIS communities have targeted the problems of similarity of spatial and spatio-temporal mobility patterns [9, 20, 14, 28, 29]. In particular, [14] proposed a Finite State Automaton based algorithm which, given a zone-based partition of the spatial domain, reduces the problems of routes and trajectories matching to the problem of recognizing regular expressions. A somewhat similar basis (discretizing the 2D space) was used in [9, 20] in order to address the processing of various spatio-temporal pattern queries, where the trajectories were represented as strings and the query patterns were detected based on string-matchings. Although the works have addressed larger categories of queries than we do, e.g., subsections of trajectories matching whereas we consider at most a containment of a shorter-trip into the longer-trip trajectory, our work uses a more intuitive distance-measure and does not require a discretization of the spatial dimension. More recent extensions [28, 29] proposed similarity measures based on the Longest Common Subsequence (LCSS) and addressed the problem of indexing large datasets for the purpose of the efficient evaluation of the similarity-based spatio-temporal queries. We have not addressed the problem of indexing in this work, however, we proposed a different distance function and we took into consideration the *rotation* in the spatial dimension. Recent results have connected the DTW with finding short-paths in combinatorial manifolds and relating it to the way light travels in media with different coefficients of refraction [15]. In this context, our approach enables an even further reduction of the number of elements in the dynamic-programming based matrix for calculating the paths in the manifolds. Similarity search for trajectories using data reduction (SVD, DFT, wavelets) is presented in [1], however the assumption is that the trajectories are already rotation-aligned.

6. CONCLUSIONS AND FUTURE WORK

We addressed the problem of matching moving objects trajectories, where rigid transformations, i.e., translation in spatial and temporal domain, as well as rotation in the spatial domain, can be applied in order to detect the similarity of their motion. We used an intuitive distance function TE_{ud} and we demonstrated that it is a special case of the Fréchet distance. Taking this into consideration along with the fact that in typical MOD settings the trajectories are assumed to have constant speed along individual trajectory-segments, we proposed both optimal and approximate algorithms for determining the translations and rotations that will minimize the respective $RTSD/TCSD$ similarity distances of two trajectories. There are several immediate extensions of our work. Firstly, we plan to adapt our results to the, so called, *decision* problem, which is, given two trajectories, and a constant ε *does there exist* transformation that will yield a similarity distance $\leq \varepsilon$? An important issue is the actual similarity-based query processing for massive trajectories' datasets, which ultimately demands an efficient indexing structures [28, 29]. Another avenue that we are planning to pursue is exploring the map-available data when the objects are constrained to move on road networks [18] and see whether some pre-processing can be done that will speed up the calculations of the similarity distances among trajectories, especially for real-time tracking data [30].

7. REFERENCES

[1] Rakesh Agrawal, Christos Faloutsos, and Arun N. Swami. Efficient similarity search in sequence databases. In *FODO*, 1993.

[2] A. Alt, C. Knauer, and C. Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38, 2004.

[3] H. Alt, O. Aicholzer, and G. Rote. Matching shapes with a reference point. In *Symp. Computational Geometry*, 1994.

[4] H. Alt, A. Efrat, G. Rote, and C. Wenk. Matching plannar graphs. *Journal of Algorithms*, 49, 2003.

[5] H. Alt and L. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In *Handbook of Computational Geometry*, 1999.

[6] H. Alt, C. Knauer, and C. Wenk. Matching polygonal curves with respect to the fréchet distance. In *STACS*, 2001.

[7] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk. Fréchet distance for curves, revisited. In *ESA*, 2006.

[8] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3), 1991.

[9] P. Bakalov, M. Hadjielefteriou, and V. Tsotras. Time relaxed spatiotemporal trajectory joins. In *GIS*, 2005.

[10] Thomas Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2), 2002.

[11] H. Cao, O. Wolfson, and G. Trajcevski. Spatio-temporal data reduction with deterministic error bounds. *VLDB Journal*, 15(3), 2006.

[12] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry: algorithms and applications*. Springer-Verlag New York, Inc., 1997.

[13] S. E. Dreyfus. An appraisal of some shortest – path algorithms. *Operations Research*, 17(3), 1969.

[14] C. du Mouza and R. Rigaux. Multi-scale classification of moving objects trajectories. In *SSDBM*, 2004.

[15] A. Efrat, Q. Fan, and S. Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *J. Mathematic Imaging and Vision*, 2007. to appear.

[16] T. Eiter and H. Mannila. Distance measures for point sets and their computation. *Acta Inf.*, 34(2), 1997.

[17] M.T. Goodrich, J.S.B. Mitchell, and M.W. Orletsky. Approximate geometric pattern matching under rigid motions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(4), 1999.

[18] R.H. Güting, V.T. de Almeida, and Z. Ding. Modeling and querying moving objects in networks. *VLDB J.*, 15(2), 2006.

[19] R.H. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[20] M. Hadjielefteriou, G. Kollios, P. Bakalov, and V. Tsotras. Complex spatio-temporal pattern queries. In *VLDB*, 2005.

[21] P.J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. In *Symp. Computational Geometry*, 1992.

[22] K. Imai, S. Sumino, and H. Imai. Minimax geometric fitting of two corresponding sets of points and dynamic furthest voronoi diagrams. *IEICE Trans. Inf. and Syst.*, E81-D(11), 1998.

[23] Forbes Online Issue http://www.forbes.com/home, April 2006.

[24] F. P. Preparata and M. I. Shamos. *Computational Geometry: an introduction*. Springer Verlag, 1985.

[25] J. Schiller and A. Voisard. *Location-based Services*. Morgan Kaufmann Publishers, 2004.

[26] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

[27] G. Trajcevski, O. Wolfson, K. Hinrichs, and S. Chamberlain. Managing uncertainty in moving objects databases. *ACM TODS*, 29(3), 2004.

[28] M. Vlachos, M. Hadjielefteriou, D. Gunopulos, and E. Keogh. Indexing multidimensional time-series. *VLDB Journal*, 15(1), 2006.

[29] M. Vlachos, G. Kollios, and D. Gunopulos. Elastic translation invariant matching of trajectories. *Machine Learning*, 58, 2005.

[30] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *SSDBM*, 2006.