Battlespace Movement Prediction for Time Critical Targeting

Mark J. Carlotto, Mark A. Nebrich, and Kristin O'Connor
General Dynamics Corp., 1400 Key Blvd., Suite 100, Arlington VA 22209

Abstract

A system is described for predicting the location and movement of ground vehicles over road networks using a combination of vehicle motion models, context, and network flow analysis. Preliminary results obtained over simulated ground vehicle movement scenarios demonstrate the ability to accurately predict candidate TCT locations under move-stop-move and other typical vehicle behaviors. Limitations of current models are discussed and extensions proposed.

Introduction

Efficient planning and execution of aerospace power at the operational and tactical levels is predicated on the ability to predict the location and movement of Time Critical Targets (TCTs) over time scales of minutes to hours into the future. Such a capability requires accurate knowledge of their location, destination, route, and behavior en route. State-of-the-art multiple hypothesis/feature aided trackers provide estimates of recent target states (location and ID) but cannot extrapolate these state estimates too far into the future using kinematic models alone. In order to predict the location of a TCT at some future time it is necessary to model where it is likely to go, and how it is likely to get there.

Predict is a software system developed under DARPA's Dynamic Tactical Targeting (DTT) program [1] for predicting the location and movement of ground vehicles over road networks using a combination of vehicle motion models, context, and network flow analysis. Preliminary results obtained over simulated ground vehicle movement scenarios demonstrate the ability to accurately predict candidate TCT locations under move-stop-move and other typical vehicle behaviors. The following sections discuss the prediction approach and algorithm, describe the software architecture and implementation, and present experiment results over several simulated operational scenarios.

Movement Prediction Algorithm

Let (i_0, j_0) be the location of a vehicle at time t_0 . The objective is to predict possible locations of the vehicle at time $t_0 + T$. Vehicles are limited to movement along road networks (cross-country movement is not allowed in the present implementation). The road network is stored in vector (arc-node) format and represented at run-time as a road category image $k(i, j)$. It is assumed that vehicles move independently of one another, and move purposefully toward one of N possible destinations along minimum travel time routes. Each road pixel (grid cell) represents a region $\Delta x \Delta y$ meters in size on the ground ($\Delta y = \Delta x$ for square pixels). A vehicle's motion is modeled statistically by the mean and standard deviation of its speed as a function of vehicle type m and road category k . We define the unit travel time array

$$\Delta t(i, j) = \Delta x / v(i, j) \quad (1)$$

to represent the time it takes a vehicle traveling at speed $v(i, j)$ to move through grid cell (i, j) . Two arrays are computed:

$$\begin{aligned} \Delta t_{\min}(i, j) &= \Delta x / v_{\max}(i, j) = \Delta x \sqrt{\Delta v_v^{k(i, j), m} + \Delta v_v^{k(i, j), m}} \\ \Delta t_{\max}(i, j) &= \Delta x / v_{\min}(i, j) = \Delta x \sqrt{\Delta v_v^{k(i, j), m} - \Delta v_v^{k(i, j), m}} \end{aligned} \quad (2)$$

which represent the mean plus and minus the 1-sigma times required to traverse a grid cell.

The total travel time array represents the time it takes a vehicle to move from a given starting point (i_0, j_0) to any point in the road network. The time to get to location (i_p, j_p) is computed recursively

$$t(i_p, j_p) = t(i_{p-1}, j_{p-1}) + \Delta t(i_p, j_p) \quad (3)$$

where the (i_p, j_p) indices are determined by the F* algorithm (see appendix). Two total travel arrays are computed assuming the vehicle is moving at its minimum and maximum speeds. The spatial sets

$$I_{\min}(i, j) = \begin{cases} 1, & t_{\min}(i, j) \leq T \\ 0, & \text{otherwise} \end{cases} \quad I_{\max}(i, j) = \begin{cases} 1, & t_{\max}(i, j) \leq T \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

define the vehicle's locations over the time interval $t_0 < t \leq t_0 + T$, where $t_{\min}(i, j)$ and $t_{\max}(i, j)$ are the total travel time arrays corresponding to the minimum and maximum unit time arrays. The 'exclusive-or' of these sets

$$I^*(i, j) = I_{\min}(i, j) \oplus I_{\max}(i, j) \quad (5)$$

gives the locations that can be reached at time $t = t_0 + T$ (Fig. 1).

Fig. 1 Locations that can be reached at $t = t_0 + T$

If a vehicle's motion is known with complete certainty ($\Delta_v = 0$), predicted locations are points in space-time. Uncertainty in motion ($\Delta_v > 0$) causes points to spread into regions where a vehicle may be located at a given time, or an interval of time when the vehicle may pass through a particular point in space.

It is assumed that vehicles move along minimum time routes to known destinations. Let $t_n(i, j)$ be a vehicle's travel time from location (i, j) in a track along a minimum time route to destination n , and let $t_n(i_{-1}, j_{-1})$ be the travel time from a previous location (i_{-1}, j_{-1}) in a track to the same destination. The probability that a vehicle is moving along route R_n , $p(R_n)$, is the probability that it is moving to site n

$$p(R_n) = \begin{cases} \frac{1}{N} & \text{if } t_n(i, j) - t_n(i_{-1}, j_{-1}) < 0 \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where N is the number of destinations with decreasing travel time. If we define the membership function

$$\mu(I_q, R_n) = \begin{cases} 1, & I_q \leq R_n \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where I_q^* is the q -th predicted location, we can assign a probability to the location based on the relative number of minimum time routes that pass through it:

$$p(I_q) = \prod_n \prod (I_q, R_n) p(R_n). \quad (8)$$

Fig. 2 illustrates the operation of the prediction algorithm. Fig. 2a shows a road network constructed from a vector file over Utah. The cell size is a system parameter (set here to 100 m/cell). Road class is depicted by gray level. The unit travel time array (Fig. 2b) depends on the vehicle's speed that is being modeled and varies spatially by road class. Here the ranges of values within the network are small (dark) compared to the off-road value (light). The total travel time (Fig. 2c) increases from the vehicle's current location (near the center).

a) Road network $k(i, j)$

b) Unit travel time array $t(i, j)$

c) Total travel time array $t(i, j)$

d) Reachable locations $I^*(i, j)$

Fig. 2 Predict example

Fig. 2d shows the locations that a vehicle can reach at time $t_0 + T$ given its current location at time t_0 . Uncertainty in location is due to uncertainty in speed (given by a vehicle's minimum and maximum speeds) over the time interval $(t_0, t_0 + T)$ and grows linearly with T . Reachable locations depend on the point of origin and travel time and are independent of destination. Travel time relative to each destination is measured, destinations where the travel time is decreasing are selected, and minimum travel time routes computed (Fig. 2e). Vehicles that reach a destination at $t < t_0 + T$ remain at the destination (i.e., there is no vehicle dwell model in the current system). Predicted locations are shown in Fig. 2f. Each location is weighted by the relative number of possible minimum time routes that pass through it (Eq. 8).

Fig. 3 Top-level Predict software architecture

Software Architecture

Fig. 3 shows the top-level *Predict* software architecture. Prediction parameters are specified in an ASCII file. The initialization process builds the lines of communication (LOC) or road network image and computes the unit travel time arrays for the vehicle of interest from its speed files. This process also includes adding connecting roads between off-road sites and the road network. The actual prediction process computes the total travel arrays as a function of the vehicle's current location, performs contextual analysis based on the sites (destinations) file, and generates a file of predicted locations.

Predict communicates with Fusion, Dynamic Sensor Management (DSM), and other processes in DTT by way of an XDA interface. A system controller (not shown) coordinates the operation of *Predict* with these other processes. As a vehicle moves over time, the vehicle locations are updated in the pin.txt file and the prediction process is repeated generating a new output (pout.txt) each time. The controller generates predictions for confirmed tracks whose state is "tracking", whose track update time differences exceed a given threshold (typically 30 seconds), and whose track positions have moved by at least one grid cell. It also accumulates TCT track data provided by Fusion for updating vehicle speed files. In this way *Predict* can adapt to actual operating conditions over time.

Measure of Performance

Predict assigns a probability to each location (Eq. 8). A detection is a prediction made at t_0 for $t_0 + T$ with probability $\hat{P} \geq \bar{P}$ that corresponds to the true location of the vehicle at $t_1 = t_0 + T$, where $\bar{P}$ is the effective operating point of the algorithm. False alarms are predicted locations with probability $\hat{P} \geq \bar{P}$ that do not correspond to true vehicle locations. The track probability $P_D(\bar{P})$ and false alarm rate $N_{FA}(\bar{P})$ are averages over a vehicle's track. The parametric plot of $P_D(\bar{P})$ vs. $N_{FA}(\bar{P})$ over operating point is the receiver operating characteristic (ROC) curve of the prediction algorithm over a track.

Fig. 4 Kansas data set

Experimental Results

Kansas Data Set - Fig. 4a shows 11 simulated vehicle tracks produced by Toyon's Ground Vehicle Simulator (GVS) [2] over a road network 5522 sq. km. in size in Kansas. Fig. 4b indicates the locations of possible sites. Fig. 4c plots the instantaneous speed of one of the vehicles, a T-72 tank, over one of the simulated tracks (track #4). Predictions were made 30 minutes into the future. Best performance was achieved using track history data to update the vehicle speed files. Fig. 4d is the ROC curve for track #4. A P_D of 0.9 was achieved at a false alarm rate of about 0.13 FA/sq. km. Table 1 summarizes the performance over all tracks. In track #8, the vehicle had stopped at a site for most of the run and no predictions were performed since it was not moving toward any other site.

Table 1 Summary of Kansas results

Track	P_d	FA/sq. km.	Track	P_d	FA/sq. km.
1	.95	0.12	7	.29	0.15
2	.84	0.1	8	0	0.18
3	.81	0.12	9	.67	0.1
4	.9	0.13	10	.88	.05
5	.88	0.05	11	.73	.05
6	.84	0.17			

Utah Data Set - This experiment involved predicting the movement of a single vehicle over an extended period of time over a road network in Utah (Fig. 5a). Over a period of 100 hours, the vehicle made 126 stops at 71 possible destinations (engagement sites, staging areas, and garrisons). The average stop time was 17.5 minutes, the minimum stop time was 10 seconds, and the maximum stop time was almost 25 minutes. Unlike Kansas where movement was over the a single road class, a range of speeds were observed in Utah as the vehicle traveled over different road classes (Fig. 5b).

a) Utah road network and known destinations

b) A portion of recorded speeds showing variation with road class

Fig. 5 Utah data set

The data were divided into 11 tracks. 6 of the tracks were too short to process (less than 30 minutes). Results for the remaining 5 tracks are summarized in Table 2.

Table 2 Summary of Utah results

Track	P_d	FA/sq. km.	Track	P_d	FA/sq. km.
1	.82	.07	7	-	-
2	.91	.07	8	.67	.05
3	-	-	9	-	-
4	-	-	10	.89	.06
5	.71	.02	11	-	-
6	-	-			

Summary and Future Work

A system to predict future locations of vehicles moving through road networks was described. Vehicle motion was modeled by a vehicle's average speed and standard deviation as a function of road category expressed in the form of travel time arrays. The F* algorithm was used to compute the minimum travel time to all locations in the network. Minimum travel time arrays are used to determine the locations in the network that can be reached at a given time into the future. By moving backwards in the direction of decreasing travel time from a set of possible destinations, minimum travel time routes to those destinations can be determined. A method of assigning probabilities to predicted locations based on the number of minimum time routes that pass through each location was described.

Future work will examine other probability assignments and the development of off-road models. Since the F* algorithm operates on arrays *Predict* can be easily extended to off-road situations. In the current system, off-road movement is effectively inhibited by setting the off-road speed to a small value. Increasing off-road speed significantly increases the search area. Future work will explore the use of terrain data (e.g., land cover and slope) to control the increase in search area.

Appendix - Minimum Travel Time Algorithm

Let $\{x_i\}$ be values in a network, where c_{ij} is the cost (time) to go from node i to node j. Consider the linear programming problem:

$$\begin{aligned} x_i + c_{ij} &\geq x_j \\ x_0 &= 0 \\ \text{minimize } x_n \end{aligned} \tag{A-1}$$

Ford [3] developed an efficient iterative algorithm for solving this problem. The nodes of the network are initialized as:

$$x_i = \begin{cases} 0, & i = 0 \\ \infty, & i \neq 0 \end{cases} \tag{A-2}$$

where the values represent path costs. For each connected pair of nodes,

$$\text{if } x_i + c_{ij} < x_j, \quad x_j = x_i + c_{ij} \tag{A-3}$$

It can be shown that Ford's algorithm yields the minimum cost solution, i.e., it finds the path (the set of nodes) with the lowest cost from an origin to a destination node. The indices $i_1, i_2, \dots, i_n$ define the minimum cost path where

$$x_{i_{k+1}} = x_{i_k} + c_{i_k i_{k+1}} \tag{A-4}$$

and are obtained by moving backward from the destination node in the direction of decreasing path cost. The value of the minimum cost path is

$$\sum_{i=0}^n x_i. \tag{A-5}$$

Fig. A-1 Illustration of Ford's algorithm

Fig. A-1 illustrates the operation of the algorithm for a simple network. Nodes that change state in each iteration are highlighted. The resultant minimum cost path is shown in Fig. A-2.

Fig. A-2 Minimum cost path found by Ford's algorithm

The F* algorithm is Ford's algorithm applied to images [4]. Let $c(i, j)$ be an array of costs. We define $p(i, j)$ as the path cost array which stores the path cost values (the x_i values of the nodes). The F* algorithm computes $p(i, j)$ from $c(i, j)$ using a variant of Rosenfeld and Pfaltz's distance transform [5]. A series of alternating passes on the arrays are performed. First is the top-to-bottom pass. Each row is processed left to right by

$$p(i, j) = \min \begin{matrix} c(i, j) + p(i-1, j-1), \\ c(i, j) + p(i-1, j), \\ c(i, j) + p(i-1, j+1), \\ c(i, j) + p(i, j-1), \\ p(i, j) \end{matrix} \quad (A-6)$$

and right to left by

$$p(i, j) = \min \begin{matrix} c(i, j) + p(i, j+1), \\ p(i, j) \end{matrix} \quad (A-7)$$

Next a bottom-to-top pass is performed in which the indices are flipped. As in Ford's original algorithm, F* iterations are performed until no node (no $p(i, j)$ value) changes state.

References

1. http://www.darpa.mil/body/NewsItems/darpa_fact.html
2. <http://toyon.com/gvs.html>
3. L. R. Ford, Network Flow Theory, *Rand Technical Report P-923*, August 14, 1956.
4. M.A. Fischler, J.M.Tenenbaum, and H.C.Wolf, "Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique," *Computer Graphics and Image Processing*, Vol. 15, pp 201-223 (1981)
5. A. Rosenfeld and J.J. Pfaltz, "Sequential operations in digital picture processing," *J. Assoc. Computing Machinery*, Vol. 13, No. 4, Oct, 1966, pp 471-494.