LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-404638

"Mtrack 1.0": A multi-vehicle, deterministic tracking algorithm

C. J. Carrano

June 12, 2008

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

"Mtrack 1.0": A multi-vehicle, deterministic tracking algorithm

Carmen J. Carrano
January 22, 2008
Lawrence Livermore National Laboratory, 7000 East Ave, Livermore, CA 94550

Abstract

As overhead persistent surveillance sensors become more capable at acquiring wide-field image sequences for longer time-spans, the need to exploit such data becomes ever greater. A desirable use of such sensors is to be able to track all the observable vehicles while they persist in the imagery. Typically, this imagery is characterized by lower resolutions (e.g. >= 0.5 m/pixel) and lower frame rates (e.g. ~ few Hz). We describe our initial implementation of a multiple-vehicle tracking algorithm for overhead persistent surveillance imagery. Subsequent reports will then present results of applying this tracker to specific persistent surveillance imagery as well as algorithm improvements and extensions for dealing with wider-field imagery.

1.0 Introduction

Tracking multiple vehicles in persistent surveillance image sequences is a very difficult problem in general for a number of reasons relating either to the scenery or the sensor:

- Vehicles are small (e.g. several to tens of pixels), or low contrast
- Many vehicles, some of which may be close together, with variable speeds and paths including stops.
- Obscurations of the path by buildings, overpasses, trees, terrain, or other vehicles.
- Other scene clutter that may appear to be moving such as parked cars, sun glinting, tall building edges, etc.

- Low frame rates
- Low resolution
- Grayscale information only
- Image stability issues

The initial deterministic approach we take here is based in part on our earlier single-vehicle deterministic tracking algorithm [1], but a number of modifications are needed to handle multiple vehicles. Our initial method relies on the mover map, path dynamics, and image features to perform tracking. It does very well when the vehicles stay separated and the obscurations are small enough such that the vehicles keep a constant speed and direction under the obscuration. We attempt to track vehicles that come to a stop such as at a stop sign or stop light, but have mixed results for such cases depending on the appearance and dynamics of target. It is arguable whether we should allow for stopping at the first stage of the tracker, but instead just generate reliable mover track segments that can be linked together in a later stage of tracking.

2.0 Pre-Processing

The basic processing steps that the raw camera data must go through before tracking is even possible are as follows given in Figure 1. Note that the blocks one and two could be swapped, but for the testing done thus far, the ordering was as in Figure 1.

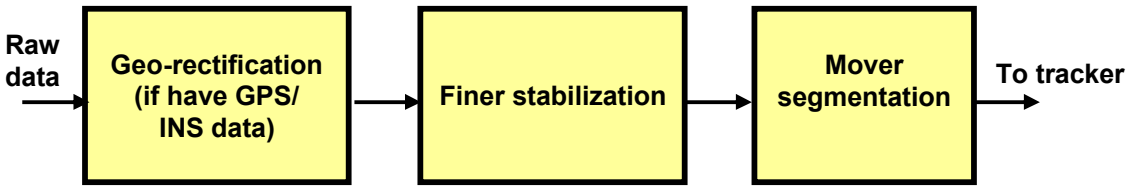


Figure 1: Basic processing steps prior to tracking

2.1 Geo-rectification

If streamed GPS and inertial navigation system data is available with the image sequence, it is possible to geo-rectify and resample the data to a NADIR looking view at a specific ground sampling interval. This has the advantage that it is much more straightforward to translate pixel positions in the image to latitude and longitude so that we get can get actual locations and paths of the vehicles for integration with geographical information systems. Describing the algorithm for doing this is beyond the scope of this report but can be found in [2].

2.2 Stabilization

As with most real-life navigations systems, the navigation data used in performing the geo-rectification is subject to errors. Likewise, the cameras won't necessarily be perfectly calibrated in position. As a result, the geo-rectified image sequence will not be suitably stabilized for best mover detection. If we can stabilize the image down to single pixel accuracy we will have optimal conditions for the mover detection step. Typically, for airborne imagery an affine-based stabilization works very well but a dense correspondence approach can also be used, especially if there is significant terrain relief or tall buildings. The stabilization algorithm we commonly use is described in [3] but can be found in common textbooks on the subject [4].

2.3 Mover segmentation

The mover segmentation is a very important step because the detected motion regions are used directly to guide the tracker. The purpose here is to generate a binary image for each intensity image in the sequence where a 1 indicates a motion region and a zero does not. After originally experimenting with a 3-frame-differencing approach [1], we are currently experimenting with a median-filter based approach which does a better job at filling in gaps in the larger vehicles, except at the slower velocities. It is a non-recursive, highly adaptive technique that uses a sliding time window. The user can pick the frame buffer size. With the lower frame frames, buffer sizes of 5 to 7 frames work well.

Although increasing the buffer size helps with slower movers, we have to be careful with larger images because we need to be careful about the memory usage as the frame buffer increases. If the image sizes become too big, it would be possible to split up the data as needed and apply mover segmentation on smaller regions.

The algorithm is conceptually quite simple. A block diagram of the algorithm is shown in Figure 2. An estimate of the stationary background at frame N is estimated by computing the temporal median at each pixel location over some number of frames, B. The absolute value of the subtraction of the background estimate from the current (foreground) frame N yields an absolute difference image where brighter regions have a greater probability of being in motion and darker regions are less likely to be in motion. At this point, a threshold based on the statistics of the difference image is applied. Typically 5*sigma to 8*sigma is good for selecting the motion regions. We have built in the ability to let the threshold vary spatially, so instead of calculating and using the sigma of the whole image at once to determine a single threshold, we can calculate it locally with some given region size. This would be useful for images whose mean intensity has some large scale variations. After the threshold step, we have always found it useful to reject all regions less than or greater than a certain pixel count. The pixel counts depend on the resolution of the imagery. Sometimes we also find it useful to apply a morphological closing operation with a small square or circular mask to clean up the blobs. Example imagery at each major step of the procedure is shown in Figure 3.

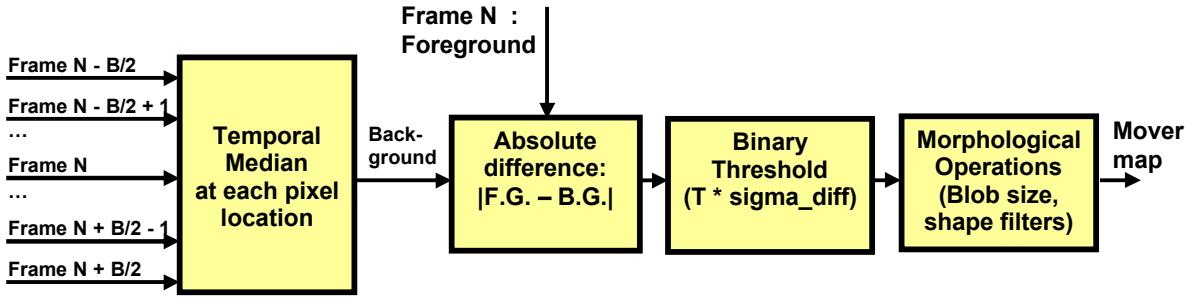
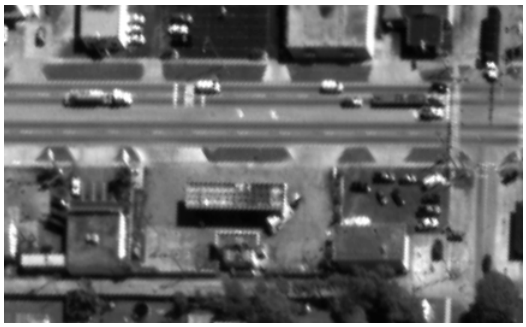
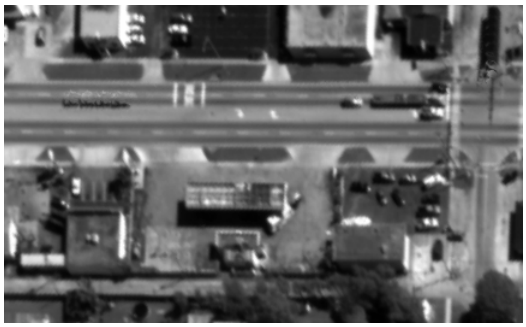


Figure 2: Block diagram of median-filter based motion segmentation.



3a: Foreground current frame N



3b: 7-frame temporal median

3c: Absolute difference

3d: Thresholded/etc.

Figure 3: Example imagery at each step of median-filter based mover segmentation.

Depending on how well the stabilization was done and the scenery content, what we see in the mover map may be exactly what we wanted (moving vehicles) or it may contain a number of clutter sources. These clutter sources come from buildings, especially tall buildings, terrain altitude variations, other moving objects (e.g. trees in the wind), and sun glints off water or parked cars. If the camera platform was stationary most of this clutter would not exist, except that caused by the wind, but if the camera platform is airborne and rotating about a particular location, for instance, then it will be an issue. There are various ways to deal with the clutter sources. Sometimes these can be cleaned up with the region size filter, but often these clutter sources will make it through the region size filters. A straightforward way to deal with movers not associated with vehicles on roads is to mask out regions where vehicles can't go. This mask can be either manually or automatically generated from the image itself or knowledge of the road network

3.0 Tracking algorithm

We now describe the current implementation of the multi-vehicle tracker.

3.1 Track file format

In order for the tracking to work, we need to be able to keep a record of each vehicle being tracked as well as some information about that vehicle. Currently, we keep track of the following information:

- Frame #
- Track ID
- Center position of region (x, y in pixels)
- Bounding box (x0, y0, xsize, ysize)
- x and y velocity (pixels/frame)
- track length in # of frames
- number of frames missed
- track status (0 for just initialized, 1 for moving, -1 for static, -2 for missed)

As the tracking algorithm matures it is possible that we could add additional information or use a separate working track file to store more key features of the tracks or the vehicles themselves (e.g. intensity or color histograms, certainty of the position/velocity estimates, …) We currently don't store lots of information about vehicle appearance because we have its position information and can extract out a template of the vehicle as the tracker proceeds.

This track-file format is useful for research purposes only, and is not intended to be used in a final product. A final track-file format should be one that will integrate into further exploitation tools and will necessarily include latitude and longitude and date/time-stamps.

3.2 Track initialization

Before vehicles can be tracked, they must be found in the first place. Ideally, we would like to have a perfect vehicle detector to initialize all the potential tracks, though many of the detected vehicles would be stationary and of little interest for tracking. Our track initialization approach is simple; on the first frame of interest, all validly detected motion regions are considered to be candidate movers and are initialized with a position and a bounding box. For example, the nine regions detected in Figure 3d are be initialized as shown in Figure 4. We can see here the importance of getting the mover segmentation right and potential tradeoffs with the threshold and morphological operations. We see here that the big truck on the left is separated into two regions and the big truck on the top-right is separated into three regions. If those trucks were traveling faster, there would be no gaps. If we allow a larger radius in the morphological operation, we can close those gaps, but this comes at the expense of not being able to distinguish two or more vehicles when they are close together because their regions would be merged. Future work may include adding in the extra step of trying to connect motion regions that appear to be from the same vehicle by analyzing object edge or boundary maps.

Because parked vehicles can start moving at any time or they can move in from the edge of the imagery, with each new frame it is necessary to check if there are new movers to initialize. As the track association proceeds, all movers that have been accounted for are zeroed out in the mover map and any remaining movers are then initialized as new movers.

Figure 4: Newly initialized tracks shown with initial bounding boxes and track ID's

3.3 Track association for just initialized tracks

Once new tracks are initialized, we only have a single frame of observation, which means we don't know the speed or direction of the vehicle yet. To accommodate this, we allow for a larger search space than usual and only use target features for association. The two features we currently use are the peak of the cross-correlation and an average intensity difference between the current vehicle and the candidate vehicle in the next frame. The candidate target with the highest cross-correlation peak whose intensity difference is low enough gets the match. The matching vehicle then gets a track status of 1, a track length of 1, and a velocity assigned to it.

Future work may include trying to detect the orientation of the vehicle as well as the road direction itself thus giving a better idea of where the vehicle could be headed, but the current algorithm does not do that. Improved performance may also include using a normalized and even perhaps a more direction sensitive correlation metric as well as other target features such as intensity (or color) histograms.

3.4 Track association once moving

Now that we have the initial velocity estimate for each track, we can predict the next position. The tracker loops through each vehicle in motion and tries to find the next location of the vehicle. Using the mover map, it only searches motion regions that fall inside its allowable search area. The maximum radius of the search area is defined at the start of the algorithm based on realistic maximum velocity changes we would expect. A diagram of this is shown in Figure 5.

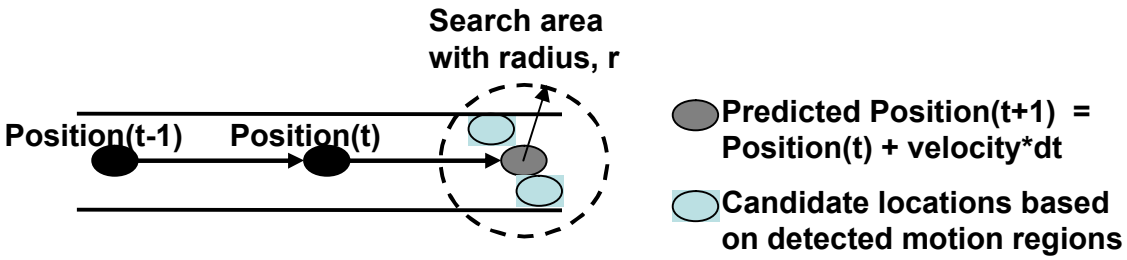
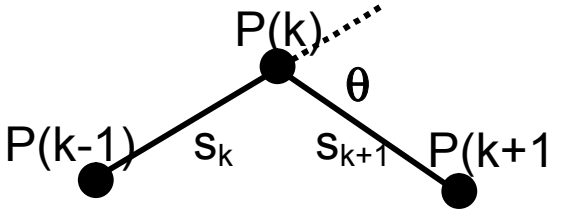


Figure 5: Diagram of the track association search area.

Currently four metrics are then computed for each candidate location and the candidate location with the "best" combination of metrics is selected. If the vehicle track has been established for a few frames and the vehicle is moving slow enough, we also check for the static case where the vehicle may have stopped. The same metrics from the single vehicle tracker are being used [1]. The following metrics are used:

- Peak of the cross correlation between template and candidate, which have each been demeaned and Hanning windowed. This helps us match the shape.
- Average intensity difference (or hue difference for color). This helps to prevent switching from a bright to a dark vehicle or visa-versa.
- Path coherence – a measure of agreement between the object trajectory and real life constraints. A low value is best.
    - Positive
    - Normalized
    - Reflects local absolute angular deviations
    - Responds equally to positive and negative velocity changes
    - Weights w1 and w2 allow emphasis of angular or velocity deviations. Currently we have w1=w2=0.5



$$\Phi\{P(k-1),P(k),P(k+1)\} = w_1(1-\cos\theta) + w_2\left(1-2\frac{\sqrt{s_k s_{k+1}}}{s_k + s_{k+1}}\right)$$

- Speed difference (i.e. acceleration) - path coherence was not quite enough information so looking at the simple speed difference is helpful. A low value is best.

The candidate target with a combination of good enough metrics gets the match. Ideally, the chosen target has the highest correlation peak, lowest intensity difference, best path coherence, and lowest speed difference, but often times that is not the case. If the dynamics have high scores, we can allow for less correlation and visa-versa. If no match can be found, we allow the vehicle to go missing (with a status = -2) and assign the

predicted position to it. If we don't detect the same vehicle within a specific number of frames, the track is terminated. The metrics chosen here are by no means optimal and better metrics are being considered.

We allow the bounding box to slowly change as the region size or shape changes. This is useful when the vehicle speeds up to reveal its true size or when a longer vehicle turns a corner. But it can cause confusion when a vehicle slows down causing the motion region to either shrink (on small vehicles) or break up (larger vehicles). The ideal thing to do would be to keep the vehicle area constant once we are sure of its footprint and not let it shrink.

3.4.1 Multi-target specific considerations

Tracking multiple or in fact every detected vehicle in an image sequence can be much more complicated than tracking only one vehicle, especially when the traffic density increases and the road network is complex. We can break down the situations into several cases.

- Best case: The vehicle matches exactly one moving region in the next frame. We update the track file with this new location and velocity and add one to the track length and keep or set track status to 1 (moving).
- Two or more vehicles match the same region. How best to handle this is currently under investigation. The current code is set to only allow one match per region because it zeros out matched regions in the mover map after they are matched the first time, which means that another future vehicle in the track association loop wouldn't even check that mover region because it no longer exists. An example of this is given in Figure 6a and 6b and 6c. We have experimented with allowing multiple matches to the same region by not zeroing out the mover map and allowing vehicles to be merged for a maximum number of frames before merging the two track IDs, but this approach needs more work.
- Mover region starts breaking up. This can happen when two or more close together vehicles begin to separate. In this case the best match to the larger region is chosen to keep the same track ID. The region that didn't get matched will then become a newly initialized region with a new track ID. An example of this is shown in Figure 6c and 6d with track ID 332 breaking up into track ID 332 and ID 431. Another reason this can occur is when a vehicle (this is more common for larger vehicles) slows down and the motion region starts to break up. Often, the front and back of a large vehicle will have separate motion regions when traveling slowly and the degree to which this occurs is dependent on the mover detection and segmentation algorithm, the GSD, how many frames are used, and the frame-rate. The same method for handling this scenario is used, but ideally we would want to improve how we handle this situation. Options include adding in some additional image processing to detect the vehicle size and shape at the first detection, or we could get it from when the vehicle is moving faster. In either case, once we know the estimated true size we want conserve the vehicle area and not let it break up.

Figure 6a

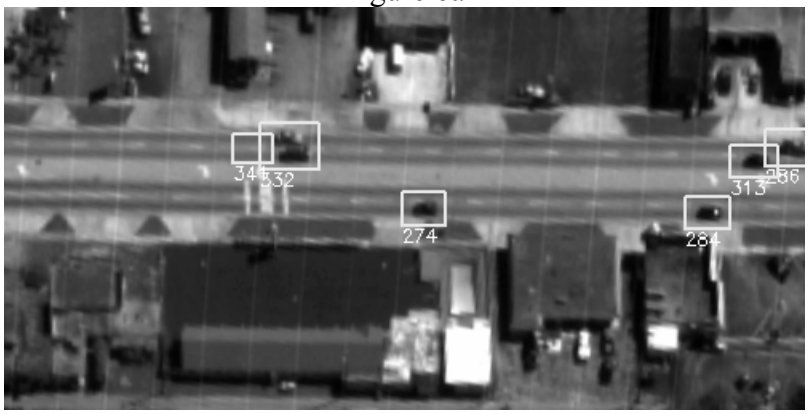


Figure 6b

Figure 6c

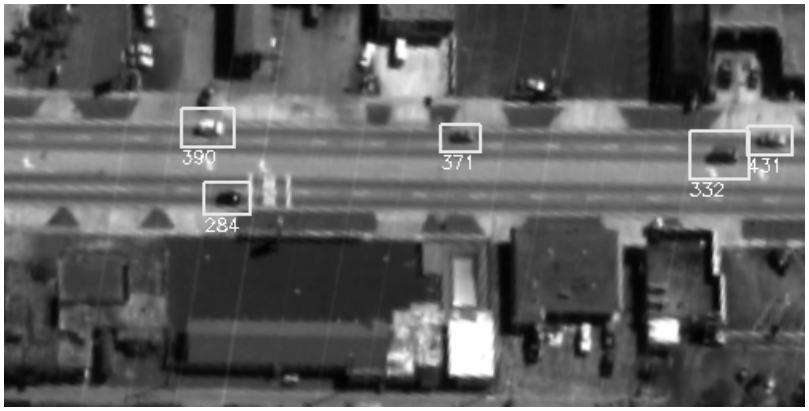


Figure 6d

Figure 6: Example of two vehicles merging together (ID341 and ID332), with the rule that vehicles can't merge. The position ID341 is predicted and still shown in Figure 6b, but is lost several frames later in Figure 6c since the cars (ID 332) are still in close proximity. The now separated vehicle gets a new ID of 431 in Figure 6d.

3.5 Track filters

After the tracking for the sequence is completed and a track-file generated, we find it useful to remove certain tracks that will probably be of little interest. The filters currently implemented are the following.

- Track length filter. User specifies the minimum length of the track in number of frames. All tracks less than that length are eliminated.
- Missing track filter. This is useful to do to the track file prior to creating a movie. It removes all entries from the track-file with a missing status (= -2).
- Salient track filter. The purpose of this filter is to remove any tracks that don't seem to be really doing anything. It looks at the track status column and it too many of them are stationary status (= -1), it removes them.

Note that these filters are completely optional. We mostly use the track length filter.

4.0 Performance

Generally, the performance of the mtrack algorithm is best when the vehicles have high contrast with respect to the road and are well separated from each other with minimal clutter or obscurations. We have had success with tracking through obscurations, either trees or buildings, primarily in situations when the vehicle keeps a constant velocity under the obscuration and does not stop or turn beneath it.
The detailed performance results of the current multi-tracker algorithm will be discussed in a follow-on report.

5.0 Conclusions

We have developed and implemented an initial version of a multi-vehicle tracking algorithm in IDL for tracking vehicles in persistent surveillance imagery. Future reports will include extensions for scaling the algorithms up to handle much wider field data as well as various improvements to the tracking algorithm.

References

1. C. Carrano, "Eztrack: A single vehicle deterministic tracking algorithm", June 2007, UCRL-TR-400667

2. M. Kartz, L. Flath, R. Frank, "Real-Time GPS/INS Correlated Geo-Registration and Image Stabilization of Streaming High-Resolution Imagery Utilizing Commercial Graphics Processors ",UCRL-ABS-204226

3. M. Duchaineau, **"**Progressive Dense Correspondence with Applications to Video Analysis", UCRL-ABS-225824

4. G. Wolberg, Digital Image Warping, IEEE Computer Press, 1990