Building robust appearance models using on-line feature selection

R. Porter, R. Loveland, E. Rosten

ABSTRACT

In many tracking applications, adapting the target appearance model over time can improve performance. This approach is most popular in high frame rate video applications where latent variables, related to the objects appearance (e.g., orientation and pose), vary slowly from one frame to the next. In these cases the appearance model and the tracking system are tightly integrated, and latent variables are often included as part of the tracking system's dynamic model. In this paper we describe our efforts to track cars in low frame rate data (1 frame / second), acquired from a highly unstable airborne platform. Due to the low frame rate, and poor image quality, the appearance of a particular vehicle varies greatly from one frame to the next. This leads us to a different problem: how can we build the best appearance model from all instances of a vehicle we have seen so far. The best appearance model should maximize the future performance of the tracking system, and maximize the chances of reacquiring the vehicle once it leaves the field of view. We propose an online feature selection approach to this problem and investigate the performance and computational trade-offs with a real-world dataset.

1. INTRODUCTION

Geographically referenced (geo-spatial) video acquisition systems are now in practical use. Wide area imaging sensors are placed on helicopters, balloons, small aircraft or unmanned aerial vehicle and geographically referenced video is communicated to a ground station in real-time. Compared to satellite imagery, which provides data at time scales of months or years, geo-spatial video provides data to observe and model temporal phenomena at time scales of seconds or minutes. Geo-spatial video exploitation presents new challenges for computer vision researchers. First, many objects of interest (e.g. vehicles and people) cover very few pixels and therefore specific recognition is very difficult. Second, moving object detection in this imagery is an unsolved problem. Data arrives at about 1 or 2 frames per second, which means point-like moving objects move anywhere from 1 to 200 pixels. In addition, the oblique viewing angles and incomplete digital elevation maps mean buildings and other landmarks suffer from parallax. This introduces a large amount of motion clutter. Finally, registration is often required in real-time and is therefore approximate, e.g., stationary objects might move up to 30 pixels over a short period of time. All these factors combined lead to unique, and extremely difficult recognition and tracking problems.

Lucas-Kanade[1,2] is perhaps the most well known template tracking algorithm, and works by having a template (a patch of an image) and a model for distorting the patch. The parameters of the distortion (often translation and rotation but can also include complex warps such as projective warps and even appearance models) are adjusted to minimize the sum squared error between the distorted image patch and the image. If the tracker is expected to work over a long period of time, then the template will cease to be a good match for the image, and tracking will be lost. Updating the template is difficult: even if tracking is very accurate the template appearance can drift over time and cease being an accurate representation of the object in question. Consequently considerable effort has been put in to extending the template model to account for appearance variations. Bergen et. al. uses arbitrary smooth, non-parametric warps, which allows the shape of the object to change in non-trivial ways[3]. Other approaches use a set of basis patches (learned from training examples) and compute the warp parameters and linear combination of basis patches for tracking[4,5]. Matthews et. al. also tackle the problem of updating the template[6]. Tracking is performed in two stages, first to the current template, then to the original template. If the relative appearance change between the two stages is small, then tracking is good, and the current template is updated from the current frame. This is extended to active appearance models by recomputing the set of appearance bases each time the template is updated. Several researchers approach adaptive appearance models in tracking as an online learning problem[7]. Jepson et. al. maintain a three part mixture model for the appearance and adapt the model parameters over time with an online EM-algorithm[8]. Perhaps most relevant to our paper, Collins et. al. implement online feature selection within a mean shift tracking system[9].

Recognition and tracking are intimately related and therefore in many tracking algorithms, recognition and tracking are tightly integrated. This type of approach is particularly powerful in high frame-rate, high resolution applications such as face recognition, where successful tracking depends on accurate estimation of latent variables related to 3d geometry, such as pose, and camera position. In our paper, we keep the recognition and tracking systems as independent as possible: the recognition system is responsible for producing likelihood images for the location, and the tracking system produces location and velocity estimates. This separation is partly motivated by the fact that our tracking system will eventually combine many different inputs, not just recognition. Examples of other inputs include moving object detection, and prior information related to geographic information systems such as road maps. It is also likely that the tracking system will also be used to combine information from other types of sensors.

2. THE DATASET

The number of pixels on target is very small (e.g., 6 by 20 pixels). There is also a large variation in ground sample distance, and large variations in lighting due to shadows and sun glint. Combined, this means the variation in appearance of the same vehicle can be very high, as shown on the left of Figure 1. There are also occasional occlusions, caused by trees and overpasses. On the right of Figure 1 we highlight the registration problems found in the dataset. This figure shows the results of stacking 100 frames, and then slicing the stack along a column, so that the horizontal axis corresponds to rows in the image, while the vertical axis corresponds to time. The view selected has us looking head on into the highway, so that the lanes can be seen on either side of the median, with cars appearing as the transient light blips. The wavy columns show the jitter present in the system; given perfect registration, the bright column of the median would be perfectly vertical in the image. It is apparent from the image that the jitter has both short (i.e. sub-frame interval) and long term frequency components, and reaches about 6 pixels per frame in the worst case present here. It is clear that registration correction will be essential for accurate tracking performance, however, in this paper we investigate what can be done with this data as provided.

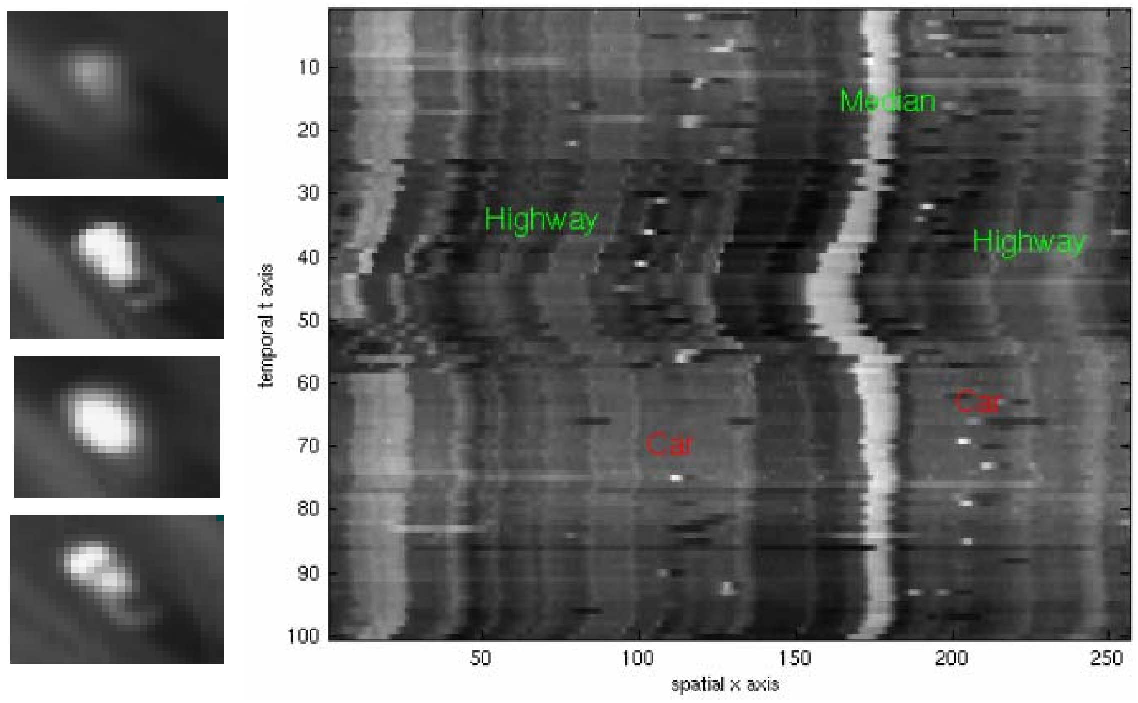

Fig. 1. Left: Four different views of the same vehicle taken over a short 30 second period. Right: a slice through the spatio-temporal image cube to illustrate registration artifacts.

3. RECOGNITION SYSTEM

In this section we investigate the performance of recognition in the geo-spatial video data set. Recognition is typically a delicate balance between specificity and invariance, appropriate to the problem. As illustrated in Figure 1, our problem is extremely difficult since the inter-class variability is very similar to the intra-class variability. In an effort to be as specific as possible, our recognition system is based on template matching, and our initial experiments investigated the best choice of distance function. We compared:

$$SAD_{(i,j)} = \sum_{k,l} \left| I_{(i-k,j-l)} - T_{(k,l)} \right| , \tag{1}$$

$$SSD_{(i,j)} = \sum_{k,l} \left(I_{(i-k,j-l)} - T_{(k,l)} \right)^2 , \tag{2}$$

$$NCC_{(i,j)} = \frac{\sum_{k,l} \left(I_{(i-k,j-l)} - \mu_{(i,j)} \right) \left(T_{(k,l)} - \mu_T \right)}{\sqrt{\sum_{k,l} \left(I_{(i-k,j-l)} - \mu_{(i,j)} \right)^2 \sum_{k,l} \left(T_{(k,l)} - \mu_T \right)^2}} . \tag{3}$$

where $I_{(i,j)}$ is the image intensity at location (i, j), T is the template and μ is the mean intensity over a template sized area and k,l vary over the template pixels. The template pixels are defined by masks which are manually specified. Examples of these masks are shown in the top row of Figure 2. Only pixels that lie within these masks contribute to the scores in Equations 1 through 3. The template matching algorithm is made rotationally insensitive by applying the template at 8 different rotations (45 degrees apart) and selecting the maximum score for NCC, and the minimum score for SAD and SSD.

We identified 5 instances of 10 different vehicles within the video dataset, and manually delineated each vehicle with a polygon paint tool. Figure 2 shows three of the largest vehicles within the set of 10, and also shows the vehicle masks associated with the single trailer vehicle. Each vehicle is used as the target class in turn. The template associated with this vehicle is applied to the 4 other images which contain the vehicle of interest. These images are typically around 512 by 512 pixels. The output from the (rotated) template match algorithm is then thresholded. We count detection if there is an above threshold pixel within the vehicle masks. We count a false alarm if any other pixel, is above threshold. This experiment is repeated for every vehicle within the set of 5, and for every one of the ten vehicles, for a total of 50 experiments. The ROC curves are averaged and shown in Figure 3, where it is clear that the NCC distance function is the best choice for this problem.

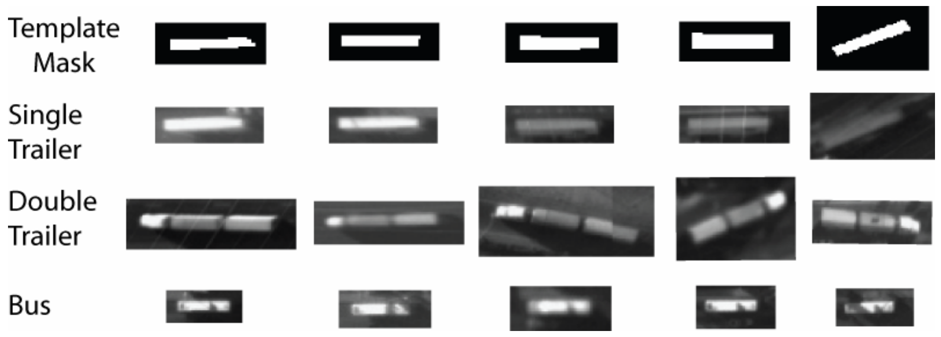

Fig. 2. Example of vehicles used to evaluate the different distance functions. The top row shows an example of the masks used to define which pixels contribute in Equations 1 through 3.

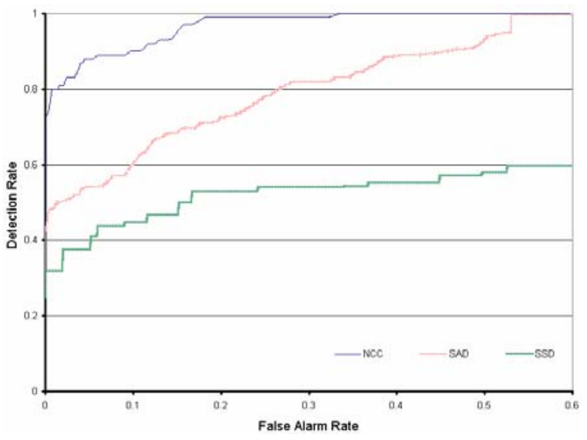

Fig. 3. Detection Rate versus False Alarm Rate for the three distance functions over 50 trials.

4. TRACKING SYSTEM

In the traditional tracking problem we assume at a given time t, a set of observations y_t, and state variables x_t. The goal is to estimate the current state, x_t, given all observations seen so far. Assuming a first order Markov process, and applying several independence assumptions, we arrive at the standard Bayesian tracking recursion[10] :

$$\begin{aligned} p\left(x_t \middle| y_{1:t-1}\right) &= \int p\left(x_t \middle| x_{t-1}\right) p\left(x_{t-1} \middle| y_{1:t-1}\right) dx_{t-1} \\ p\left(x_t \middle| y_{1:t}\right) &\propto p\left(y_t \middle| x_t\right) p\left(x_t \middle| y_{1:t-1}\right) \end{aligned} \tag{4}$$

The first step is a prediction step; the second is filtering. The technique requires us to provide a model of state evolution, $p\left(x_t \middle| x_{t-1}\right)$, and a likelihood function, $p\left(y_t \middle| x_t\right)$ appropriate to our application. We use the popular Particle Filter to efficiently estimate Equation 4 using sequential Monte Carlo techniques[11]. The output from the tracking system is the estimated mean of $p\left(x_t \middle| y_{1:t}\right)$.

4.1 State Evolution

In our application the state variable x_t has four variables: two for position and two for velocity. We implement a simple linear motion model where the position is updated based on the current velocity. Due to the low frame rate, and high level of jitter, the velocity is propagated with additive noise. The noise distribution is an asymmetric Gaussian whose major axis is orientated to the velocity direction. The aspect ratio of the Gaussian is proportional to the velocity magnitude. This distribution is motivated by the observation that vehicles traveling at high speed are less likely to change direction.

4.2 Likelihood Function

In this paper, the only input to the tracking system is the recognition algorithm, which is applied to each frame independently. This means we only have an observation related to the position state variables. For computationally efficiency, the recognition algorithm is only applied to a finite window surrounding the current state estimate. In our experiments this window was 256 pixels by 256 pixels. For positions outside this window, the likelihood function returns 0.

We found that using the recognition algorithm output directly was a poor choice for the likelihood function. In simple terms, the template distance function is not a smooth function of the position. Peaks in the distance function are often single pixel wide, which means there is little difference between the likelihood of states close to correct, and the states which are widely incorrect. Our solution is to apply an adaptive threshold to the distance function output, in which the 25 highest pixel values are set to 1, and the remainder set to 0. We then apply Gaussian smoothing with a 15 pixel variance.

5. ADAPTIVE APPEARANCE MODELS

We assume that a track is manually initiated, by specifying a location of a vehicle in a particular frame, and the polygon mask, much like those used in the recognition experiments. The simplest approach is static and involves repeated application of the user defined template on all subsequent frames. We will compare several adaptive recognition systems to this static system in our experiments.

5.1 Adaptive Single Template model (AST)

At one extreme, the state estimate is used to redefine the recognition model at each frame. This approach is problematic due to its high sensitivity to estimation errors, however we include it in our experiments for comparison. A key factor in obtaining specificity with our template matching approach is the asymmetric masks which separate the vehicle pixels from the background. If we are to update the template, using on the tracking systems predicted location, this mask information is no longer available. Our solution to this problem is to affine warp the new template to the original template, using a hierarchical mean-squared error registration technique provided by Philippe Th´evenaz's Pyramidal Sub-registration software[12]. The package performs a multi-scale affine match in order to avoid being trapped in local minima during the optimization process. We found large vehicle rotations (e.g. vehicles turning corners) were sometimes problematic, and therefore we rotate the image in 22.5 degree intervals, and run the registration algorithm 16 times to find the best match.

5.2 Growing Multiple Template model (GMT)

A second technique is to accumulate templates throughout the track. Since our dataset is relatively short (44 frames) it was computationally feasible to keep the first user defined templates, and all subsequently predicted templates as the recognition model. As templates are accumulated they are warped to the user defined template as with the AST method. All templates are applied in turn, and the maximum response is retained. We then threshold and smooth as described in section 4.2. This approach is typically far too expensive to use in practice. We implement it in this paper for comparison only.

5.3 Adaptive Multiple Template Model (AMT)

Ideally an adaptive appearance model would combine the strengths of the above approaches and sensibly incorporate new information when appropriate. We would like the execution time of the system to be constant throughout the track, similar to the AST and static template methods, but we would also like the model to capture more or less variability as required, similar to the GMT approach. We can consider this a feature selection problem: at each time step we simply choose a subset of the templates from all the templates we have seen so far. One way to choose this subset is to use a training set. The basic idea is illustrated in Figure 4. Positive training samples selected by tracking system. Negative training samples are selected from a surrounding grid. In our experiments we choose 8 negative examples spaced 100 pixels apart.

We store features (templates) and training samples separately. Features are image patches that have been extracted from the image, and warped to the first frame template. This warping was described in Section 5.1, and is necessary to align the vehicle patch with a template mask. We also store image patches the same size as the original template for each training sample, but these are not modified. Each template is applied to all training samples. In figure 4 the grayed areas correspond to the samples where the template is applied to samples drawn from the same frame. Applying templates to all samples is order $O(T^2)$, however, it is computed incrementally, as shown by the dashed samples in Figure 4, and therefore at any time T, only 2kT-k evaluations are required. In our experiments T was small, and this execution time was negligible compared to applying the recognition model itself. In a complete system samples should be discarded.

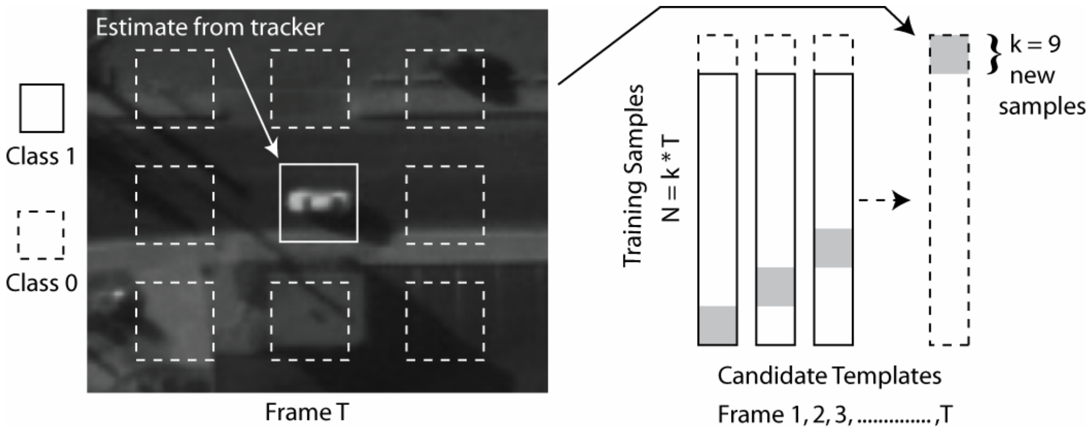

Fig. 4. At each time step we add a new feature, or candidate template, and we also add a number of labeled training samples.

After updating the training set, we must choose the subset of templates from the candidate templates stored so far. We make this choice incrementally using the AdaBoost discrete boosting algorithm[13]. AdaBoost optimizes a large-margin loss function for an additive model of the form:

$$h(x) = \sum_{i=1}^{F} w_i g_i(x) \tag{5}$$

where F is the maximum number of features and $g_i(\vec{x}) \in \{-1,1\}$ is a weak learner, or in our case, a NCC score which has been thresholded. The linear combination in Equation 5 starts with zero terms. For each of F iterations we add one new feature $g_i(\vec{x})$ to the model. To choose which feature to add, we take each candidate template in turn, and find the minimum error threshold on the training set. We choose the feature with the minimum, minimum error threshold. This operation is O(TNlogN).

Several other incremental algorithms have been suggested for online feature selection could also be used for this problem[14]. For example, our algorithm does not discard candidate features over time. Obviously, as T gets large, this will become essential. One of the advantages of the AdaBoost algorithm is that it allows us to easily incorporate biases into the training set by weighting samples differently. For example, we may like to place high importance on the first sample, since it is known to be correct. In our experiments the first 9 samples receive as much weight as the rest of training set combined. In our application it would also be possible to weight all additional samples by a confidence estimate which could be derived from the tracking system. Finally, it is possible that high detection rate is more important for this application than a low false alarm rate. In our experiments we share the weight amongst positive and negative classes equally.

Once we have selected F features, they must be combined to produce the final likelihood image. Ideally we would use Equation 5 to combine features, however the NCC output is poorly conditioned, and the thresholds found during Boosting are typically a poor choice for the final model. Our solution was to take the maximum response from the selected templates, as we do in the GMT algorithm. Although this method of combination sounds very different from Equation 5, the two models are somewhat similar when the output $h(\vec{x})$ is thresholded, as it is in our system (Section 4.2). That is,

$$T_0\left(\sum_{i=1}^{F} w_i T_{V_i}\left(NCC_i\right)\right) = T_V\left(\underset{i\in\{1..F\}}{Max} NCC_i\right) \quad when \;\; w_i = 1 \; and \; T_{V_i} = T_v \;\; for\, all\, i \tag{6}$$

where $$T_V(x) = \begin{cases} 1 & if\ x > V \\ 0 & otherwise \end{cases}. \tag{7}$$

6. EXPERIMENT

We compare the adaptive appearance models described in the previous section using real world data. We manually identified 5 vehicles whose trajectories cover a significant proportion of the dataset, as shown in Figure 5.

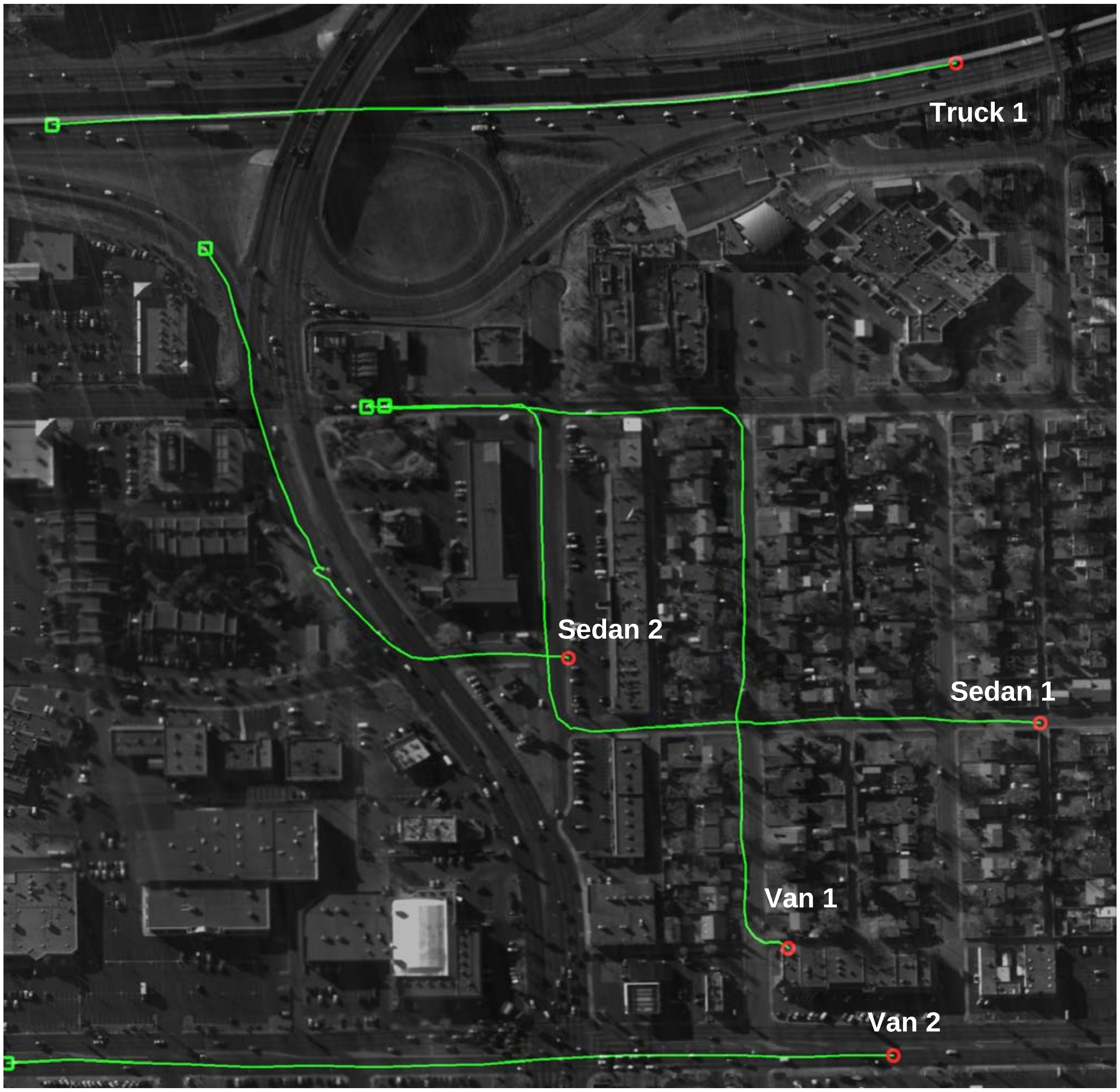

Fig. 5. The five routes that were manually identified to evaluate algorithm performance.

We compare these manually specified trajectories to predicted trajectories to evaluate accuracy. Notice in the Sedan 2 track, the ground truth trajectory appears noisy due to image jitter. At this point the vehicle was stationary in traffic for a number of frames. For each frame, the predicted location must be within a radius of 10 pixels to be considered correct. The accuracy is then the percentage of frames within the track that are predicted correctly. Since the Particle Filter is stochastic, we applied each method 5 times, and averaged the results. For the AMT method we used F=5. The results are summarized in Table 1.

Table 1. Accuracy estimates for the 5 trajectories shown in Figure 5.

Trajectory	Static	AST	GMT	AMT
Van 1	65.4	44.3	10.0	67.2
Sedan 1	20.0	10.0	7.5	22.5
Sedan 2	31.8	10.6	10.0	28.8
Truck	41.9	30.5	37.4	32.8
Van 2	94.4	97.8	100.0	100.0

6.1 Discussion

The performance of the different adaptive appearance models varied greatly, and it is clear from Table 1, that it was hard to improve upon the single static template. We hypothesize that this is partly due to the extreme difficulty in solving this problem with recognition alone. Visual inspection of the static template trajectories often show the tracking system output off-target, and it was only though large numbers of particles (50000), that the trajectory was reacquired. Since the system has difficulty remaining on track for enough time to collect reasonable templates the naïve AST and GMT techniques perform extremely poorly. We observe that the AMT technique performs fairly well under these circumstances, perhaps in part, to the large weight placed on the first frame samples.

The Van 2 route is the easiest of the tracking problems to solve and the adaptive techniques consistently outperformed the static algorithm. Upon closer inspection, we observed that the static technique missed the last frame of the trajectory, where the Van had moved halfway out of the image. It was interesting to note that the adaptive techniques all performed better in this instance. Given these two examples, the AMT technique appears to combine the best qualities of both static and adaptive approaches, although, further improvements will be required in order to guarantee equivalent performance to the static case e.g., by incorporating tracking system confidence for predictions.

7. CONCLUSIONS

We have presented a novel application of incremental learning algorithms for online feature selection in tracking. The approach is general purpose, computationally efficient and can incorporate prior information easily. We applied the algorithm to a difficult real-world problem and obtained promising results. In future work we look to supply additional inputs, and prior knowledge, to the tracking system, so that we can provide the necessary context to overcome the difficult image quality and registration problems, and obtain reasonable tracking performance.

REFERENCES

1. S. Baker and I. Matthews. "Lucas-kanade 20 years on: A unifying framework.", *International Journal of Computer Vision*, 56(3), 221-255, (2004).
2. B. D. Lucas and T. Kanade. "An iterative image registration technique with application to stereo vision.", *7th International Joint Conference on Artificial Intelligence*, 674-679, (1981).
3. J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. "Hierarchical model-based motion estimation.", *Proc. of ECCV*, 237-252, (1992).
4. G. Hager and P. Belhumeur. "Efficient region tracking with parametric models of geometry and illumination", *IEEE Trans Pattern Analysis and Machine Intelligence*, 20(10), 1025-1039, (1998).
5. M. Black and A. Jepson. "Eigen-tracking: Robust matching and tracking of articulated objects using a view-based representation", *Proc. of ECCV*, 329-342, (1998).
6. I. Matthews and T. Ishikawa and S. Baker. "The template update problem", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6), 810-815, (2003).
7. Kuang-Chih Lee and David Kriegman, " Online learning of probabilistic appearance manifolds for video-based recognition and tracking ", *Proc. CVPR,* 852-859, (2005).
8. Allan D. Jepson, David J. Fleet and Thomas F. El-maraghi, "Robust Online Appearance Models for Visual Tracking", *Proc. CVPR,* 415-422 (2001).
9. R. T. Collins, Y.X. Liu, and M. Leordeanu, "Online selection of discriminative tracking features", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 27(10), 1631-43 (2005).
10. Lawrence D. Stone, Carl A. Barlow and Thomas L. Corwin, *Bayesian Multiple Target Tracking,* Artech House, Boston, 1999.
11. Michael Isard and Andrew Blake, "CONDENSATION - conditional density propagation for visual tracking", *Int. J. Computer Vision* 29(1), 5-28, (1998).
12. P. Th´evenaz, U.E. Ruttimann, and M. Unser, "A pyramid approach to subpixel registration based on intensity", *IEEE Transactions on Image Processing,* 7(1), 27–41, (1998).
13. Y.Freund, and R.E. Schapire. "Decision-theoretic generalization of on-line learning and an application to boosting", *Journal of Computer and System Sciences,* 55(1), 119-139, (1997).
14. K. Glocer, D. Eads, and J. Theiler. "Online Feature Selection for Pixel Classification.", *Proc. ICML* 22, 249-256 (2005).