A recurrent velocity filter for detecting large numbers of moving objects

R. Porter*, A. Fraser, R. Loveland, E. Rosten
Space and Remote Sensing Sciences Group,
Los Alamos National Laboratory, Los Alamos, NM, USA 87545

ABSTRACT

We present a method for detecting a large number of moving targets, such as cars and people, in geographically referenced video. The problem is difficult, due to the large and variable number of targets which enter and leave the field of view, and due to imperfect geo-projection and registration. In our method, we assume feature extraction produces a collection of candidate locations (points in 2D space) for each frame. Some of these locations are real objects, but many are false alarms. Typical feature extraction might be frame differencing, or target recognition. For each candidate location, and at each time step, our algorithm outputs a velocity estimate and confidence which can be thresholded to detect objects with constant velocity. In this paper we derive the algorithm, investigate the free parameters, and compare its performance to a multi-target tracking algorithm.

Keywords: multi-target tracking, moving object detection, wide-area video.

1. INTRODUCTION

Wide area imaging sensors can be placed on helicopters, balloons, small aircraft or unmanned aerial vehicles and geographically referenced video communicated to a ground station in real-time. Compared to satellite imagery, which provides data at time scales of days or months, geo-spatial video provides data to observe and model temporal phenomena at time scales of seconds or minutes. Wide area video presents new challenges for object detection and tracking. Many objects of interest (e.g. vehicles and people) cover very few pixels and therefore accurate detection is very difficult. Data arrives at about 1 or 2 frames per second, which means *point-like* moving objects move anywhere from 1 to 200 pixels. Data is typically acquired at oblique viewing angles, which means buildings and other tall landmarks suffer from parallax which introduces a large amount of motion clutter. In addition, registration is often required in real-time and is therefore approximate, e.g., stationary objects might move up to 10s of pixels. Robust moving object detection can play two roles in this dataset:

1. The first is as a preprocessor to a tracking system. Optimal multi-target tracking must deal with a combinatorial number of hypotheses and many algorithms do not scale well with the number of observations. Moving object detection can help with this problem by filtering through 10's of thousands of detections and placing greater emphasis on locations most likely to be real targets.

2. The second role of moving object detection is to provide more general statistics. By accumulating local velocity estimates over a particular location, over long periods of time, we can produce many useful data products. For example, consistent velocity measurements along a line could indicate the presence of a road. This is much simpler problem than tracking all moving objects, and typically, track identity is less important than accumulating representative statistics.

In Section 2 we introduce the algorithm, and discuss its parameters and implementation. In section 3 we use synthetic data to investigate the filter's parameter choices and then evaluate the performance of the algorithm in comparison to a multi-target tracking system in Section 4. We provide an example of a real-world geo-spatial video application in Section 5 and discuss future directions in Section 6.

*rporter@lanl.gov; phone 1 505 665 7508; fax 1 505 664 0362; pooka.lanl.gov

2. RECURRENT VELOCITY FILTER

2.1 Definition

To introduce the Recurrent Velocity Filter (RVF) we first consider the situation where we have one observation, per time step, over a number of time steps: $\mathrm{X_T} = \{\mathrm{x_1}, .., \mathrm{x_T}\}$. We observe approximate positions, and we imagine they are generated by an object with a constant velocity trajectory. We compute the first order differences, $\mathrm{D_T} = \{\mathrm{d}_{2,1}, \mathrm{d}_{3,2}, .., \mathrm{d}_{\mathrm{T,T-1}}\}$, where $\mathrm{d_{i,j}} = \mathrm{x_i} - \mathrm{x_j}$and estimate the velocity v given the differences:

$$\mathrm{p}(v|\mathrm{D_T}) \propto \mathrm{p}(\mathrm{D_T}|v)\mathrm{p}(v). \tag{1}$$

We assume that the differences are independent and identically distributed:

$$\mathrm{p}(\mathrm{D_T}|v) = \prod_{\mathrm{t=1}}^{\mathrm{T}} \mathrm{p}\big(\mathrm{d_{t,t-1}}|v\big). \tag{2}$$

Now we consider the more complex situation where we have a large, variable number of observations each time step: $\mathrm{X_T} = \Big\{\{\mathrm{x}_{1,1}, .., \mathrm{x}_{1,\mathrm{N}(1)}\}, \ldots.., \{\mathrm{x}_{\mathrm{T},1}, .., \mathrm{x}_{\mathrm{T,N(T)}}\}\Big\}$. In this case, we imagine the observations are generated by $\mathrm{N(t)}$ different objects, each with a constant velocity trajectory. Note that the number of trajectories assumed responsible for generating the observations varies over time and is equal to the number of observations at a given time. We would like to estimate the velocities of the trajectories, $V_t = \big[v_{t,1}|v_{t,2}| \ldots |v_{t,N(t)}\big]$, where $v_{\mathrm{t,i}}$ is the velocity of the trajectory generating observation i at time t. Following (1), we base the estimate on differences, which in this case, includes all pair-wise differences between observations in consecutive frames:

$$\mathrm{d_{i,j}} = \mathrm{x_{t,i}} - \mathrm{x_{t-1,j}} \quad \forall\ \mathrm{i, j, t}. \tag{3}$$

To simplify the problem we treat the trajectories at any given time as independent:

$$\mathrm{p}(V_t|\mathrm{D_t}) \propto \prod_{\mathrm{i=1}}^{\mathrm{N(t)}} \mathrm{p}\big(v_{\mathrm{t,i}}|\mathrm{D_t}\big). \tag{4}$$

Estimating $\mathrm{p}\big(v_{\mathrm{t,i}}|\mathrm{D_t}\big)$ is still difficult since we do not know which observations were generated by the trajectory (although we do know the trajectory ends with observation $x_{t,i}$). We define a mapping, $M(t) \in \mathcal{M}$, which identifies a unique sequence of observations through time. $\mathcal{M}$ is the set of all possible sequences and we assume that it is distributed uniformly and independently, so that $P(M) = {}^{1}\!/_{|\mathcal{M}|}$. If we were given the correct M for the trajectory ending at observation i at time T (i.e. a trajectory where $M(T) = i$), the likelihood function (2) would be straightforward:

$$\mathrm{p}\big(\mathrm{D}|v_{T,i}, \mathrm{M}\big) = \prod_{\mathrm{t=1}}^{\mathrm{T}} \mathrm{p}\big(\mathrm{d}_{\mathrm{M(t),M(t-1,)}}|v\big). \tag{5}$$

However, since the correct mapping is not known, we must consider all possible sequences. From (1) and (5):

$$\mathrm{p}\big(v_{\mathrm{T,i}}|\mathrm{D_T}\big) \propto \sum_{\mathrm{M}\in\mathcal{M}} \prod_{\mathrm{t=1}}^{\mathrm{T}} \mathrm{p}\big(\mathrm{d}_{\mathrm{M(t),M(t-1,)}}|v\big)\, \mathrm{p}\big(v_{\mathrm{T,i}}\big). \tag{6}$$

This sum grows exponentially in time, but we can reorder the operations as follows:

$$\mathrm{p}\big(v_{\mathrm{T,i}}|\mathrm{D_t}\big) \propto \sum_{\mathrm{j\in N(t-1)}} \mathrm{p}\big(\mathrm{d_{i,j}}|v\big) \sum_{\mathrm{k\in N(t-2)}} \mathrm{p}\big(\mathrm{d_{j,k}}|v\big) \ldots \sum_{\mathrm{n\in N(1)}} \mathrm{p}\big(\mathrm{d_{m,n}}|v\big)\mathrm{p}\big(v_{\mathrm{T,i}}\big). \tag{7}$$

This can be implemented recursively, which leads to our basic definition of the Recurrent Velocity Filter:

$$\mathrm{p}\big(v_{\mathrm{t,i}}|\mathrm{D_t}\big) \propto \sum_{\mathrm{j=1}}^{\mathrm{N(t-1)}} \mathrm{p}\big(\mathrm{d_{i,j}}|v\big)\mathrm{p}\big(v_{\mathrm{t-1,j}}|\mathrm{D_{t-1}}\big). \tag{8}$$

While this equation is more efficient to implement than (6), it is still problematic since the number of terms in the posterior is exponential in time. One way to address this problem is to approximate the density functions in (8) with histograms. An alternative explored in this paper is to approximate the posterior with a single term at each time step.

2.2 Implementation with Gaussians

We represent the density functions with Gaussians:

$$\begin{gathered} \mathrm{p}\left(\mathrm{d}_{\mathrm{i,j}} \mid v\right) = \mathcal{N}\left(\mathrm{d}_{\mathrm{i,j}}, \Sigma_{\mathrm{P}}\right) \\ \mathrm{p}\left(v_{0,\mathrm{i}} \mid \mathrm{D}_0\right) = \mathcal{N}\left(\mu_0, \Sigma_0\right) \end{gathered} \tag{9}$$

where $\Sigma_{\mathrm{P}}, \mu_0, \Sigma_0$ must be provided. If we approximate:

$$\mathrm{p}\left(v_{\mathrm{t-1,j}} \mid \mathrm{D}_{\mathrm{t-1}}\right) \propto \mathcal{N}\left(\mu_{\mathrm{t-1,j}}, \Sigma_{\mathrm{t-1,j}}\right) \tag{10}$$

then we can write (8) as:

$$\mathrm{p}\left(v_{\mathrm{t,i}} \mid \mathrm{D}_{\mathrm{t}}\right) \propto \sum_{\mathrm{j=1}}^{\mathrm{N(t-1)}} \mathcal{N}\left(\mathrm{d}_{\mathrm{i,j}}, \Sigma_{\mathrm{P}}\right) \mathcal{N}\left(\mu_{\mathrm{t-1,j}}, \Sigma_{\mathrm{t-1,j}}\right) \tag{11}$$

or:

$$\mathrm{p}\left(v_{\mathrm{t,i}} \mid \mathrm{D}_{\mathrm{t}}\right) \propto \sum_{\mathrm{j=1}}^{\mathrm{N(t-1)}} z_{\mathrm{j}} . \mathcal{N}\left(\mu_{\mathrm{j}}^{\mathrm{c}}, \Sigma_{\mathrm{j}}^{\mathrm{c}}\right) \tag{12}$$

Where $z_j, \mu_{\mathrm{j}}^{\mathrm{c}}, \sigma_{\mathrm{j}}^{\mathrm{c}}$ are found via:

$$\begin{gathered} \mathcal{N}(\mathrm{a}, \mathrm{A}) . \mathcal{N}(\mathrm{b}, \mathrm{B}) = z . \mathcal{N}(\mathrm{c}, \mathrm{C}) \\ \mathrm{C} = \left(\mathrm{A}^{-1} + \mathrm{B}^{-1}\right)^{-1} \\ \mathrm{c} = \mathrm{CA}^{-1}\mathrm{a} + \mathrm{CB}^{-1}\mathrm{b} \\ z = \mathrm{z}_0 \exp\left[-0.5\left(\mathrm{a}^{\mathrm{T}}\mathrm{A}^{-1}\mathrm{a} + \mathrm{b}^{\mathrm{T}}\mathrm{B}^{-1}\mathrm{b} - \mathrm{c}^{\mathrm{T}}\mathrm{C}^{-1}\mathrm{c}\right)\right] \\ \mathrm{z}_0 = (2\pi)^{-2/\mathrm{D}} |\mathrm{C}|^{1/2} |\mathrm{A}|^{-1/2} |\mathrm{B}|^{-1/2} \end{gathered} \tag{13}$$

We approximate the sum (12) with a single Gaussian:

$$\mathrm{p}\left(v_{\mathrm{t,i}} \mid \mathrm{D}_{\mathrm{t}}\right) \propto \mathcal{N}\left(\mu_{\mathrm{t,i}}, \Sigma_{\mathrm{t,i}}\right) \tag{14}$$

and also calculate a confidence score for the estimate: $k_{t,i}$. The confidence score will be thresholded to identify the observations which are most likely to belong to a constant velocity trajectory. We will investigate two different approximations. The first is inspired by probabilistic data association (PDA) and the second by nearest neighbor data association (NN):

RVF-PDA

$$\begin{gathered} \mu_{\mathrm{t,i}} = \frac{1}{\sum_{j=1}^{N(t-1)} z_j} \sum_{\mathrm{j=1}}^{\mathrm{N(t-1)}} \mathrm{z_j} \mu_{\mathrm{i,j}}^{\mathrm{c}} \\ \Sigma_{\mathrm{t,i}} = \frac{1}{\sum_{j=1}^{N(t-1)} z_j} \sum_{\mathrm{j=1}}^{\mathrm{N(t-1)}} \mathrm{z_j} \Sigma_{\mathrm{i,j}}^{\mathrm{c}} \\ k_{t,i} = \sum_{j=1}^{N(t-1)} z_j \end{gathered} \tag{15}$$

RVF-NN

$$\begin{gathered} \mu_{\mathrm{t,i}} = argmax_{\mathrm{z_j}} \left\{ \mu_{\mathrm{i,j}}^{\mathrm{c}} \right\}_1^{\mathrm{N(t-1)}} \\ \Sigma_{\mathrm{t,i}} = argmax_{\mathrm{z_j}} \left\{ \Sigma_{\mathrm{i,j}}^{\mathrm{c}} \right\}_1^{\mathrm{N(t-1)}} \\ k_{t,i} = \max \left\{ z_j \right\}_1^{N(t-1)} \end{gathered} \tag{16}$$

2.3 Parameters and Implementation

The free parameters for the RVF are $\sum_P, \sum_0$, and μ_0. $\sum_0$ and μ_0 specify the prior velocity estimate for the observations. These estimates are used for the observations in the first time step. The prior estimate could also be used for observations in other time steps. This leads to an additional term in (11) which captures the possibility that observations are not associated with any trajectories. This additional term also provides a way to include any prior information about the observations that may be available. For example, observations may come with velocity estimates from other sources. In our experiments we did not have this additional information, and $\sum_0$ was set sufficiently large that the contribution of the additional term was negligible.

A significant parameter for the algorithm is the covariance of the likelihood function: $\sum_P$ (which we call RVF variance). In the simplest case we implement a symmetric Gaussian, in which case the covariance matrix is defined with a single scalar:

$$\sum_P = \begin{bmatrix} \sigma_P & 0 \\ 0 & \sigma_P \end{bmatrix} \tag{17}$$

An alternative, which might work better in some applications, is to replace the symmetric Gaussian with an asymmetric Gaussian, where the major axis is aligned with the direction of $d_{i,j}$ and whose aspect ratio is proportional to the magnitude of $d_{i,j}$.When the magnitude is zero, the ratio is one, and distribution is symmetric. This type of noise model is useful when high speed objects have more uncertainty in their speed than their direction. In this paper we restrict ourselves to the symmetric noise model (17), in which case (13) can be implemented with many less operations. Pseudo code for this implementation is provided in Table 1.

Table.1. Pseudo code for the algorithm in the case of the symmetric Gaussian.

```
Given $\sigma_P$, $\mu_0$, $\sigma_0$

FOR each observation i in frame 0                 % Setup initial conditions
   $\mu_{0,i} = \mu_0$
   $\sigma_{0,i} = \sigma_0$
   $w_{0,i} = 0$
END FOR

FOR frame $t = 1$ to $\infty$                      % Run the filter forever
    FOR each observation i in frame $t$
        FOR each observation j in frame $t-1$
           $d_{i,j} = x_{t,i} - x_{t-1,j}$
           $\sigma_c = \sigma_P * \sigma_{t-1,j} / (\sigma_P + \sigma_{t-1,j})$
           $\mu_c = {}^{\sigma_c}/_{\sigma_P} * d_{i,j} + {}^{\sigma_c}/_{\sigma_{t-1,j}} * \mu_{t-1,j}$
           $w_c = 1 / (\sigma_P + \sigma_{t-1,j}) \exp(-0.5*(\|d_{i,j}\|/\sigma_P + \|\mu_{t-1,j}\|/\sigma_{t-1,j} - \|\mu_c\|/\sigma_c))$
           $\mu_{t,i}$ += $w_c * \mu_c$
           $\sigma_{t,i}$ += $w_c * \sigma_c$
           $w_t$ += $w_c$
        END FOR
        %  RVF - PDA                                          RVF - NN
        $\mu_{t,i} = \mu_{t,i} / w_t$                 or      $\mu_c$ with largest $w_c$
        $\sigma_{t,i} = \sigma_{t,i} / w_t$                 or      $\sigma_c$ with largest $w_c$
        $w_{t,i} = w_t$                         or      largest $w_c$
    END FOR
    Flag observations with $w_{t,i}$ > threshold
END FOR
```

2.4 Relationship to Other Methods

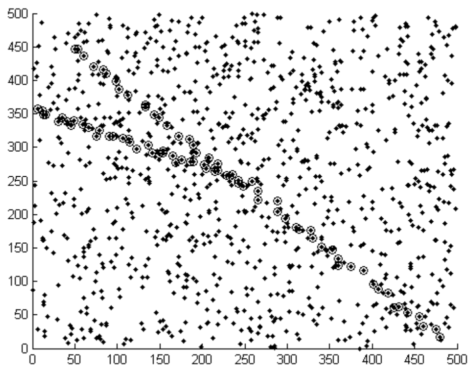
The RVF has some similarities to the Hough transform in that pattern parameters, such as velocity, are estimated from independent and identically distributed observations (see Stevens[1] for a probabilistic perspective on the Hough transform). However, these approaches are typically applied offline over a much longer temporal window and they use a single global transformation[2]. Our algorithm is online and local.

The RVF explicitly trys to associate observations. The data association problem in the RVF is also encountered in tracking. Some tracking methods assume the number of objects is known and then use a variety of techniques to associate observations with targets[3]. Other methods include the number of objects as part of the estimation problem[4]. In this context, the RVF appears to be most similar to Probabilistic Data Association[5]. However, in the RVF there are no dynamics and no state variables (such as tracks) that are persistent across time.

The RVF addresses the same combinatorial problem for data association as multi-target tracking algorithms. However, one advantage of the RVF is that it solves a smaller problem than traditional tracking algorithms. In the applications we have described, a typical tracking algorithm would estimate both the position and velocity of each object, leading to a 4-dimensional problem. In comparison, the RVF only solves a 2-dimensional problem and hence it is possible that more complex parameterizations can be used. For example, we could use 2-d histograms to represent the distributions. Due to the rapid increase in histogram size with dimension, this approach is often not possible with tracking algorithms[6].

3. PARAMETER INVESTIGATION

We performed a number of experiments with synthetic data to investigate the various aspects of the RVF. We randomly initialize up to five linear trajectories in the first frame. We randomly choose the start position and velocity for each trajectory: the position is within 500 by 500 pixels and the velocity is between -10 and 10 in both horizontal and vertical components. We propagate the trajectories for 50 frames and subject the positions in each frame to additive Gaussian noise, which we call observation noise. For each frame we also randomly add 20 additional points as clutter. We apply the RVF to the sequence, and threshold the weight $w_{t,i}$ to determine if the observation is part of a trajectory. Figure 1 provides an example of a typical input and output from an experiment accumulated over 50 frames.



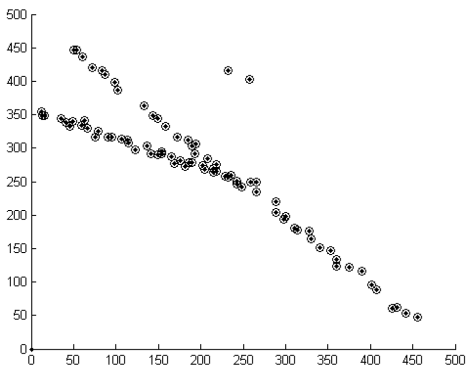


Fig. 1a. Accumulated observations over 50 frames. The observations generated by the synthetic trajectories are circled. 1b. The observations that were calculated by the RVF to be above threshold.

For our first experiment we investigate how detection performance varies with the observation noise and the RVF noise. We set μ_0=[0 0] and σ_0=1500. We increased the observation noise variance from 0 through to 12. For each value of observation noise, we run the RVF with σ_P = 0.1, 1, 5, 10, 50, 100, 500, 1000 ten times and average the ROC curves. In figure 2 we show the value of σ_P with the best detection rate for various numbers of false alarms. It is clear that the RVF variance is related to the observation variance but not in a simple way. We hypothesize that the RVF variance is capturing two different things. The first is our estimate for the observation variance, and the second is the weighting between old observations and new data. For example, we could introduce a second parameter α to implement weighting between old and new observations:

$$p(v_{t,i}|D_t) \propto \sum_{j=1}^{N(t-1)} \left(p(d_{i,j}|v)\right)^{\alpha} \left(p(v_{t-1,j}|D_{t-1})\right)^{1-\alpha}. \tag{18}$$

However, when we make our Gaussian approximation (14) we are renormalizing between each iteration and this means α simply scales the value of σ_P. This has the desired effect: the higher the variance σ_P, the less weight we place on the new observation, but it also means it is difficult to relate σ_P to the observation noise, as illustrated in Figure 2a.

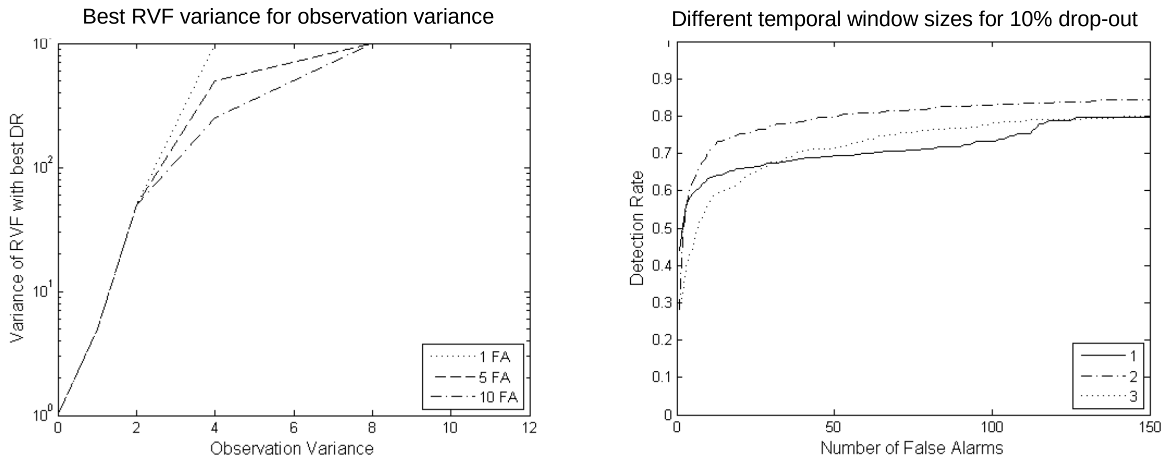


Fig. 2a. The relationship between RVF-NN variance and observation noise. 2b. Performance of RVF-NN with different temporal window sizes.

Another free parameter in the algorithm is the temporal window size. In our derivation of the RVF we only considered associations in the previous time step. However, in practice observations are prone to drop-out and it is possible that an observation should be associated with samples further back in time:

$$p(v_{t,i}|D_t) \propto \sum_{w=1}^{W} \sum_{j=1}^{N(t-1)} p(d_{i,j}|v)p(v_{t-w,j}|D_{t-w}). \tag{19}$$

We investigate the effect of observation drop-out and temporal window size in a second experiment. We use an observation variance of 2 and randomly removed samples with probability 0.1. We set the RVF variance at 50 and the results for different window sizes, averaged over 10 runs, and shown in Figure 2b. The first observation is that drop-out significantly reduces the performance of the window size 1 algorithm (the performance with no drop-out can be seen in Figure 3). Performance was improved with a temporal window size of 2, but further increases had a negative effect. We also note that the computational cost of the algorithm increases rapidly with temporal window size. If we consider all possible associations in each frame then the window length 2 is twice as expensive as the window length 1 implementation. In practice, it is typical to only include associations within a certain distance, which is determined by the maximum speed of targets.

4. PERFORMANCE COMPARISON

We compare the RVF to multi-target tracking algorithms using the same synthetic test set that was used in Section 3. Multi-target tracking is a difficult problem and many different methods have been proposed. In this paper we implement a fairly simple algorithm which uses multiple independent Kalman filters and greedy data association at each time step. The algorithms are summarized in Table 2. Note that on the first frame, we start with zero Kalman filters and so the code begins with step 6.

Table.2. Outline of multi-target tracking algorithms used in the experiments.

```
FOR T = startTime to endTime
    1. Predict - All Kalman filters predict position for current frame.
    2. Associate – Assign observations to Kalman filters.
        a. PDA
            i. Calculate observation as a weighted combination of actual observations
            ii. Weight of observation is proportional to the probability that the observation
                was generated by the track.
        b. NN
            i. Uses the highest probability observation as calculated in PDA.
    3. Likelihood - Calculate the probability that the assigned observation was generated by the
       track.
    4. Identify – Flag observations whose probabilities are lower than a threshold (0.001).
    5. Update – Update the Kalman filters using assigned observations.
    6. Initialize – Initialize new Kalman filters on observations that were flagged in step 4.
```

After we have applied the multi-target tracking algorithm to the dataset, we post-process the tracks to measure performance. For each track we calculate the average likelihood over the length of the track and then flag tracks that have an average likelihood above a certain threshold (adjusted to generate the ROC curves). Finally, to calculate the error we collect all the updated locations (step 5) for all flagged tracks and compare the point set to the ground truth. Track locations within 10 pixels of a moving object location are considered a detection. Multiple detections per point are not penalized. All other track locations are considered false alarms.

We compare the detection performance for the four methods:

1. RVF-PDA: Recurrent Velocity Filter using (15)
2. RVF-NN: Recurrent Velocity Filter using (16)
3. KF-PDA: Kalman filters with weighted sum of observations.
4. KF-NN: Kalman filters with nearest neighbor observations.

There is zero process noise in the synthetic data sets, and we set the process noise of the Kalman filter to 0.1. We performed four experiments with different amounts of observation noise. This value is also used for the observation noise of the Kalman filters. The Kalman filter parameters are therefore very close to optimal for the data set. The RVF variance was held constant across all 4 experiments at a value of 150. The results in figure 3 are averaged over 10 experiments.

The RVF-NN consistently outperforms the RVF-PDA. At small noise variance the Kalman filter approaches outperform the RVF approach. This is partly due to the fact that the RVF missed detections early in the sequence. That is, it takes several frames for the confidence values to reach above threshold. When the Kalman filters succeed, they typically include the complete track. At higher levels of noise variance we observe the RVF performed extremely robustly, outperforming the Kalman filter implementations.

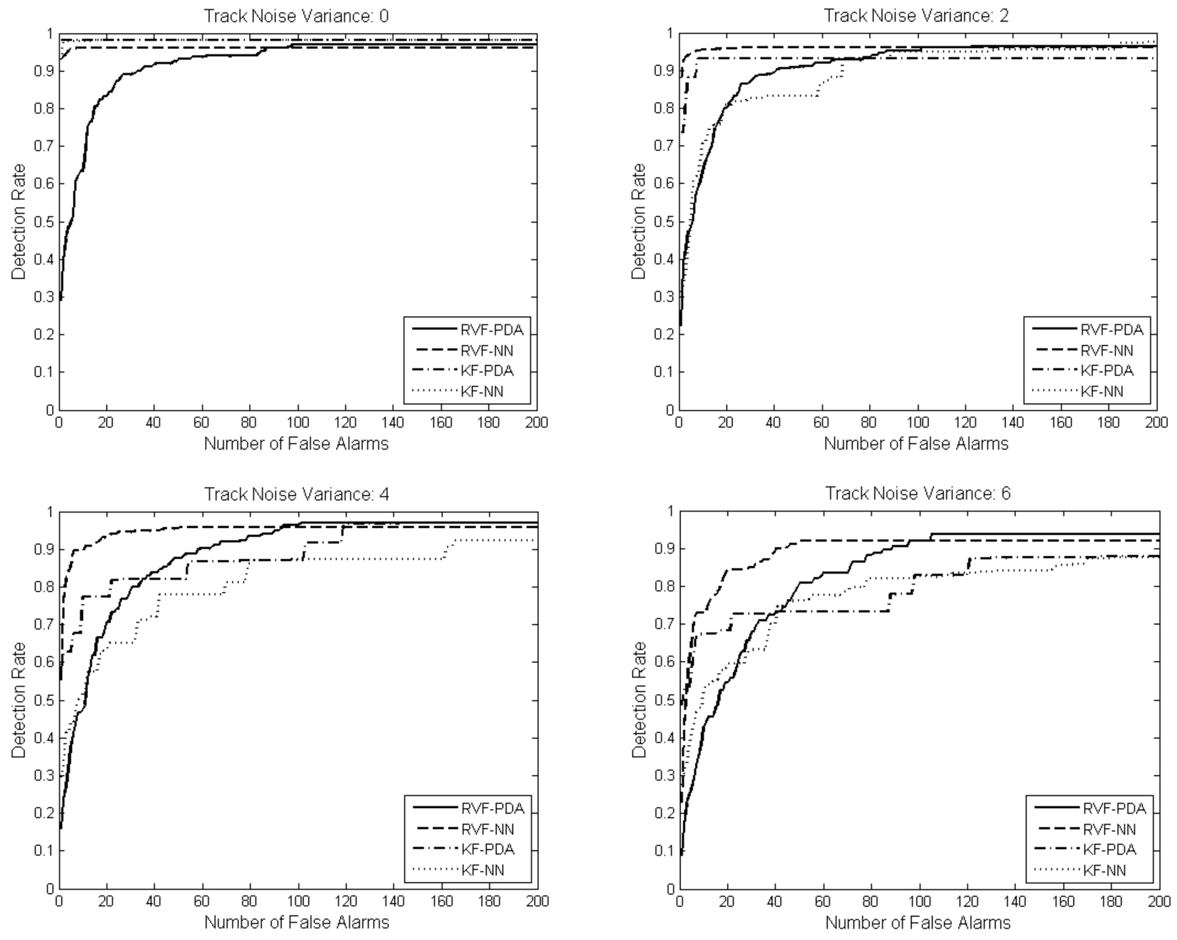


Fig. 3. Performance comparison between RVF and Kalman filter implementations for various amounts of observation noise.

5. REAL WORLD EXAMPLE

For illustrative purposes, we applied the RVF-NN algorithm to a wide area video data set. The sequence is 2000 by 2000 pixels and 100 frames long. The stabilization is not ideal, and the accumulated moving object detection hits are shown in Figure 4a. There are roughly 150 hits per frame. An un-optimized C implementation of the RVF-NN running on a 3GHz Xeon processor took approximately 1ms per frame. We timed the RVF-NN processing 1000 points per frame at 100ms per frame. The accumulated observations that were above threshold according to the RVF are shown in Figure 4b. Ground truth was not available for this dataset, but the results show several distinct line segments which correspond to vehicles moving in the scene. The number of false alarms was typically less than 5 per frame. Given this much reduced dataset, a simple nearest neighbor tracking system achieved reasonable results.

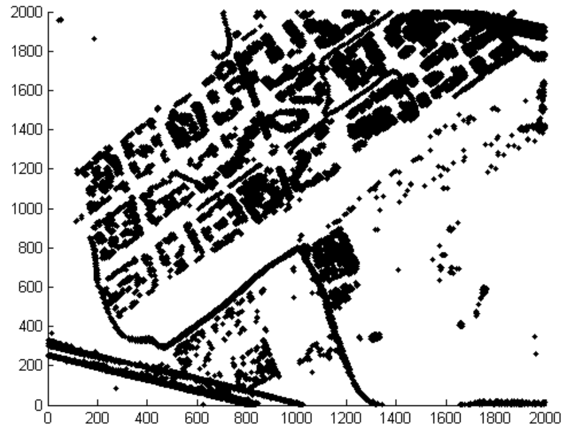
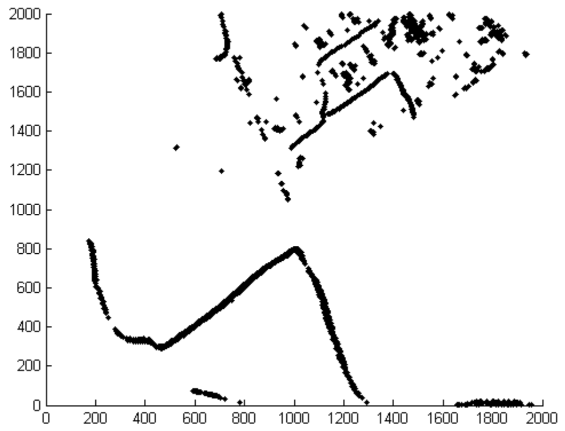



Fig. 4a. Accumulated hits from moving object detection. The double straight line on the bottom left is due to edge artifacts.
4b. Accumulated hits from RVF-NN.

6. CONCLUSION

We have presented a novel method to assign velocity attributes and confidence scores to noisy position observations. The method has very few parameters, and the default values appear to work well in a wide variety of situations. The method is simple to implement and extremely fast to execute. In future work we plan to investigate the performance of non-symmetric Gaussian parameters, and also evaluate how the RVF should be integrated within a tracking system. One option is to use the RVF as a pre-filter and only pass observations to the tracking system that are above threshold. A second option would be to use the velocity estimate to augment the observation vector and supply the tracking system with the covariance matrix that is predicted for each observation. This does not reduce the number of observations, but depending on what multi-target tracking system is used, it may help reduce the number of hypotheses that need to be evaluated.

REFERENCES

[1] Stevens, R.S., "Probabilistic approach to the Hough transform", Image and Vision Computing, 9(1), 66-71, (1991).

[2] Carlson, B.D., E.D. Evans, and S.E. Wilson, "Search Radar Detection and Track With the Hough Transform, Part I-III", IEEE Trans on Aerospace and Electronic Systems, 30(1), 102-124, (1994).

[3] Hue, C., J.P. Le Cadre, and P. Perez, "Tracking multiple objects with particle filtering", Technical Report, IRISA, (2000).

[4] Sidenbladh, H. and S. Wirkander, "Tracking random sets of vehicles in terrain", IEEE Workshop on Multi-Object Tracking, Madison, WI, (2003).

[5] Bar-Shalom, Y., and Tse, E., "Tracking in a cluttered environment with probabilistic data association", Automatica, 11, 451-460, (1975).

[6] Stone, L.D., C.A. Barlow, and T.L. Corwin, "Bayesian multiple target tracking", Artech House Radar Library, Artech House Publishers, Boston, (1999).