

A Recurrent Velocity Filter for Detecting Large Numbers of Moving Objects

Reid Porter, Ed Rosten, Rohan Loveland

Abstract

We present a method for estimating the velocities of 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 objects 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 points are moving objects, but many are not. Typical feature extraction might be frame differencing, or a target recognition system, e.g., a generic car detector. For each candidate, and at each time step, our algorithm outputs a velocity histogram, from which trajectory information can then be derived. In this paper we investigate the free parameters of the algorithm, assess its computational requirements, and evaluate its performance on both synthetic and real-world geographically referenced video data.

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. The combination of high clutter and poor specificity means finding reliable correspondences between frames is challenging. 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, 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. Finally, 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.

In many geo-spatial video applications, e.g., town planning, we are interested in accumulating route information from a large, unknown, number of objects over a long period of time. Robust moving object detection can play two roles in this problem.

The first is as a preprocessor to a tracking system. Many multi-target tracking systems have been proposed [7] which could be applicable to geo-spatial video. Some of these 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], [8]. How these approaches perform at low frame rates, and scale to thousands of objects in high clutter, is an open question. Moving object detection can help solve this problem by filtering through 10's of thousands of detections and giving high weight to locations most likely to be moving objects.

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 finding good correspondences

In Section 2 we describe the proposed approach and include a brief discussion of how our filter compares to other methods. In section 3 we use synthetic data to investigate the filter's parameter choices. We then turn our attention to a real-world geo-spatial video dataset in Section 4. This section also describes a prototype feature extraction system which is required to generate candidate locations from data.

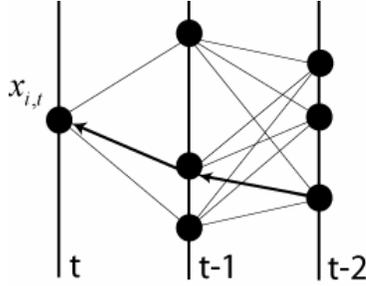


Figure 1: Objects are assumed to travel along discrete trajectories through points in each frame.

2. A Recurrent Velocity Filter

At each time t (typically associated with a frame), we are given a set of locations:

$$X_t = \{x_{i,t}\}_{i=1..N(t)} \quad (1)$$

where $x_{i,t} \in \mathbb{R}^2$, and $N(t)$ is the number of points, some of which are moving objects, but many are not. We assume one, or more, objects pass through point $x_{i,t}$, and we would like to estimate a velocity distribution at this point, given the data we have seen so far. We assume that objects follow a discrete trajectory through points in each frame, as shown in Figure 1. We also assume that the velocity distribution can be estimated from a finite temporal window. This leads us to our basic algorithm:

$$p(v_{i,t}) = \eta (p(v_{i,0}) + K \sum_{s=1}^T \sum_{j \in W_t(i)} p(x_{i,t} \equiv x_{j,t-1}) p(v | x_{i,t}, x_{j,t-s}) p(v_{j,t-s})) \quad (2)$$

In equation 2, η is a normalization term. The first probability on the right hand side, $p(v_{i,0})$, accounts for new observations and the possibility that the point does not belong to a trajectory. In this paper it is the uniform distribution. The first term within the summation, is the probability that point $x_{i,t}$ corresponds to point $x_{j,t-1}$. We assume there is no correspondence information and so this probability is constant. However, we do restrict which correspondences are considered by bounding the maximum velocity. This constraint is implemented by $W_t(i)$, which includes all points within a certain window of point i for a given time step. The second term in the summation is a local velocity estimate given that $x_{i,t}$ corresponds to point $x_{j,t-1}$. This distribution must be chosen for the problem, and typically depends on the distance between the points, $x_{i,t_1} - x_{j,t_2}$, which is invariant to translation. We discuss our choices for this distribution in Section 2.2. The third term within the summation is the prior estimate for velocity given the correspondence. This velocity was calculated in the previously frame at point $x_{j,t-1}$. At time $t = 0$ this distribution is assumed uniform.

We implement equation (2) by representing the probability distributions as histograms. The storage requirements are the primary concern, and are proportional to $TN(t)V_{\max}^2$. This may be expensive for some applications that require extremely precise velocity estimation, but in many cases, the cost is comparable to other point matching algorithms used to find correspondences. For example, a maximum velocity of 16 pixels would mean our algorithm has similar storage requirements to matching 128 element SIFT descriptors [6].

Also of computational concern is how the neighborhood radius increases with temporal window size T . That is, the number of potential neighbors at frame $t-s$ is on the order $(sV_{\max})^2$ which means the algorithm is only practical for small temporal windows. There are a number of ways that the local velocity estimate could depend on the temporal window size. In this paper, we assume constant velocity, and linearly scale velocity estimates by s .

Typically we post-process the distributions associated with each point to extract information relevant to an application. In the experiments in this paper, we use the mode of the distribution, as a track detector. We also use the mean of the distribution as an estimate for the velocity.

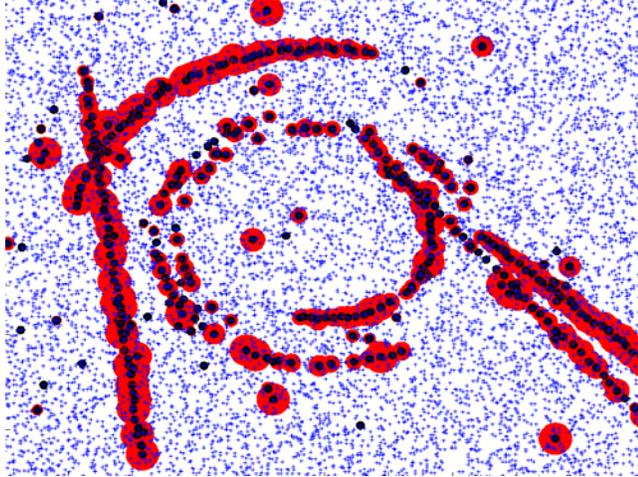


Figure 3: Example of synthetic data and filter output accumulated over 50 frames. Points above threshold are marked in black. The size of the circular surround indicates how much points are above threshold.

2.1. Relationship to Other Methods

The recurrent velocity filter has some similarities to the standard Bayesian tracking recursion [9]. However in Equation 2 we can see that our system is stateless. Our approach also has some similarity with the Hough transform track detection algorithms [1]. However, these algorithms typically involve a much longer temporal window and apply a single global transformation. Our algorithm is online and local. Finally, our algorithm implements probabilistic voting and therefore has some resemblance to other voting methods like tensor voting [5]. Typically, these strategies (sensibly) parameterize votes and do not propagate complete histograms.

2.2. Local Velocity Distributions

We investigate three different distributions for $p(v | x_{i,t_1} - x_{j,t_2})$ which are illustrated in Figure 2. The first is a symmetric Gaussian, where the variance reflects our estimate of velocity variability from one frame to the next. The second is an asymmetric Gaussian, which is orientated to the difference direction, and whose ratio of variances is made linearly proportional to the magnitude of the difference. At zero speed, the ratio is 1. At the maximum speed the ratio is 0.1. This distribution is motivated by vehicles traveling on roads, where fast moving vehicles have less chance of changing direction than slow moving vehicles. The third distribution takes this a step further and weights directional variations in proportion to velocity. This was implemented in polar coordinates as:

$$p(v) \propto e^{-\theta^2/\omega} e^{-r^2/\sigma} \quad (3)$$

This distribution gives higher probability to velocities which are consistent with a point moving along a curve.

3. Synthetic Experiments

We performed a number of experiments with synthetic data to investigate the various parameters of the filter. For each frame, we uniformly sample 200 points from a 500 by 500 spatial grid. We then randomly generate a target trajectory. This trajectory can be either linear or circular, and any speed from 4 to 16 pixels (our maximum velocity). The track is then subject to jitter by randomly perturbing the point by up to 2 pixels. At each frame we also include a probability of trajectory drop out, in which case, the track point is removed.

In our typical experiment, we generate a data set and apply the tracking algorithm in Equation 2. At each point, and for each frame, we record the highest probability in the distribution. We threshold this value to determine if the point belongs to a track or not. A typical output from the filter is shown in Figure 3. Points above threshold are marked in black. How much they are above threshold is indicated by the size of the circular surround. It can be seen that the filter strengthens the

seven trajectories despite high levels of clutter (trajectories usually cannot be detected by visual inspection). Also, it can be seen that, in this case, linear trajectories accumulate greater weight than curved trajectories, which is consistent with the local velocity distribution that was used.

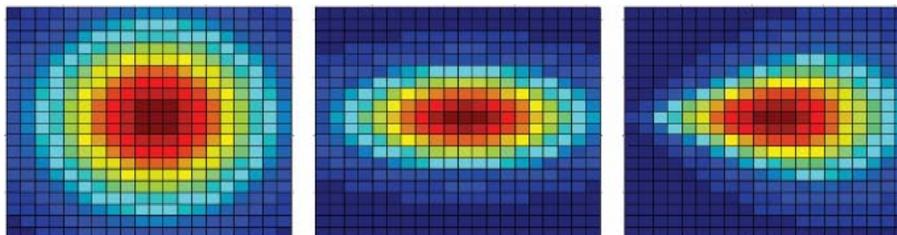


Figure 2: (Left) Symmetric Gaussian, (Middle) Asymmetric Gaussian and (Right) Co-circular distributions.

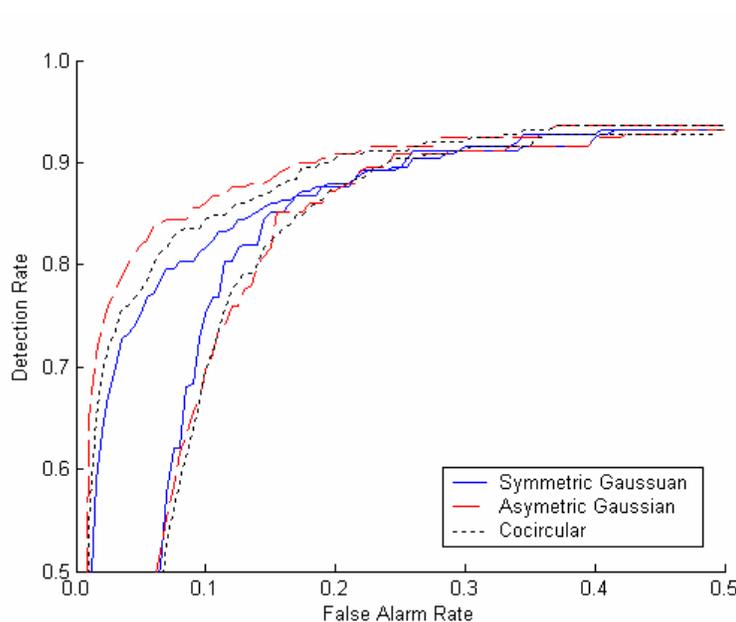


Figure 4: Performance comparison of local velocity distributions. (Top) Filters applied to linear trajectories and (Bottom) filters applied to circular trajectories.

Our first experiment was to compare the performance of the three different distributions shown in Figure 2. Jitter was at 1 pixel, and the drop-out probability was 0.2, and results were averaged over 10 experiments. We threshold the highest probability recorded at each point at multiple values to produce the Receiver Operating Characteristic curves shown in Figure 4. The two sets of curves are associated with the two different types of trajectory. The top three curves are for linear trajectories, and the bottom three curves are generated from circular trajectories. These results confirm the problem specific nature of the local velocity distributions. The asymmetric distributions with speed-proportional aspect ratios outperformed the symmetric distribution in the first experiment since they are tuned to linear trajectories. Likewise the symmetric distribution is a better match for the circular trajectories where direction variation does not depend on speed. The co-circular distribution was not the best distribution for either type of trajectory. However, all synthetic trajectories had constant speed, and therefore further experiments are required to evaluate its strengths in more detail.

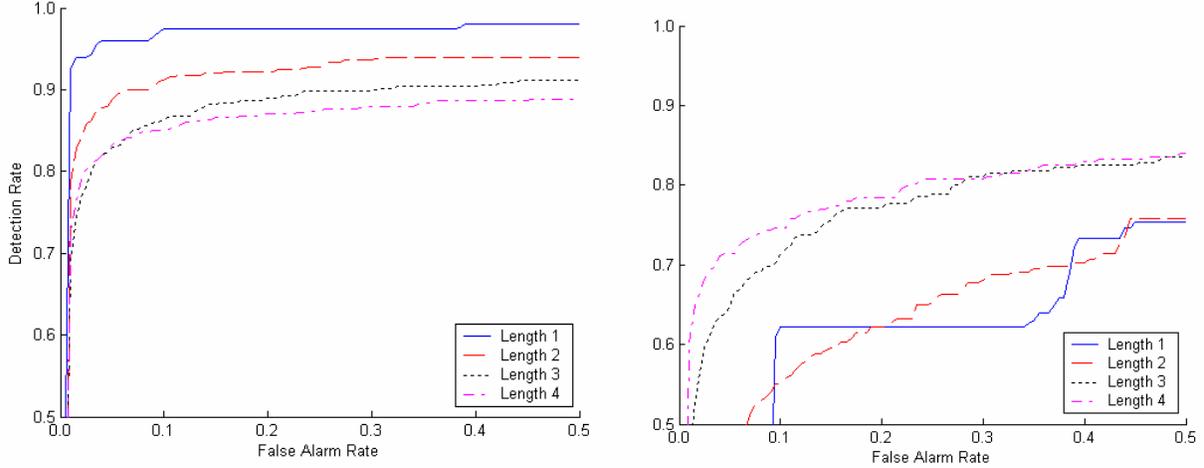


Figure 5: Performance of asymmetric Gaussian filter for varying temporal window length and drop-out probability of 0.0 (left) 0.6 (right)

The temporal window size T , has a large impact on the filter's computational complexity. It is therefore important to evaluate the benefit of increasing the window size. In the second set of experiments we investigated the relationship between this window size and target drop out rate. Since we assumed constant velocity, we used linear trajectories and the asymmetric Gaussian distribution for a local velocity distribution. We varied the drop out probability from 0.0 to 0.6. Figure 5 shows the results at these two extremes. We hypothesize the poorer performance of large windows in Figure 5 (top), compared to small windows, is due to the increased track / clutter ratio which leads to longer convergence times. For the high drop-out probability, we see in figure 5 (bottom) that the performance is improved with the larger window lengths as we would expect.

We also investigated the relationship between the constant K and the drop out rate. We observed a similar trend to Figure 5. For low, or zero drop-out probability, small values of K (1-5) had slightly better performance. However for any significant drop-out probability the larger values of K (10-100) produced more reliable results. In both cases, the performance difference was minimal compared to the effect of the temporal window size.

4. Real World Experiments

We investigated the practical utility of our approach using a small collection (45 frames) of geo-referenced images over a busy freeway intersection. An example frame is shown in Figure 6. The resolution of the imagery is approximately 0.5 meter per pixel at 1 frame per second. The geo-projection, registration, and calibration were applied in real time, and as a result, the data products are not ideal for automated analysis. As shown in Figure 7 (left) vehicles suffer from large variations in illumination and ground sample distance. In Figure 7 (right) we see an example of a difference image, where poor registration has led to significant motion clutter.

4.1. Preprocessing and Point Extraction

To obtain a discrete set of candidate vehicles for each frame requires significant pre-processing. This is a challenging research problem unto itself, and there are many different ways to approach the problem. In this paper we will briefly describe our initial approach.

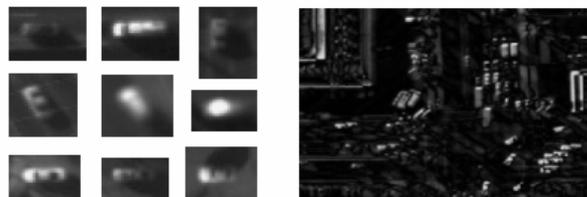


Figure 7: (Left) Examples of vehicles. (Right) Example of frame differencing.



Figure 6: A typical frame in the geo-spatial video dataset.

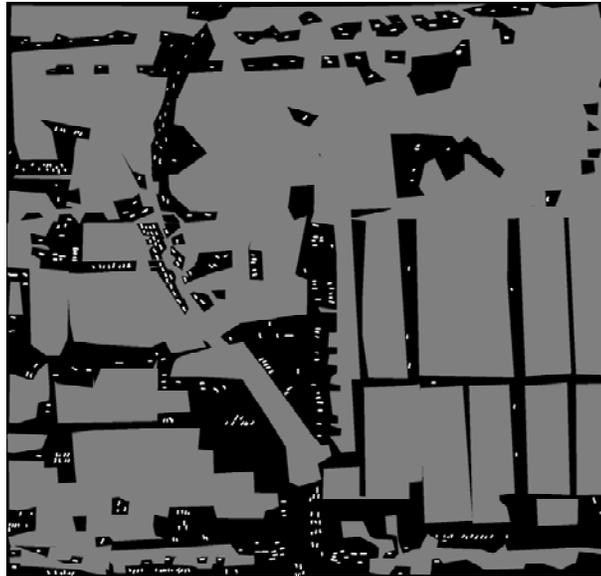


Figure 8: The training data used to optimize the car detector. Pixels are labeled vehicle (white), background (grey) or left unlabelled (black).

The main part of the algorithm is a generic car detector which can be applied directly to the image data. This is implemented with a sequence of morphological image operations, which are optimized for the problem in offline training. Figure 8 shows the training data that we used for this optimization. White represents pixels assigned to the target class, grey represents pixels assigned to the background, and black pixels are left undecided and do not contribute to the optimization. The training data contains both stationary and moving vehicles as the target class.

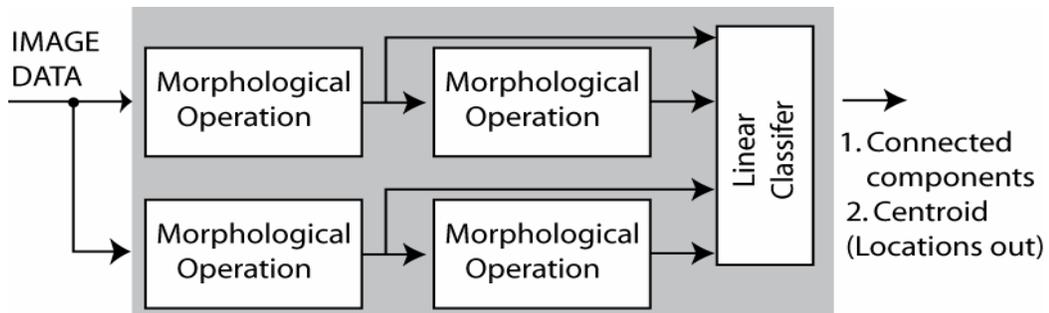


Figure 9: The car detector used as a preprocessor to produce target locations.

The structure of the detector is shown in Figure 9. It comprises two, two-stage morphological operators and a linear classifier. The parameter space for each morphological operation is summarized in Table 1. All functions exploit reconstruction [10]. When the rectangle structuring element is used, we use the maximum response from 8 rotations of the structuring element for Opening filters, and the minimum response for Closing filters. This leads to symmetric filters for highly asymmetric linear image features, which appear often in the scene due to roads and buildings.

The morphological pipeline parameters are optimized using a steady-state Genetic Algorithm. For each candidate set of parameters: 1) the pipeline is applied to the data, 2) the linear classifier is found using Fisher's Linear Discriminant, 3) a threshold is found that minimizes misclassification error through exhaustive search, and 4) fitness is assigned based on an object-centric error measure which we will describe shortly. For the Fisher Discriminant and the threshold, the training data is weighted so that the two class errors (target and background) have equal weight. We evolved populations of 25 chromosomes, for 25 generations. The probabilities for mutation (for each parameter in Table 1), and uniform crossover (which randomly swaps complete operations), were both 0.15.

The training data of figure 8 was produced by hand, and target vehicles are specified with a relatively sloppy markup. Each vehicle is designated by a swath of pixels which may cover only part of the vehicle, and may, or may not also contain background. To help minimize the effect of inconsistent markup, the final fitness assigned to a candidate is not a pixel-based error, but an object-based misclassification error. We apply connected components to the training data and require that the detector only to correctly predict one of the pixels within each swath. The error for the background class remains the same as before and is based on pixels. This type of error is also found in many multiple instance learning problems [2].

The final stage of our preprocessor finds the connected components of the detector output and calculates the mean location for each component. Since we are interested in detecting moving objects, the final step is to remove points which appear within 4 pixels on two consecutive frames.

Parameter	Possible Values
Function Reconstruction	{Open, Close, TopHat, BlackHat, Area Opening, Area Closing, H-Dome, H-Bowl }
Structuring Element Shape	{Square, Circle, Rectangle}
Window Size	3..21
Offset	0..255 (for H-Dome, H-Bowl)
Aspect Ratio	1..WindowSize (for rectangle)

Table 1: Parameters associated with morphological operation.

4.2. Velocity Filter Results

The fastest vehicles in the scene are on the interstate, where they can travel approximately 200 pixels / frame. We set our maximum velocity at 256 pixels / frame and used histograms with 64 by 64 bins by quantizing the differences, $x_{i,t_1} - x_{j,t_2}$, by a factor of four. We used a temporal window of 2, investigated both symmetric and asymmetric Gaussian distributions, and set $K = 10$.

The initial results, shown in Figure 10, appear promising. In the top of Figure 10 we see the points predicted by the preprocessor as crosses. Points with a high probability from the symmetric Gaussian filter are shown circled. A clearer picture can be seen at the bottom of Figure 10 for the asymmetric Gaussian filter, which also illustrates the velocity predicted by the first moment. The two results are fairly similar, with the asymmetric filter obtaining fewer false alarms, but also fewer detections. We observe that points detected by the symmetric filter, but not detected by the asymmetric filter, are those vehicles which are moving along the curved entry and exit ramps.

At the bottom of figure 10, we observe that the right hand side of the freeway has a large number of detections which appear to be driving on the wrong (right) side of the road. This is due to registration errors in the dataset which shift the freeway by approximately 30 pixels over the 45 frames. Given this, we observe that vehicles entering the scene on the left and right of the freeway do not appear to be detected. We found this was due to the latency associated with accumulating an above threshold probability. Freeway vehicles typically cross half our image in approximately 4 or 5 frames. In larger datasets, we believe this initialization time would be acceptable.

5. Conclusions and Future Work

We have presented a recurrent velocity filter framework for predicting a velocity distribution at a given set of points. The filter allows correspondences to be made between sets of points, based on trajectory smoothness, and also allows trajectory specific prior knowledge to be easily incorporated. The filter was tested on a real-world data set and results appear promising.

Our approach is easily generalized to include correspondence information when it is available. We plan to investigate this possibility as part of our future work.

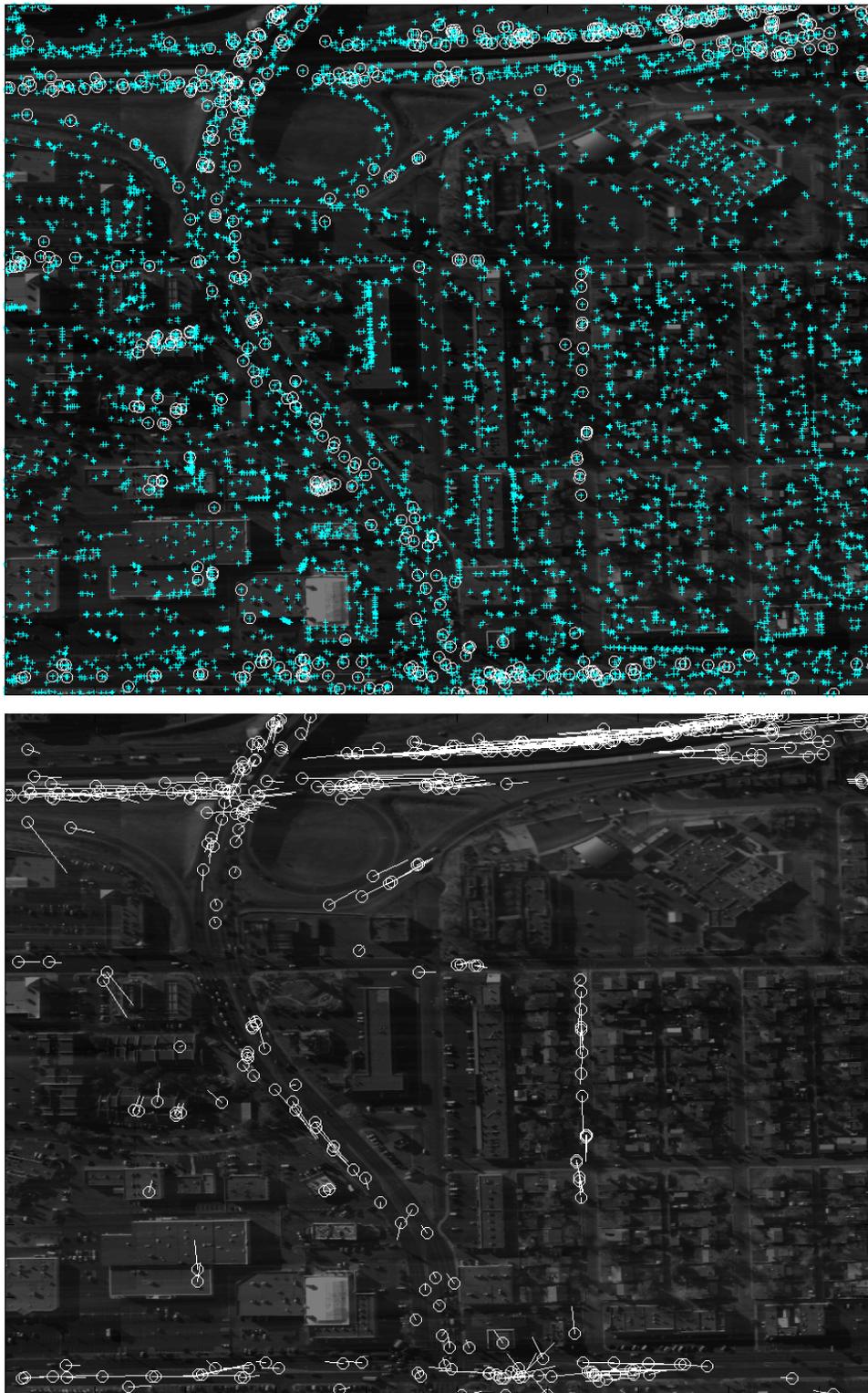


Figure 10: Output images obtained from the filter. (Top) Using the symmetric Gaussian distribution: Cross marks indicate points generated by the car detector. Circles indicate points above threshold after the velocity filter is applied. (Bottom) Using the asymmetric Gaussian distribution: Circles indicate points above threshold and lines indicate predicted velocity.

References

- [1] 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 Svstems,, 1994. **30**(1): p. 102-124.
- [2] Dietterich, T.G., R.H. Lathrop, and T. Lozano-Perez, *Solving the Multiple-Instance Problem with Axis-Parallel Rectangles*. Artificial Intelligence Journal, 1997. **89**.
- [3] Hue, C., J.P. Le Cadre, and P. Perez, *Tracking multiple objects with particle filtering*. . 2000, IRISA,
- [4] Isard, M. and J. MacCormick. *BraMBLe: A Bayesian multiple-blob tracker*. in *IEEE International Conference on Computer Vision*. 2001.
- [5] Kornprobst, P. and G. Medioni. *Tracking segmented objects using tensor voting*. in *IEEE Conference on Computer Vision and Pattern Recognition*. 2000. Hilton Head Island, South Carolina.
- [6] Lowe, D.G., *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, 2004. **60**(2): p. 91-110.
- [7] Pulford, G., *Taxonomy of multiple target tracking methods*. IEE Proceedings-Radar, Sonar and Navigation, 2005. **152**(5): p. 291-304.
- [8] Sidenbladh, H. and S. Wirkander. *Tracking Random Sets of Vehicles in Terrain*. in *IEEE Workshop on Multi-Object Tracking*. 2003. Madison, WI.
- [9] Stone, L.D., C.A. Barlow, and T.L. Corwin, *Bayesian Multiple Target Tracking*. Artech House Radar Library. 1999, Boston: Artech House Publishers.
- [10] Vincent, L., *Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms*. IEEE Transactions on Image Processing, 1993. **2**(2): p. 176-201.