# Tracking Vehicles Through Shadows and Occlusions in Wide-Area Aerial Video

**CHAD AESCHLIMAN**
**JOHNNY PARK**
**AVINASH C. KAK**
Purdue University

**We present a new approach for simultaneous tracking and segmentation of multiple targets in low frame rate aerial video. We focus on building an accurate background model that accounts for both global camera motion and moving objects in the scene. We then apply a probabilistic framework for simultaneous tracking and segmentation that incorporates this background model. By using a background model, we are able to track the object through dramatic appearance changes caused by shadows and lighting changes. Furthermore, the incorporation of segmentation into the tracking algorithm reduces the impact of common tracking problems, such as drift and partial occlusion. Results are shown for the Columbus Large Image Format (CLIF) 2007 data set, demonstrating successful tracking under significant occlusions, target appearance changes, and near similar moving objects.**

Manuscript received July 5, 2012; revised January 1, 2013; released for publication August 27, 2013.

IEEE Log No. T-AES/50/1/944799.

DOI. No. 10.1109/TAES.2013.120404.

Refereeing of this contribution was handled by H. Kwon.

Authors' addresses: C. Aeschliman, J. Park, A. C. Kak, Purdue University, Electrical and Computer Engineering, 465 Northwestern Ave., West Lafayette, IN 47906,
USA E-mail: (kak@purdue.edu).

## I. INTRODUCTION

In this paper, we consider tracking in wide-area aerial video, specifically, the Columbus Large Image Format (CLIF) 2007 data set [1]. Tracking is a critical component of many wide-area surveillance tasks, such as traffic monitoring and automated video indexing.

While tracking is difficult in any real-world scenario, wide-area surveillance video presents some unique challenges. First, the frame rate is very low—typically 1 to 2 frames/s. Because of this, targets can move many times their length and undergo significant appearance changes between frames. Second, wide-area aerial imagery typically suffers from fairly low resolution and blurring. In the CLIF 2007 data set, for example, a vehicle in an image typically has a size of about 15 pixels by 15 pixels. This makes it difficult to distinguish targets based on appearance. Finally, in aerial imagery, the camera is constantly moving; hence, the simplifying assumption of a static background is no longer valid.

### A. Related Work

Much of the tracking work in aerial video has focused on low altitude, high frame rate video, such as the Defense Advanced Research Projects Agency Video Verification of Identity data set [2]. Because of the low altitude, the resolution of targets is high, making the tracking problem much easier. Most of the challenge in tracking in these data sets is in compensating for the aircraft motion [3] and detecting targets of interest [4]. Once these problems are addressed, the actual tracking is straightforward and can be done with generic object trackers, such as the kernel-based object tracking method of Comaniciu et al. [5].

The focus of this paper is on tracking in high altitude, low frame rate video. Under this scenario, tracking becomes much more difficult, and a specialized tracking strategy is required. Reilly et al. consider detection and tracking on the CLIF 2006 data set [6]. This data set has much smaller targets compared with the CLIF 2007 data set, on the order of just a few pixels. Because the targets are too small to do appearance modeling, the authors focus on solving the data association problem between detected vehicles in subsequent frames. As a result, their method is not able to handle significant occlusions, which are common in the CLIF 2007 data set.

Dessauer and Dua developed an optical flow-based tracker for the CLIF 2007 data set [7]. They focused primarily on the best choice of feature for estimating the flow and performing target matching between frames. As with [6], their method does not account for significant target occlusions and hence is limited to vehicles in relatively open areas.

A common issue with the approaches in the existing methods is that they work only for vehicles in open areas with good visibility, such as vehicles on multilane highways. However, for many surveillance applications, vehicles of interest will also be driving on small side streets with shadows and occlusions from trees and

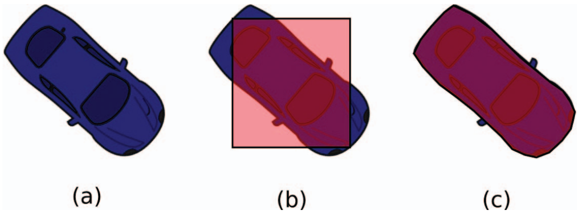


Fig. 1. Comparison of coarse and fine segmentation. Blue pixels represent target with a semitransparent red overlay, indicating segmentation (best viewed in color). (a) Target; (b) coarse segmentation with extra and missing pixels; (c) fine segmentation.

buildings. Our approach addresses this problem by combining tracking and segmentation.

### B. Combined Tracking and Segmentation

Although the problem we are interested in solving is tracking, there has been research over the last two decades showing that combining tracking and segmentation can improve tracking performance. Early approaches focused on segmenting a video sequence based on regions of similar motion [8, 9]. Recent work has focused on appearance-based segmentation using a background model [10, 11].

Solving both the tracking and segmentation problems together has two main advantages. First, tracking with segmentation is more robust against target appearance changes because information in the background can be used to aid tracking, e.g., pixels that do not match the background are more likely to be part of the target regardless of their appearance. This is particularly important for our application because shadows cause the target appearance to change dramatically. Second, by finely segmenting the target from the background, we are able to better update the appearance model of the target compared with coarse segmentation, such as a bounding box. With coarse segmentation, it is inevitable that some background pixels are included in the target model, while some target pixels are excluded (see Fig. 1). The included background pixels cause drifting and possible loss of tracking, while the excluded target pixels make the tracking less robust against partial occlusions.

This paper has two main contributions. First, we show how to build an accurate background model around each target in a low frame rate aerial video, even with a moving background. Second, we apply a modified form of the simultaneous tracking and segmentation method of Aeschliman et al. [10] using this background model and evaluate the results.

## II. CALIBRATING THE CAMERAS IN THE CLIF 2007 DATA SET

The CLIF 2007 data set consists of high-resolution images captured from six cameras simultaneously along with information from a global positioning system/inertial measurement unit (GPS/IMU). Unfortunately, the intrinsic and extrinsic calibration information for the cameras and GPS/IMU is not provided.

Tracking of vehicles within a single camera is possible without knowing the calibration information by operating purely within the image space. However, to hand off targets from one camera to the next, an accurate calibration is needed. Furthermore, by performing an extrinsic calibration to the GPS/IMU, it is possible to track targets in a georeferenced world coordinate system. This makes it possible to enforce meaningful constraints on target motion, e.g., limits on speed and direction changes.

### A. Calibration Overview

Camera calibration is traditionally done by capturing images of a scene with some known structure or pattern and then exploiting this structure to get the calibration [12, 13]. An approach along these lines is taken in [14] to calibrate one camera from the CLIF data set using known world points collected from Google Maps. The downside of this approach is that it is very time consuming to identify and mark a large number of world points, which is required to ensure an accurate calibration.

It is also possible to perform calibration using images of a generic scene, if some properties of the camera itself are known, e.g., the intrinsic parameters are the same for all images, and there is zero skew and unit aspect ratio [15]. We have used this latter approach to calibrate the CLIF data set because it allows us to take advantage of the huge number of images that are available. All necessary information is extracted from the images and GPS/IMU with only minimal human supervision. We used 660 frames from each camera for intrinsic calibration and 110 frames from each camera for extrinsic calibration.

We first calibrate the intrinsic parameters for each camera separately. To do this, we need to determine corresponding points between images taken from different views, i.e., image points that represent the same physical point in the world. To obtain corresponding points, we perform pairwise matching from each frame to nearby frames using the method described in Section II-B. In our experiments, this gave 8–10 million matches from around 8000 image pairs for each camera.

Once we have matching points, we can use the excellent Bundler tool to obtain an initial camera model and three-dimensional structure [16]. Note that Bundler makes some strong assumptions about the camera to simplify the problem. Hence, we further refined the results using a modified version of the Simple Sparse Bundle Adjustment (SSBA) tool, which has a more flexible camera model including lens distortion [17].

Once the intrinsic parameters are known, we repeated the process with some small modifications to compute the extrinsic calibration as well. First, corresponding pairs were found between image pairs both between frames from each camera and across cameras. Around 3300 image pairs from the same camera were matched along with approximately 18 000 pairs taken from different cameras.

After feeding these matches into Bundler, we corrected the result so that the camera centers would all coincide exactly with the GPS/IMU reported location. This was done for two reasons. First, it simplifies the extrinsic calibration somewhat by reducing the number of parameters because only rotation needs to be considered. Second, the actual translation between the GPS/IMU and the cameras is negligible compared with the distance scales in the rest of the system, and so any computed translation would be inaccurate. Once the camera centers had been corrected, we again modified the SSBA tool to carry out a new refinement of just the rotation parameters for the cameras.

The complete calibration results for all cameras are given in Appendix A along with an explanation of the camera model. Fig. 3 shows a georeferenced mosaic of the six cameras from the data set, which is obtained by projecting each image to the ground plane. A Google Maps image, taken from the same region, is also shown for comparison.

### B. Determining Corresponding Pairs in Aerial Images

Corresponding pairs are sets of image points in two images that correspond to the same real-world location. The first step is to find feature points in each image that should be identifiable in the other image as well. We extracted ≈ 3500 speeded-up robust features (SURF) features from each frame [18].

The next step is to determine pairwise matches between the extracted features. We simply matched each feature in the first image to the best match in the other. This approach for determining correspondences is very fast, but results in a huge number of outliers (typically 50%–90% of the matches are incorrect). Fig. 2a shows two frames that are to be matched, while Fig. 2b shows the initial matches based on SURF features. Attempting to calibrate the cameras based on very poor matches such as these would result in a very inaccurate calibration.

Fortunately, we are able to use the fundamental matrix to constrain the set of possible correct matches and thus prune out most of the outliers. We use the well-known random sample consensus (RANSAC) algorithm to do this except that we replace the random sampling with a similarity-based sampling described later in this section, which greatly speeds up the computation, while still giving good results in high-altitude aerial images.

Estimating the fundamental matrix in a fully general sense requires at least seven corresponding pairs [19]. Thus, in each iteration of the RANSAC algorithm, we need to select seven corresponding pairs from the initial matches [20]. The RANSAC algorithm will only give a good solution if at least one of the randomly selected subsets consists entirely of correspondences that agree with the true fundamental matrix between the images. With completely random sampling, the probability of getting at least one sample with all inliers in $N$ trials is given by

$$p = 1 - \left(1 - w^n\right)^N, \qquad (1)$$

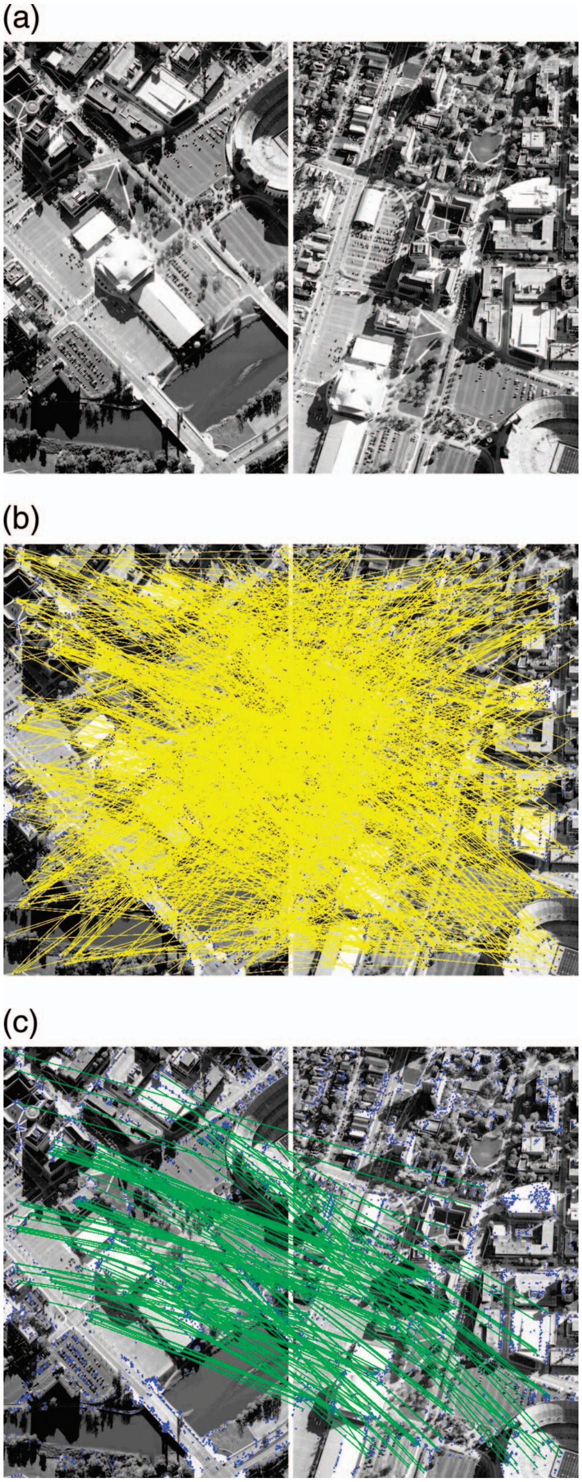


Fig. 2. Frame matching process. Out of 1018 initial matches, only 162 are selected as inliers (16%).

where $w$ is the fraction of inliers and $n = 7$ because seven random pairs are selected. Hence, the number of random samples required to ensure at least one good set with probability $p$ is given by

$$N = \frac{\log(1 - p)}{\log(1 - w^n)}. \qquad (2)$$

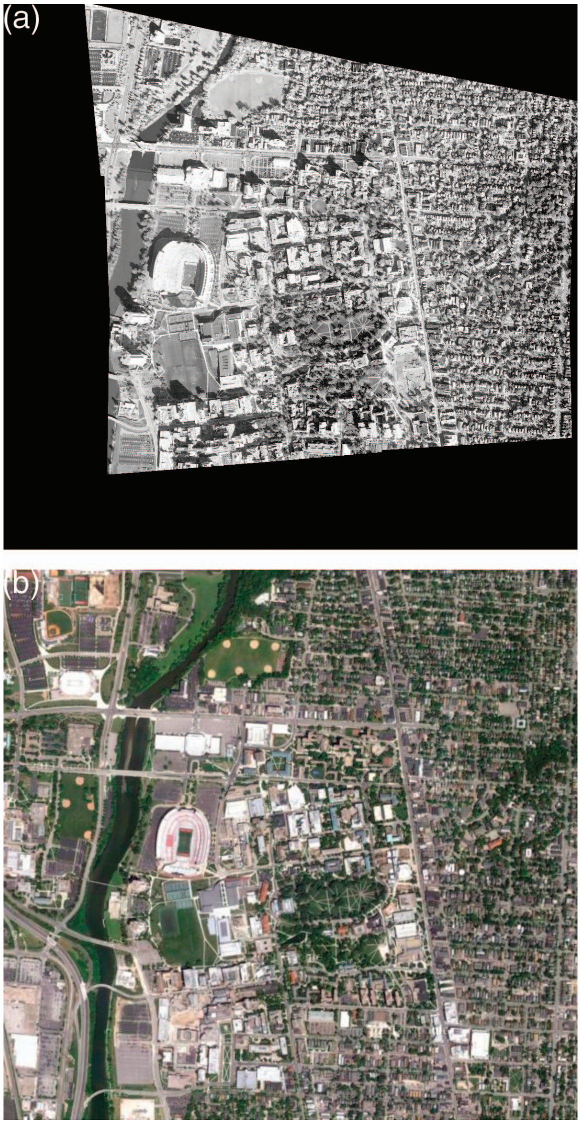


Fig. 3. Top image: georeferenced mosaic of six cameras in CLIF 2007 data set at timestamp 101110. Bottom image: aerial view of same region taken from Google Maps.

TABLE I
$N$ and Processing Time as a Function of $w$ for Matching One Image Using RANSAC for Both Random Sampling and Similarity Sampling

| | Random Sampling | | Similarity Sampling | |
|---|---|---|---|---|
| $w$ | $N$ | Time | $N$ | Time |
| 0.5 | 587 | 0.235 s | 36 | 0.014 s |
| 0.2 | $3.60 \times 10^5$ | 144 s | 689 | 0.276 s |
| 0.1 | $4.61 \times 10^7$ | 5.11 h | $6.87 \times 10^3$ | 2.75 s |
| 0.05 | $5.89 \times 10^9$ | 27.3 d | $8.23 \times 10^4$ | 32.9 s |

*Note*: $p = 0.99$ and $R = 200$, pixels. Processing times are estimated based on a measured time of 0.4 ms per iteration.

The problem is that as $w$ decreases, $N$ increases very rapidly. Columns 2 and 3 of Table I show the required number of iterations and estimated processing time. Note that these times are for a single image pair. With 20% inliers, this method would take approximately 12 wk to process all of the image pairs for intrinsic calibration of the CLIF data set. Matching images between images from different cameras have an average of only 6.5% inliers, and so the time required to match all of the image pairs for extrinsic calibration would be over 6 y!

We can greatly reduce the number of iterations required to obtain a good solution by exploiting the nature of high-altitude aerial imagery. Specifically, we assume that the transformation of a small patch of one image to another is close to a similarity transform, i.e., dominated by a rotation and translation with only minor scaling and perspective effects. Based on this assumption, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are the coordinates of corresponding points in two frames and $\|\hat{\mathbf{x}}_1 - \mathbf{x}_1\| \le R$ for an additional point $\hat{\mathbf{x}}_1$ in frame 1, then $\|\hat{\mathbf{x}}_2 - \mathbf{x}_2\| \le R$, where $\hat{\mathbf{x}}_2$ is the correct correspondence to $\hat{\mathbf{x}}_1$ in frame 2. Based on this property, we use the following procedure to sample the seven-point correspondences needed to estimate the fundamental matrix:

1) Select three-point correspondences completely randomly from the set of all initial correspondences.

2) For each selected correspondence, randomly choose an additional corresponding pair from those pairs that lie within a radius $R$ in **both** images.

3) For one of the selected correspondences, choose one more corresponding pairs as in Step 2 to give seven total correspondences.

Fig. 4 shows these steps for an example pair of images. Note that if either Step 2 or 3 fails because there are no other correspondences satisfying the criteria, we simply go back to Step 1. We call this procedure similarity sampling.

We now turn to the question of how many iterations are required with this sampling methodology. We first need to compute $\bar{p}$, the probability that any one choice of seven samples will consist of all inliers. If the assumption of a local similarity transform holds, then the probability of getting at least one sample with all inliers is given by

$$p = 1 - \left(1 - \hat{w}^4 w^3\right),^{N} \tag{3}$$

where

$$\hat{w} = \frac{w}{w + \frac{\pi R^2}{A}(1 - w)}, \tag{4}$$

with $A$ denoting the image area (see Appendix B for derivation). Hence, to obtain a given value for $p$ we need

$$N = \frac{\log(1 - p)}{\log(1 - \hat{w}^4 w^3)}. \tag{5}$$

Note that the number of iterations depends on $\hat{w}$, which, in turn, depends on $R$. The value of $\hat{w}$ varies between $w$ and 1, depending on the value of $R$. In general, $\hat{w}$ increases as $R$ decreases, resulting in a smaller value for $N$. However, there is a trade-off here because for small values of $R$, it may not be possible to find other corresponding pairs meeting the requirements. Also, small values of $R$ may reduce the quality of the solution, because corresponding

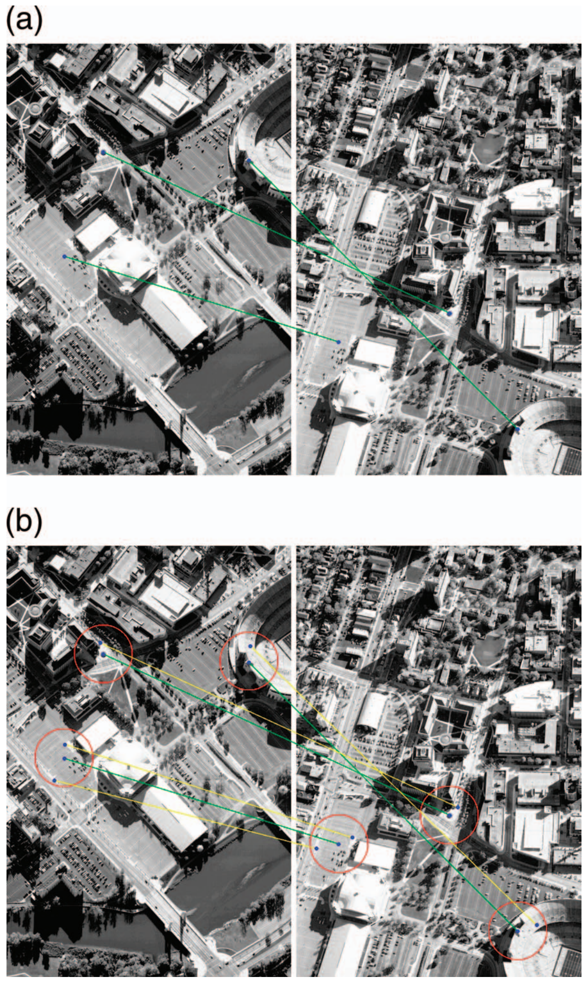


Fig. 4. Similarity sampling process.

pairs, which are very close together, provide less information about the fundamental matrix. We found that setting *R* to 200 pixels gave good results, while still being computationally very efficient.

Columns 4 and 5 of Table I show the required number of iterations and estimated processing time for the similarity sampling procedure. Compared with random sampling, substantially fewer iterations are required for the same percentage of inliers. Using this method, for images taken from the same camera, the average time to find the fundamental matrix and prune the outliers for the CLIF data set was 0.066 s per image pair. This was substantially faster than the SURF feature extraction and initial matching, which took an average of 3.8 s per image pair.

## III. GENERATING A BACKGROUND MODEL IN AERIAL VIDEO SEQUENCES

Many tracking algorithms, and in particular the combined tracking and segmentation algorithm presented in Section IV, require a static background model. In this section, we tackle the problem of generating a model for a small area of the background around a desired world location $\mathbf{x}_{\mathrm{w}}$. For vehicles on highways moving at high speed, a simple approach to forming a background model is to take the pixelwise median of three consecutive frames [21]. This approach works because the target moves more than its length from frame to frame; hence, there is a good chance that two of the three frames will contain the background. However, because we are interested in tracking on side streets with slow moving or stationary targets, this approach will not work. A vehicle may stop at a stoplight for dozens of frames in which case the median of three (or more) frames will still include the target. Instead, an approach, which is able to reasonably estimate the background even when it has never been visible, is needed.

Our approach to generating a static background model consists of two steps:

1) Obtain image patches from the previous and current frames around the predicted location of the target and register.
2) Replace the pixels in the current frame patch that correspond to moving and stationary targets using information from both the current and previous frames based on an inpainting algorithm.

The first step in the process is straightforward because the cameras are calibrated, and we have GPS information. Figs. 5a, 5b show two example image patches taken from consecutive frames at the same world coordinate. Their absolute difference is shown in Fig. 5c. Note that there is a small amount of residual error in the registration due to inaccuracies in the GPS/IMU information that shows up at sharp edges in the image, e.g., the edges of the roadway and markings on the road. Hence, the first step is to register the images by solving for the optimal translation to bring the previous frame into alignment with the current frame. The results are shown in Figs. 5d, 5e. Note that stationary objects no longer show up in the difference image.

The second step in the process is much more difficult. Our basic model for the background is the patch extracted from the current frame. For the most part, this is an accurate model because most of the patch will consist of stationary background objects. However, any pixels associated with moving or stationary targets of interest in the current frame will be included in the background model. The goal of the second step is to replace these pixels with accurate background pixels. Trivially, given the previous and current frame, there are four possibilities for a given pixel location:

1) The previous frame contains the background.
2) The current frame contains the background.
3) Both frames contain the background.
4) Neither frame contains the background.

Our approach is to attempt to classify each pixel into one of these four categories and then fill in the background model accordingly.

The first step is to determine which pixels in the current frame may not be accurate background pixels and hence need to be classified. An initial mask M of candidate pixels is obtained by simply thresholding the

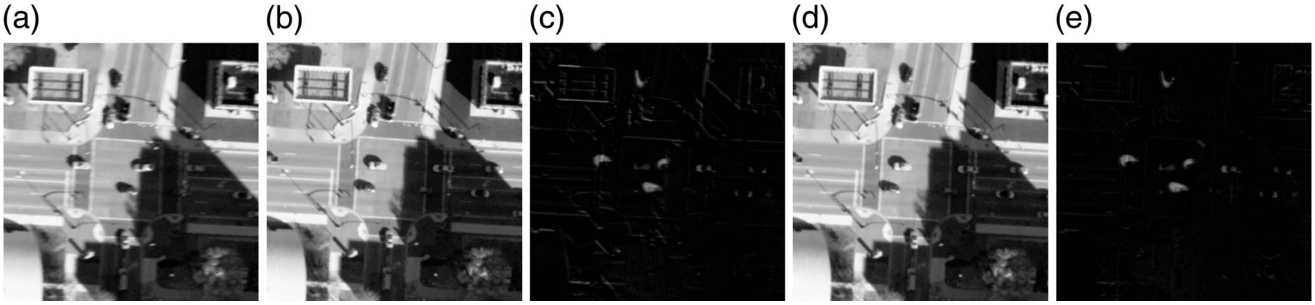


Fig. 5. Example of shifting previous frame to account for minor errors in localization.

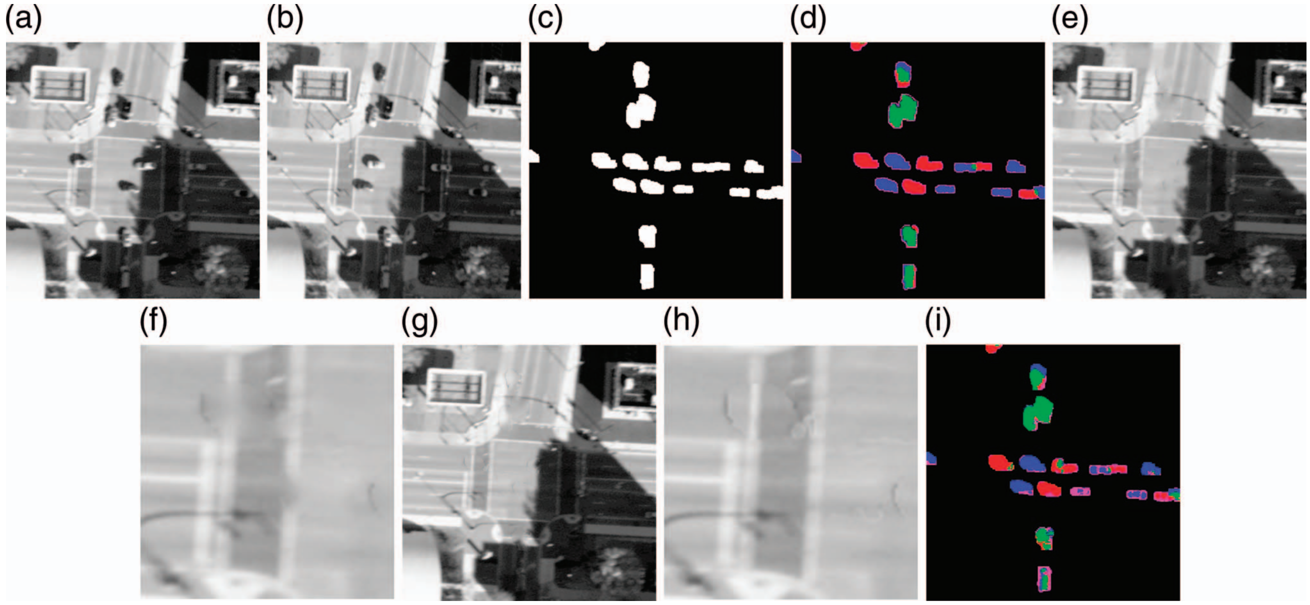


Fig. 6. Comparison of inpainting and inpainting-2 algorithms.

absolute difference between the previous and current frame. We then add to M the pixels that are predicted to belong to the target in the current frame based on the segmentation of the target in the previous frame, which needs to be done to account for stationary or slow-moving targets that may appear at the same location in both the previous and current frames. In this way, the tracking results are used to create a better model of the background. The current frame is used as the background model for all pixels not in M. The question is how to determine the classification of the pixels in M.

To classify and replace the pixels in M, we use an inpainting algorithm based on [22]. In the original inpainting algorithm, one pixel at a time is replaced based on directional and smoothness constraints until the background model is complete. This generally produces an accurate background model, particularly in areas of constant appearance such as roadways.

Starting from this algorithm, we modify the replacement of each pixel as follows. Let $v_g$ denote the value computed by this algorithm for a particular pixel. Under the assumption that this value is a reasonably accurate estimate of the background at this pixel, we can use this value to determine which of the four possible cases identified above is applicable. Let $v_1$ and $v_2$ denote the value of this pixel in the previous and current frames. We define the probability that $v_i$ contains the background to be

$$p_i = \exp\left\{-\frac{(v_i - v_g)^2}{2\sigma^2}\right\} \times f_i, \tag{6}$$

where $f_i$ is the fraction of surrounding pixels that were identified as being background pixels and $\sigma$ is a user-defined constant. We have used $\sigma = 20$ in our implementation. Based on these definitions, we set the value of the output pixel to

$$v = \begin{cases} v_1, & (p_1 \geq T) \cap (p_2 < T) \\ v_2, & (p_1 < T) \cap (p_2 \geq T) \\ \frac{v_1 p_1 + v_2 p_2}{p_1 + p_2}, & (p_1 \geq T) \cap (p_2 \geq T) \\ v_g, & \textit{otherwise} \end{cases}, \tag{7}$$

where $T$ is a threshold that controls how large the probability needs to be for a pixel to be considered a background pixel. We have used $T = 1/3$ in our implementation. We call this method inpainting-2.

Fig. 6g shows the resulting background patch after applying inpainting-2 to the current and previous frames.

Fig. 6a is the current frame, Fig. 6b the previous frame shifted, and Fig. 6c the initial mask M alluded to earlier. Fig. 6i shows the classification of each pixel based on (7): Red pixels were copied from the previous frame, blue pixels from the current frame, magenta pixels are a weighted average of the previous and current frame, and green pixels were kept at the inpainted value. The ground truth pixel classifications are shown in Fig. 6d. The classifications mostly agree, indicating that using the proposed inpainting-2 algorithm, we are able to correctly identify which pixels from each frame belong to the background.

Qualitatively, the primary difference between inpainting and inpainting-2 is that inpainting-2 is better able to preserve fine details. Comparing the detail images in Figs. 6f, 6h, we see that the road markings are better preserved with less smoothing, resulting in a more accurate background. Fig. 6e is obtained with inpainting and Fig. 6g with inpainting-2. Figs. 6f and 6h are for comparing fine detail with the two inpainting schemes.

A possible failure mode of the inpainting-2 algorithm is that false positives in the classification background pixels can result in targets not being removed from the background model. An example of this can be seen in the bottom of Fig. 6g in which a stopped van is not entirely removed (compare Fig. 6e). Nonetheless, we find that the improvement in accuracy from attempting to classify the pixels instead of simply inpainting all of them greatly outweighs the potential errors it introduces. We back up this statement quantitatively in Section V-A.

## IV. COMBINED TRACKING AND SEGMENTATION

Our basic tracking strategy is based on a combined tracking and segmentation algorithm we presented in [10]. Given a possible set of target locations, we first compute a soft assignment for each pixel in the image to one or more targets or the background. The appearance probabilities for each target and the background are then computed, and the results combined to obtain an overall score. This is repeated for several sets of target locations and the set with highest probability is retained as the final tracking solution.

The primary advantage of using combined segmentation and tracking is that it provides a principled way to incorporate information about the background and other targets into the tracking. Shadows from trees and buildings can dramatically change the appearance of the target. As a result, often the best model for the target is whatever remains after accounting for everything else, i.e., whatever does not match the background or another target.

Another advantage of incorporating segmentation into tracking is that it allows much more flexibility in the shape of the target compared with the standard rectangular bounding box. This flexibility is important because it makes it possible to exclude background pixels from the target model, reducing drift.

Mathematically, the goal of the tracking algorithm is to solve the optimization problem

$$\mathrm{X}^{(k)} = \underset{X \in \mathcal{F}}{\arg\max}\, P\{I|X\}P\{X|X^{(k-1)}\}, \tag{8}$$

where $\mathrm{X}^{(k)}$ denotes the locations of all of the targets in frame $k$, I denotes the input image, and $\mathcal{F}$ denotes the set of feasible target locations. $\mathcal{F}$ is used to constrain the solution to be physically meaningful, i.e., no targets can overlap and the range of motion of each target is limited. Although (8) is a standard optimization problem solved by nearly all probabilistic trackers, the constraints and challenges of tracking vehicles in wide-area aerial video require some special consideration, which will be discussed in the following sections.

### A. Generating Temporary Targets

Implicit in our tracking and segmentation algorithm is the assumption that every pixel was generated by either one of the known targets or the background. However, in a road environment, there will often be other moving vehicles near the known targets. These vehicles generate pixels that do not belong to any of the targets or the background. This can result in the known targets jumping to other moving objects.

A solution to this problem is to automatically detect all of the moving objects near the known targets and initialize them as temporary targets, i.e., targets that are used only for one frame. Detecting moving targets is fairly easy because we have an accurate background model. We, first, threshold the absolute difference between the previous image patch and the background model to produce a mask of pixels that belong to moving objects in the scene. After applying morphological operations to smooth the mask, we find connected components to determine possible temporary targets. The final step is to filter out any components that are too small or correspond to a known target in the previous frame. Those components that remain are used as temporary targets.

Fig. 7 shows the absolute difference and the thresholded mask for an example set of image patches. In this case, there are five connected components of which three are large enough to be targets. One of these corresponds to a known target (the white car) from the previous frame, but the other two become temporary targets. Fig. 7a is the current frame, Fig. 7b the previous frame, Fig. 7c the background model, Fig. 7d the absolute difference of the background and the previous frame, and Fig. 7e the final result that shows three likely targets.

### B. Computing Individual Probabilities

To optimize (8), we need to specify two distributions, the appearance distribution P{I|X} and the prior distribution $\mathrm{P}\{\mathrm{X}|\mathrm{X}^{(k-1)}\}$. For both distributions, we simplify the problem by treating all targets as independent. Hence, for $N$ targets, the appearance

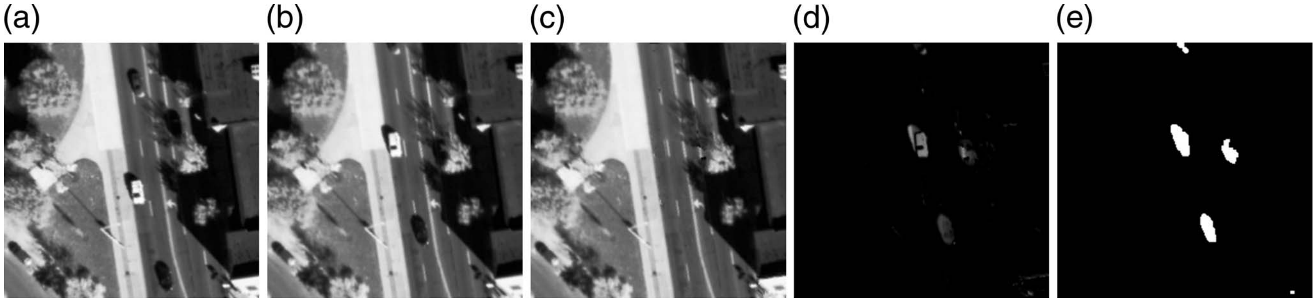


Fig. 7. Absolute difference between background model and previous frame and mask of those pixels exceeding threshold.

distribution can be decomposed into

$$P\{\mathrm{I}|\mathrm{X}\} = \prod_{i=0}^{N} P\{\mathrm{I}|\mathbf{x}_i\}, \tag{9}$$

where $x_i$ is the location of the $i$th target and $i = 0$ corresponds to the background. Similarly, the prior distribution can be decomposed into

$$P\{X|X^{(k-1)}\} = \prod_{i=0}^{N} P\{\mathbf{x}_i|\mathbf{x}_i^{(k-1)}\}, \tag{10}$$

where the prior for the background target is defined to be 1. Hence, we can rewite (8) as

$$X^{(k)} = \arg\max_{X\in\mathcal{F}} \prod_{i=0}^{N} P\{I|\mathbf{x}_i\} P\{\mathbf{x}_i|X_i^{(k-1)}\}. \tag{11}$$

Note that we can evaluate the probabilities for each target independently, which greatly improves the efficiency over evaluating the probabilities together. However, the optimization itself must consider all targets together because X must be drawn from the feasible set $\mathcal{F}$. In this section, we explain how to evaluate the individual probabilities, while the next section will explain the optimization.

1) *Appearance Distribution* $P\{\mathrm{I}|\mathrm{x}\}$: Many applications in computer vision use the Gaussian distribution to model appearance because of its simplicity and familiarity. However, it tends to underestimate the probability of rare events occurring which is unrealistic and can have a profound negative impact on performance [23–26]. For this reason, we model each pixel with a standard $t$-distribution that has probability density function

$$t(z|\mu, s, \nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi\nu s}}\left[1 + \frac{1}{\nu}\frac{(z-\mu)^2}{s}\right]^{-\frac{\nu+1}{2}}, \tag{12}$$

where $\mu$ is the location parameter specifying the median of the distribution, $s$ is the scale parameter, which is similar to variance, and $\nu$ is the degree-of-freedom parameter, which controls the heaviness of the tails [27]. In all of our experiments, we estimate $\mu$ and $s$ online from the data but fix $\nu$ at 5.

For simplicity, we treat each pixel belonging to the target as coming from independent distributions with separate means but shared scale and degree-of-freedom parameters. Thus, the appearance distribution is given by

$$P\{\mathbf{I}|\mathbf{x}\} \triangleq \prod_{x_i\in\mathcal{D}_I} t\left(I(\mathbf{x}_i)|\mu(\mathbf{x}_i - \mathbf{x}), s, \boldsymbol{\nu}\right), \tag{13}$$

where $\mathcal{D}_I$ is the domain of the image, $\mu(\mathbf{x})$ is the mean image of the target, and $s$ and $\nu$ are the common scale and degree-of-freedom parameters.

The segmentation of the target is denoted $z(\mathbf{x}_i|\mathbf{x})$ and specifies the probability that the pixel at location $\mathbf{x}_i$ was generated by the target given that the target is centered at $\mathbf{x}$. $z(\mathbf{x}_i|\mathbf{x})$ can easily be computed using Bayes' rule

$$z(\mathbf{x}_i|\mathbf{x}) = \frac{p_t}{p_t + p_b}, \tag{14}$$

where $p_t$ and $p_b$ are the appearance probabilities of the target of interest and the background for the pixel at location $x_i$. Note that only the target of interest and the background need to be considered because we enforce the constraint that no targets can overlap.

Given the segmentation, we wish to modify the appearance distribution given above so that pixels that are not likely to belong to the target have only a small impact on the overall probability, while pixels, which are definitely from the target, have a large impact on the overall probability. We can achieve this by weighting the scale factor for each pixel based on the segmentation so that (13) becomes

$$P\{\mathbf{I}|\mathbf{x}\} \triangleq \prod_{\mathbf{x}_i\in\mathcal{D}_I} t\left(\mathbf{I}(\mathbf{x}_i)|\mu(\mathbf{x}_i - \mathbf{x}), sw(\mathbf{x}_i, \mathbf{x}), \boldsymbol{\nu}\right), \tag{15}$$

with

$$w(\mathbf{x}_i, \mathbf{x}) = \exp\left\{k(1 - z(\mathbf{x}_i|\mathbf{x}))\right\}, \tag{16}$$

where $k > 0$ is a weighting factor that controls the importance of the segmentation. The idea behind (16) is that for segmentation probabilities close to 1, $w(\mathbf{x}_i, \mathbf{x}) \approx 1$, and so the scale factor is unchanged. However, for segmentation probabilities close to 0, $w(\mathbf{x}_i, \mathbf{x}) \approx \exp\{k\}$, which is a large number causing these pixels to have a large scale factor (which corresponds to a large variance) and thus limiting their impact on the overall probability. Note that the form of (16) was selected to ensure that the product of the normalizing scale factors of the probability distributions in (9) remains constant regardless of the segmentation.

2) *Prior Distribution* $P\{\mathbf{x}|\mathbf{x}^{(k-1)}\}$: The full state for each target is given by $\mathbf{y} = [x\ y\ v\ \theta\ a\ \boldsymbol{\omega}]^T$, where $(x, y)$ is the position, $v$ is the velocity, $\theta$ is the direction of travel, $a$ is the acceleration, and $\omega$ is the angular rate. Our continuous time model for the target is given by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \\ \dot{\theta} \\ \dot{a} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ a \\ \omega \\ 0 \\ 0 \end{bmatrix}. \tag{17}$$

After discretizing the system, we use an extended Kalman filter to estimate the target state and covariance at each frame. The process and measurement covariances were determined through maximum likelihood fitting of several manually marked vehicle tracks. A Kalman filter was used because of its simplicity, although more advanced techniques, such as the generative maneuvering target model [28] may yield better results.

The prediction step of the Kalman filter gives an estimated mean and covariance for the state in the next frame. By marginalizing out the position variables, we can use this as the prior $P\{\mathbf{x}|\mathbf{x}(k-1)\}$.

## C. Solving for the Optimal Positions

Recall that our goal is to optimize (11), which is equivalent to

$$\mathrm{X}^{(k)} = \arg\max_{X\in\mathcal{F}} \sum_{i=0}^{N} \log\left(P\{\mathrm{I}|\mathbf{x}_i\}P\{\mathbf{x}_i|\mathbf{x}_i^{(k-1)}\}\right). \tag{18}$$

To optimize this equation, we first compute the individual log probabilities for each target on a dense grid over the feasible positions. In our case, the feasible positions for a target are taken to be all locations with a prior probability above some threshold, up to a maximum distance $R$ from the predicted location.

If our feasible set $\mathcal{F}$ was given by $\mathbb{R}^{2N}$, then we could optimize (18) by simply maximizing the individual terms of the sum, i.e., by choosing the maximum individual probabilities for each target. However, this would ignore the physical reality that targets in wide-area aerial video cannot overlap.

We can enforce the constraint that targets cannot overlap by treating (18) as a maximum-weighted assignment problem. As a simplifying assumption, we first determine the locations of local maxima of the log probabilities for each target. We next merge the locations of the maxima from different targets, with locations that are sufficiently close treated as the same. These merged locations represent the candidate locations where there could be targets.

We next create a bipartite graph connecting the targets to the maxima locations with the edge weights given by the log probabilities. Note that we also add nil maps to the graph so that a target does not need to correspond to any of the locations and not every location needs to have a target. We can now use the Hungarian algorithm to efficiently find the assignment between the targets and locations that maximizes the sum of the log probabilities, while ensuring that each location maps to at most one target and vice versa. This is precisely the optimization problem (18). Note that any target that was mapped to a nil location is assumed to be missing in the current frame.

## D. Updating the Target Models

After tracking is completed for each frame, we update the target models of all successfully tracked targets. In all of the model parameter updates, we make use of the segmentation of the target in such a way as to ensure that pixels that were probably not generated by the target do not influence the model. This is important to avoid common tracking problems, such as drift, which occur when background objects become part of the target model.

Given the optimal location for a target $\mathbf{x}$, the nominal scale parameter $s$ is updated using a weighted variance

$$s{=}\alpha s{+}(1-\alpha)\left(\frac{\nu-2}{\nu}\right)\frac{\sum z(\mathbf{x}_i|\mathbf{x})\,(I(\mathbf{x}_i)-\mu(\mathbf{x}_i-\mathbf{x}))^2}{\sum z(\mathbf{x}_i|\mathbf{x})}, \tag{19}$$

where $\alpha$ is a weighting factor that controls the importance of the previous estimate, $z(\mathbf{x}_i|\mathbf{x})$ is the segmentation computed using (14), and the sums are taken over all pixels with nonzero segmentation. In our experiments, we have used $\alpha = 0.75$. Note that the ratio $(\nu-2)/\nu$ corrects for the difference between the variance and scale factor of a univariate $t$-distribution; a $t$-distribution with scale factor $s$ has variance $s \times \nu/(\nu-2)$.

If $z(\mathbf{x}_i|\mathbf{x})$ is 1 for all pixels, then the second term in (19) is just the sample variance of all the pixels assigned to this target. By weighting the samples using $z(\mathbf{x}_i|\mathbf{x})$, pixels that are not likely to be part of the target (i.e., $z(\mathbf{x}_i|\mathbf{x}) \approx 0$) have no impact on the variance computation. Note that the scale factor is updated for both the background and the targets.

The mean image for the targets is also updated using a weighted average in which the segmentation $z(\mathbf{x}_i|\mathbf{x})$, computed using (14), provides the weighting. This case is slightly more complicated than the scale factor because we wish to keep track of a separate mean for each pixel that makes up the target. The idea is to use a weighted mean for the appearance of each pixel over time, where the weight again comes from the segmentation to limit the impact of pixels that were not likely to be from the target

$$\begin{aligned}\mu(\mathbf{x}_i-\mathbf{x}) = \mathrm{I}(\mathbf{x}_i)&\left(\frac{z(\mathbf{x}_i|\mathbf{x})}{\min(\beta,\bar{z}(\mathbf{x}_i-\mathbf{x}))+z(\mathbf{x}_i|\mathbf{x})}\right),\\ &+\mu(\mathbf{x}_i-\mathbf{x})\left(\frac{\min(\beta,\bar{z}(\mathbf{x}_i-\mathbf{x}))}{\min(\beta,\bar{z}(\mathbf{x}_i-\mathbf{x}))+z(\mathbf{x}_i|\mathbf{x})}\right)\end{aligned} \tag{20}$$

where $\bar{z}(\mathbf{x}_i-\mathbf{x})$ is the sum of the segmentation at the pixel location $\mathbf{x}_i-\mathbf{x}$ over all previous frames. The constant term $\beta$ is used to limit the weight on the previous mean.

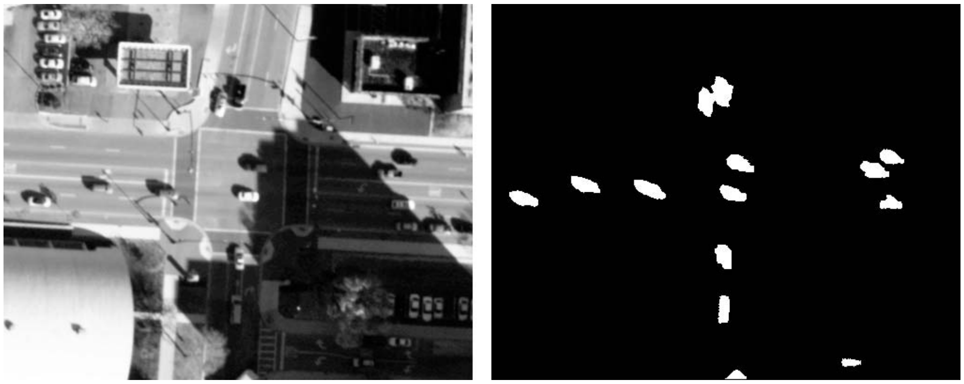


Fig. 8. Example image from background sequence used to objectively evaluate background models along with manually generated vehicle mask.

This is important because the target appearance changes over time, and so the target appearance needs to be able to change over long tracks. We have used $\beta = 10$ in our experiments.

The final step in updating the model is to add and remove pixels from the target mask. For each target, we have a pixel mask relative to the center coordinate that indicates pixels that could possibly be part of the target. These pixels are updated as part of the scale and mean updates above, with all other pixels assumed to have segmentation 0. Initially, the mask for each target is a square box. However, many of these initial pixels are actually part of the background, not the target. Although these pixels should have a segmentation of approximately 0, they add to the computational complexity and may occasionally falsely match part of the background. Furthermore, some of the target pixels are likely to be missing from the initial mask. Also, the set of pixels that make up the target can change over time as the orientation of the vehicle changes.

To compensate for these changes, we update the mask after each successful track. First, we smooth the segmentation over several frames and then threshold the result. The set of pixels that are above the threshold are assumed to definitely be part of the target and are used as an initial mask for the target. Starting from the initial mask, we then iterate twice over the mask, adding all bordering pixels. By adding these bordering pixels to the mask, we allow for new pixels to be added to the target model over time. Together, these two operations, first limiting to just the definite target pixels and then adding bordering pixels, allow the tracker to compensate for slow changes in scale and orientation and also make the tracker faster and more robust compared with a fixed square mask.

## V. RESULTS

In this section, we present results for both the background generation procedure from Section III and the combined tracking and segmentation algorithm from Section IV.

### A. Background Generation

It is difficult to objectively evaluate the algorithm for generating a static background because there is no ground truth available. Although much of the difference between subsequent frames is due to object motion, tall buildings also exhibit substantial parallax with an apparent motion of up to a few meters per frame. It is difficult to say what the ground truth is in this case. There are also differences due to occasional blurring and contrast changes that further complicate the issue.

In an attempt to obtain some sort of an objective measure, we captured 30 consecutive stabilized image patches with dimensions 400 × 500 of the same location showing a busy intersection. We then manually segmented the vehicles on the roadways in each image (note that parked vehicles were not included, but temporarily stopped vehicles were). These segmented regions represent the areas that are definitely not the background. We assumed that the rest of the image was a reasonable estimate of the background. Fig. 8 shows an example image from this set along with the manually generated vehicle mask.

We next considered each pair of consecutive images from the set of 30 and evaluated various algorithm for generating a background. The ground truth for each pixel was taken from the most recent frame in which the pixel was not marked as belonging to a vehicle. In all, we tested four methods. First, we considered simply using the mean of the pair of images as the background. This gives a very crude model that represents the baseline worst-case performance. Second, we considered the algorithm given in [21] in which the background is given by the median over sets of three images (an additional previous image was used in this case). Third, we evaluated the basic inpainting algorithm of Telea [22]. Finally, we considered our proposed inpainting-2 algorithm.

Table II gives the average absolute pixel error between the generated background and the ground truth for all four methods. Only pixels that were marked as belonging to a vehicle in the current or previous frame were used in computing the average to highlight the differences

TABLE II
Mean Absolute Pixel Error in Either the Whole Image or Just the Regions with Moving Objects for Four Different Background Modeling Methods

| Model | All Vehicle Pixels | Limited Vehicle Pixels |
|---|---|---|
| Mean | 33.21 | 29.41 |
| Median [21] | 15.33 | 10.14 |
| Inpainting | 12.33 | 11.48 |
| Inpainting-2 | 7.15 | 5.15 |

between the methods. The second column shows results for all pixels associated with vehicles in the current or previous frame, including vehicles, which were moving slowly or stopped. In the third column, we further restricted the average to only those pixels that corresponded to the background in either the current or the previous frame. By doing this, we are able to focus on the results for moving vehicles as opposed to both moving and stopped vehicles. In both cases, the proposed inpainting-2 algorithm gives the best performance by a wide margin.

### B. Tracking

In this section, we present both qualitative and quantitative results for the tracking algorithm presented in Section IV. Fig. 9 shows an example tracking result in which a vehicle is successfully tracked for 151 frames across five of the cameras before leaving the field of view. This sequence is particularly difficult because the vehicle is tracked down small side streets that have overhanging trees and shadows. The vehicle also turns at two intersections, resulting in large changes in the vehicle orientation. Because of these difficulties, the appearance of the vehicle changes dramatically over the course of the sequence, and so it is very unlikely that a tracker based only on the object appearance could successfully track the vehicle. However, by including the background information, it is possible to track the vehicle even under these difficult circumstances.

To objectively evaluate the tracking performance, we also computed two standard measures of tracking performance, which are often used to evaluate multitarget tracking algorithms. The first measure is a simplified version of the multiple object tracking accuracy (MOTA) from which is given by

$$MOTA = 1 - \frac{\sum_{t=1}^{n}\sum_{i=1}^{N_t}\left[overlap(T_{i,t}, G_{i,t}) < 0.25\right]}{\sum_{t=1}^{n} N_t}, \quad (21)$$

where $n$ is the number of frames, $N_t$ is the number of targets specified in the ground truth for the $t$th frame, and $T_{i,t}$ and $G_{i,t}$ are the bounding boxes based on the tracker and ground truth, respectively, for the $i$th target in the $t$th frame, and *overlap* computes the overlap between these bounding boxes given by

$$overlap(T_{i,t}, G_{i,t}) = \frac{\left|T_{i,t} \cap G_{i,t}\right|}{\left|T_{i,t} \cup G_{i,t}\right|}. \quad (22)$$

The second objective measure we used is called the multiple object tracking precision (MOTP) and is just the average overlap for all targets given by

$$MOTP = \frac{\sum_{t=1}^{n}\sum_{i=1}^{N_t} overlap(T_{i,t}, G_{i,t})}{\sum_{t=1}^{n} N_t}.$$

For the single target example from Fig. 9, we obtained a MOTA of 1 because the vehicle is successfully tracked in every frame in which it is visible. The MOTP for this sequence is 0.58. The precision is fairly low because the tracker makes use of the vehicle shadow, which causes the bounding box to be shifted somewhat compared with the ground truth. To verify the advantages of combining tracking and segmentation, we also attempted to track the same vehicle assuming a rectangular bounding box for the target instead of per pixel segmentation. This method was able to track the target as long as it remains in the open with minimal changes in appearance. However, when the target enters a small side road with heavy tree shadows, the tracker latches onto the background and the target is lost (see Fig. 1). This occurs about halfway through the sequence resulting in a MOTA of only 0.49 if segmentation is not used.

To further evaluate the algorithm, we also tested it on a much larger set of vehicles consisting of 17 vehicles. In this case, we tracked all vehicles that crossed a particular intersection heading west on a busy road. Tracking vehicles on this road was challenging due to several large building shadows. Fig. 11 shows the resulting vehicle tracks. Most of the vehicles simply continued on the road for the length of the sequence; however, three of the 17 vehicles turned into parking lots. Of these, the two shown in magenta and green in Fig. 11 were successfully tracked, while the third was lost.

For this sequence, the MOTA is 0.86, and the MOTP is 0.45. Fig. 12 shows a typical tracking failure from this sequence. In this case, a gray-colored vehicle nearly disappears when it enters a building shadow. The vehicle then turns; however, the tracker expects it to continue straight. By the time the vehicle re-emerges from the building shadow, it is too far from the predicted location to be reacquired.

## VI. CONCLUSIONS

In this paper, we have presented a complete framework for tracking vehicles in wide-area aerial video. While other research has focused on tracking vehicles in wide-open areas such as highways, we have focused on small side streets. This brings additional challenges such has shadows from buildings and occlusion by trees.

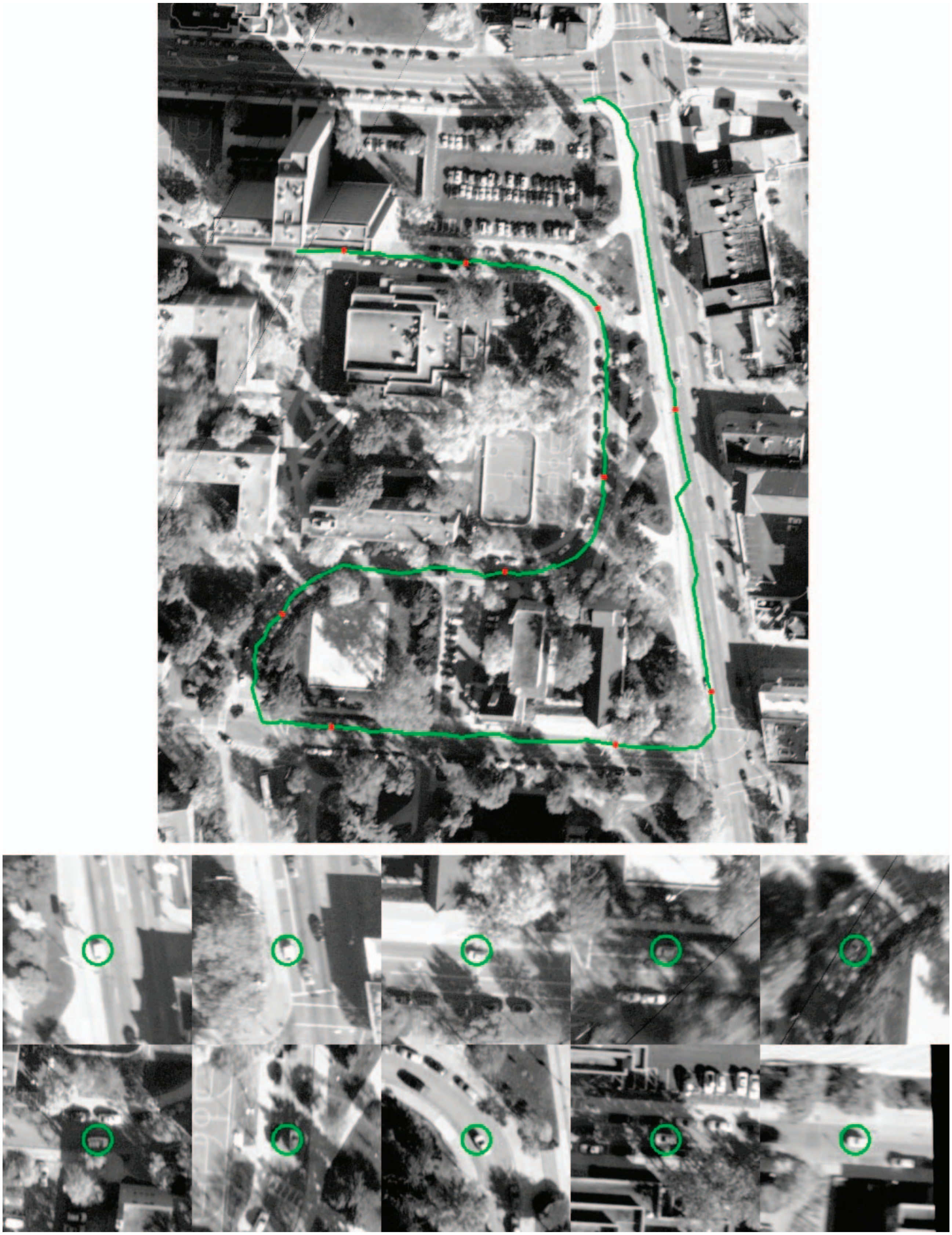


Fig. 9. Tracking single vehicle on side street with significant shadows/occlusion. Top figure shows map of entire tracking sequence starting from upper right corner. Bottom figures shows results for every fifteenth frame, with corresponding locations marked in red on map.

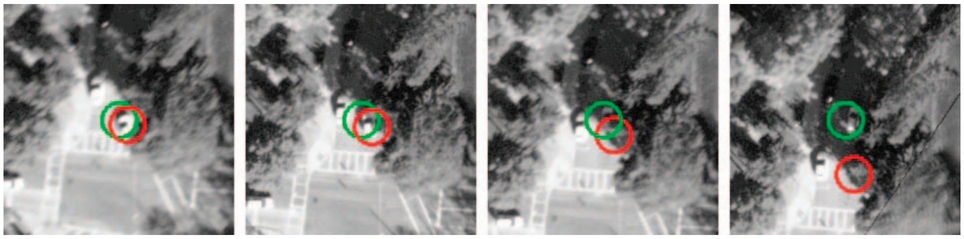


Fig. 10. Comparison of tracking with and without segmentation. Green result is with segmentation. Red result is without segmentation.

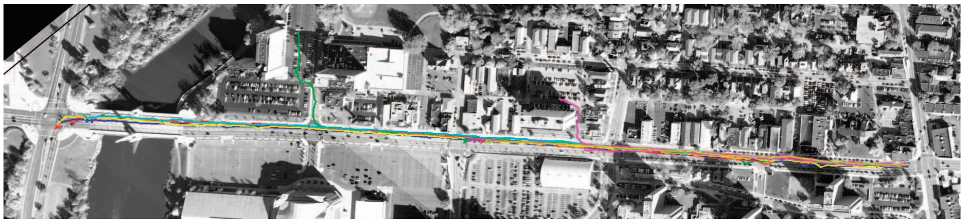


Fig. 11. Tracking multiple targets on busy road.

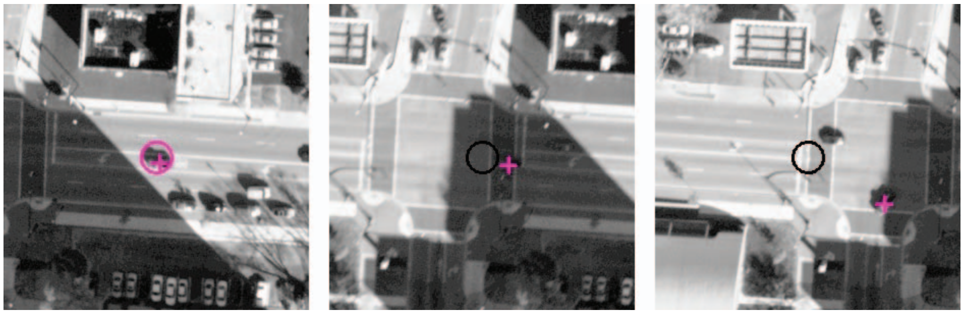


Fig. 12. Tracking failure in which vehicle turns while obscured by building shadow. Magenta "+" shows ground truth location, while black circles show predicted target location.

To address these challenges, we present a method for simultaneously segmenting and tracking vehicles. The segmentation depends on an accurate background model that we obtain via a novel inpainting algorithm. By incorporating segmentation into the tracking, we are able to track the vehicle without drifting even when the appearance of the target changes dramatically.

Future work will focus on two areas. First, we would like to automatically detect moving vehicles and track them without user input. We believe the method described in Section IV-A for generating temporary targets could be adapted to this purpose. Special care will be needed to avoid starting tracks on the tops of tall buildings, which undergo significant parallax. Second, we would like to improve the computational efficiency of the proposed algorithm. This is important to use this method for tracking the huge number of vehicles that may be present in wide-area aerial video.

## APPENDIX A. CALIBRATION RESULTS FOR THE CLIF 2007 DATA SET

Our camera model is of the form

$$\mathrm{P} = \mathrm{K}[\mathrm{R}_{cg}\ \mathbf{t}_{cg}], \tag{23}$$

where K is the matrix of intrinsic parameters and $\mathrm{R}_{cg}$ and $\mathbf{t}_{cg}$ capture the rotation and translation between the camera and GPS/IMU. Hence, the image of world point $\mathbf{x}_w$ is given by the image point

$$\mathbf{x}_i = \mathbf{P}\mathbf{x}_w. \tag{24}$$

Furthermore, it is assumed that the measured coordinate $\mathbf{x}d = \{xd, yd\}$ differs from the true image coordinate $\mathbf{x}_i = \{x_i, y_i\}$ due to lens distortion with the transformation given by

$$x_d = k_r x_i + 2p_1 x_i y_i + p_2(r^2 + 2x_i^2), \tag{25}$$

$$y_d = k_r y_i + 2p_2 x_i y_i + p_1(r^2 + 2y_i^2), \tag{26}$$

where $r = \sqrt{x_i^2 + y_i^2}$ and

$$k_r = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6. \tag{27}$$

The matrix K can be written as

$$\mathrm{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Using this form for the matrix K, Table III gives the intrinsic calibration results for all six cameras. For the extrinsic calibration, we set $\mathbf{t}_{cg} = \mathbf{0}$ for all six cameras

TABLE III
Intrinsic Calibration Results

| Camera | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $f_x$ | 14842.6 | 14832.0 | 14891.2 | 14799.1 | 14866.1 | 14825.0 |
| $f_y$ | 14794.1 | 14837.8 | 14853.2 | 14789.1 | 14864.2 | 14834.6 |
| $c_x$ | 1304.67 | 1396.33 | 1355.56 | 1339.13 | 1358.86 | 1284.32 |
| $c_y$ | 1943.50 | 2027.40 | 1979.65 | 1981.63 | 1973.12 | 1933.06 |
| $k_1$ | 0.369541 | 0.368353 | 0.351070 | 0.379073 | 0.336898 | 0.382780 |
| $k_2$ | −2.97691 | −2.89313 | −2.97542 | −3.53149 | −1.68117 | −3.23134 |
| $k_3$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $\rho_1$ | −0.002689 | 0.001948 | −0.000062 | 0.000901 | −0.001372 | −0.003216 |
| $\rho_2$ | 0.000767 | 0.002364 | 0.001031 | −0.000213 | 0.001554 | −0.002103 |

TABLE IV
Extrinsic Calibration Results

| Camera | | $R_{cg}$ | |
|---|---|---|---|
| 0 | −0.003056 | 0.999994 | −0.001514 |
| | −0.992290 | −0.003221 | −0.123895 |
| | −0.123899 | 0.001124 | 0.992294 |
| 1 | 0.005981 | 0.999974 | −0.004125 |
| | −0.992494 | 0.006440 | 0.122127 |
| | 0.122150 | 0.003363 | 0.992506 |
| 2 | −0.002183 | 0.985095 | −0.171997 |
| | −0.991783 | −0.024133 | −0.125634 |
| | −0.127912 | 0.170309 | 0.977053 |
| 3 | 0.001986 | 0.985428 | −0.170079 |
| | −0.992039 | 0.023357 | 0.123748 |
| | 0.125918 | 0.168479 | 0.977630 |
| 4 | −0.000729 | 0.986122 | 0.166018 |
| | −0.991915 | 0.020355 | −0.125263 |
| | −0.126904 | −0.164767 | 0.978135 |
| 5 | 0.013562 | 0.985288 | 0.170362 |
| | −0.991413 | −0.008912 | 0.130467 |
| | 0.130066 | −0.170668 | 0.976706 |

because the distance between the cameras and the GPS/IMU is negligible. The rotation matrices $\mathrm{R}_{cg}$ are given in Table IV. Note that cameras one to five are delayed by one time-step relative to the GPS/IMU and camera 0, e.g., timestamp 101000 of the GPS/IMU corresponds to timestamp 101000 for camera 0 but timestamp 101001 for cameras one to five.

## APPENDIX B. DERIVATION OF INLIER PROBABILITY FOR MODIFIED RANSAC ALGORITHM

Recall that we choose three pairs from the initial correspondences completely randomly, followed by four more pairs that are also selected randomly but with a constraint on their proximity to the initial three pairs. Denote by $\mathbf{x}_i$, $i = \{1, 2, 3\}$, the initial three pairs and $\mathrm{y}_i$, $i = \{1, 2, 3, 4\}$, the additional four pairs. Let $S_*$ denote the event that the pair $*$ is an inlier, e.g., $S_{\mathrm{x}_2}$ means that $\mathbf{x}_2$ is an inlier. Furthermore, assume that the fraction of inliers in the initial set of correspondences is given by $w$. We are first interested in computing the probability that all the selected points are inliers for a single random selection, i.e.,

$$\hat{p} = P\left\{\left(\bigcap_{i=1}^{3} S_{\mathbf{x}_i}\right)\bigcap\left(\bigcap_{i=1}^{4} S_{\mathbf{y}_i}\right)\right\}. \tag{28}$$

We first split (28) using Bayes' rule to obtain

$$\hat{p} = P\left\{\bigcap_{i=1}^{4} S_{\mathbf{y}_i}\,\middle|\,\bigcap_{i=1}^{3} S_{\mathbf{x}_i}\right\} P\left\{\bigcap_{i=1}^{3} S_{\mathbf{x}_i}\right\}. \tag{29}$$

Note that the $\mathbf{x}_i$'s are independent of each other and that the $\mathbf{y}_i$'s are independent given the $\mathbf{x}_i$'s. Hence, we can rewrite (29) as

$$\hat{p} = \prod_{i=1}^{4} P\left\{S_{\mathbf{y}_i}\,\middle|\,\bigcap_{i=1}^{3} S_{\mathbf{x}_i}\right\} \times \prod_{i=1}^{3} P\left\{S_{\mathbf{x}_i}\right\}, \tag{30}$$

$$= \left(P\left\{S_{\mathbf{y}}\,\middle|S_{\mathbf{x}_*}\right\}\right)^4 \left(P\left\{S_{\mathbf{x}}\right\}\right)^3, \tag{31}$$

where $\mathrm{S}_\mathbf{y}$ and $\mathrm{S}_\mathbf{x}$ are shorthand for any $S_{\mathbf{y}_i}$ and $S_{\mathbf{x}_i}$ respectively, and $S_{\mathbf{x}_*} = \bigcap_{i=1}^{3} S_{\mathbf{x}_i}$.

It is clear that $P\{S_\mathbf{x}\} = w$, because these samples are drawn from the initial set of correspondences. Computing $P\{S_\mathbf{y}|S_{\mathbf{x}_*}\}$ is more complicated and requires us to rephrase the problem slightly. Let $\mathbf{z}$ be a randomly selected correspondence form the initial set and let $R_{\mathbf{x},\mathbf{z}}$ denote the event that two corresponding pairs $\mathbf{x}$ and $\mathbf{z}$ are within a radius $R$ in both images. Then, $P\{S_\mathbf{y}|S_{\mathbf{x}_*}\}$ is equivalent to

$$P\{S_\mathbf{z}|R_{\mathbf{x},\mathbf{z}} \cap S_\mathbf{x}\}. \tag{32}$$

Note that rather than constraining the selection of the second point based on the first (as we did when selecting $\mathbf{y}$) and then asking if it is an inlier, we are now considering an additional randomly selected point and asking whether or not this point is an inlier given that it is within a certain radius of the first point. Both events have equal probability but the latter is easier to manipulate.

We can now use Bayes' rule to rewrite (32) as

$$P\{S_\mathbf{z}|R_{\mathbf{x},\mathbf{z}} \cap S_\mathbf{x}\} = \frac{P\{R_{\mathbf{x},\mathbf{z}}|S_\mathbf{z} \cap S_\mathbf{x}\}P\{S_\mathbf{z}\}}{P\{R_{\mathbf{x},\mathbf{z}}|S_\mathbf{z} \cap S_\mathbf{x}\}P\{S_\mathbf{z}\} + P\{R_{\mathbf{x},\mathbf{z}}|\bar{S}_\mathbf{z} \cap S_\mathbf{x}\}P\{\bar{S}_\mathbf{z}\}}, \tag{33}$$

where we have made use of the fact that $S_{\mathbf{x}}$ and $S_{\mathbf{z}}$ are independent. Trivially, $P\{S_{\mathbf{z}}\} = w$ and $P\{\bar{S}_{\mathbf{z}}\} = 1 - w$. Hence, the only remaining probabilities are $P\{R_{\mathbf{x},\mathbf{z}}|S_{\mathbf{z}} \cap S_{\mathbf{x}}\}$ and $.P\{R_{\mathbf{x},\mathbf{z}}|\bar{S}_{\mathbf{z}} \cap S_{\mathbf{x}}\}$. To compute these probabilities, we need to know something about the spatial distribution of the corresponding pairs in the images. We assume that both the inliers and outliers are uniformly distributed. Furthermore, we assume that the coordinates of the matching pair in the two images are independent.

For $P\{R_{\mathbf{x},\mathbf{z}}|S_{\mathbf{z}} \cap S_{\mathbf{x}}\}$, if $\mathbf{x}$ and $\mathbf{z}$ are within radius $R$ in one image than the must be in the other as well by the assumption of a local similarity transform. Hence, $P\{R_{\mathbf{x},\mathbf{z}}|S_{\mathbf{z}} \cap S_{\mathbf{x}}\}$ is just the probability that two randomly selected inliers are within a radius $R$ in one image, which is given by $\frac{\pi R^2}{A}$, where $A$ is the image area.

For $P\{R_{\mathbf{x},\mathbf{z}}|\bar{S}_{\mathbf{z}} \cap S_{\mathbf{x}}\}$, knowing that $\mathbf{x}$ and $\mathbf{z}$ are within radius $R$ in one image tells us nothing about their relationship in the other image by assumption. Hence, $R_{\mathbf{x},\mathbf{z}}$ is true only if $\mathbf{x}$ and $\mathbf{z}$ happen to be within a radius $R$ in both images independently. This probability is given by $\frac{\pi R^2}{A} \times \frac{\pi R^2}{A} = \left(\frac{\pi R^2}{A}\right)^2$. Putting everything together, we have

$$P\{S_{\mathbf{z}}|R_{\mathbf{x},\mathbf{z}} \cap S_{\mathbf{x}}\} = \frac{\frac{\pi R^2}{A} w}{\frac{\pi R^2}{A} w + \left(\frac{\pi R^2}{A}\right)^2 (1 - w)} \quad (34)$$

$$= \frac{w}{w + \frac{\pi R^2}{A}(1 - w)}. \quad (35)$$

Thus, (30) can be rewritten as

$$\hat{p} = \hat{w}^4 w^3, \quad (36)$$

where

$$\hat{w} = \frac{w}{w + \frac{\pi R^2}{A}(1 - w)}. \quad (37)$$

Finally, we need to compute the probability $p$ of getting at least one success in $N$ independent trials. We note that the probability of the complement, i.e. of no successes in $N$ trials is given by $(1 - \hat{p})^N$. Hence, the probability of at least one success is given by

$$p = 1 - (1 - \hat{p})^N. \quad (38)$$

REFERENCES

[1] Columbus Large Image Format (CLIF) 2007. Available: https://www.sdms.afrl.af.mil/.

[2] Collins, R. Zhou, X., and Teh, S.
An open source tracking testbed and evaluation web site.
In *Proceedings of the IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2005.

[3] Ali, S., and Shah, M.
Cocoa: tracking in aerial imagery.
*Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications III*, **6209** (2006), 62090D–6.

[4] Cao, X., Wu, C., Lan, J., Yan, P., and Li, X.
Vehicle detection and motion analysis in low-altitude airborne video under urban environment.
*IEEE Transactions on Circuits and Systems for Video Technology*, **21**, 10 (2011), 1522–1533.

[5] Comaniciu, D., Ramesh, V., and Meer, P.
Kernel-based object tracking.
*IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**, 5 (2003), 564–575.

[6] Reilly, V., Idrees, H., and Shah, M.
Detection and tracking of large number of targets in wide area surveillance.
In *Proceedings of the European Conference on Computer Vision*, Heraklion, Crete, Greece, 2010, 186–199.

[7] Dessauer, M., and Dua, S.
Low-resolution vehicle tracking using dense and reduced local gradient features maps.
*Proceedings of SPIE*, **7694** (2010), 76941I.

[8] Meyer, F., and Bouthemy, P.
Region-based tracking using affine motion models in long image sequences.
*CVGIP: Image Understanding*, **60**, 2 (1994), 119–140.

[9] Smith, S., and Brady, J.
Asset-2: real-time motion segmentation and shape tracking.
*IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**, 8 (1995), 814–820.

[10] Aeschliman, C., Park, J., and Kak, A. C.
A probabilistic framework for joint segmentation and tracking.
In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, 2010, 1371–1378.

[11] Jaikumar, P., Singh, A., and S. Mitra, S.
Background subtraction in videos using Bayesian learning with motion information.
In *Proceedings of the British Machine Vision Conference*, Leeds, United Kingdom, Sep. 2008, 615–624.

[12] OpenCV. Available:
http://sourceforge.net/projects/opencvlibrary.

[13] Zhang, Z.
A flexible new technique for camera calibration.
*IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**, 11 (2000), 1330–1334.

[14] Redmill, K., Martin, J., and Ozguner, U.
Aerial image registration incorporating GPS/IMU data.
*Proceedings of SPIE*, **7347** (2009), 73470H.

[15] Triggs, B.
Autocalibration and the absolute quadric.
In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997, 609–614.

[16] Snavely, N., Seitz, S. M., and Szeliski, R.
Photo tourism: exploring image collections in 3D.
*ACM Transactions on Graphics*, **25** (2006), 835–846.

[17] Simple Sparse Bundle Adjustment (SSBA). Available:
http://www.inf.ethz.ch/personal/chzach/opensource.html.

[18] Bay, H., Tuytelaars, T., and Van Gool, L.
Surf: speeded up robust features.
*Computer Vision–ECCV 2006*, (2006), 404–417.

[19] Hartley, R., and Zisserman, A.
***Multiple View Geometry in Computer Vision***, Vol. **2**.
Cambridge, United Kingdom: Cambridge Univ. Press, 2000.

[20] Fischler, M., and Bolles, R.
Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography.
*Communications of the ACM*, **24**, 6 (1981), 381–395.

[21] Xiao, J., Cheng, H., Sawhney, H., and Han, F.
Vehicle detection and tracking in wide field-of-view aerial video.
In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, 2010, 679–684.

[22] Telea, A.
An image inpainting technique based on the fast marching method.
*Journal of Graphics Tools*, **9**, 1 (2004), 23–34.

[23] Nadarajah, S., and Kotz, S.
Estimation methods for the multivariate t distribution.
*Acta Applicandae Mathematicae: An International Survey Journal on Applying Mathematics and Mathematical Applications*, **102**, 1 (2008), 99–118.

[24] Simoncelli, E. P.
Statistical modeling of photographic images.
*In Handbook of Image and Video Processing*. 2005, 431–441.

[25] Lange, K. L., Little, R. J. A., and Taylor, J. M. G.
Robust statistical modeling using the t distribution.
*Journal of the American Statistical Association*, **84** (1989), 881–896.

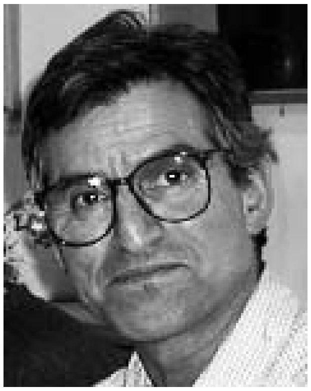
[26] Peel, D., and McLachlan, G.
Robust mixture modelling using the t distribution.
*Statistics and Computing*, **10**, 4 (2000), 339–348.

[27] Kotz, S., and Nadarajah, S.
*Multivariate t Distributions and Their Applications*.
Cambridge, United Kingdom: Cambridge Univ. Press, 2004.

[28] Fan, X., Fan, G., and Havlicek, J.
Generative model for maneuvering target tracking.
*IEEE Transactions on Aerospace and Electronic Systems*, **46**, 2 (2010), 635–655.

**Chad Aeschliman** is an algorithm engineer at Delphi Electronics and Safety, Kokomo, IN. His research interests are in multisensor tracking using both radar and vision with an emphasis on probabilistic methods. He received the B.S., M.S., and Ph.D. degrees from the School of Electrical and Computer Engineering at Purdue University in 2005, 2008, and 2012, respectively.

**Johnny Park** is a research assistant professor in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, IN. His research interests span various topics in distributed sensor networks, computer graphics, computer vision, and robotics. He received B.S., M.S., and Ph.D. degrees from the School of Electrical and Computer Engineering at Purdue University in 1998, 2000, and 2004, respectively.



**Avinash Kak** is a professor of electrical and computer engineering at Purdue University. His research focuses on algorithms, languages, and systems related to wired and wireless camera networks, robotics, computer vision, etc. His coauthored book *Principles of Computerized Tomographic Imaging* was republished as a classic in applied mathematics by the Society of Industrial and Applied Mathematics. His other coauthored book, *Digital Picture Processing*, also considered by many to be a classic in computer vision and image processing, was published by Academic Press in 1982. His more recent books are contributions towards "The Objects Trilogy" he is creating. The first book in the trilogy, *Programming with Objects*, was published in 2003 by John Wiley & Sons. The second, *Scripting with Objects*, was published in 2008, also by Wiley. The last book, to be titled *Designing with Objects,* is currently in the works.