Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2010, Article ID 124681, 22 pages
doi:10.1155/2010/124681

Research Article

Real-Time Multiple Moving Targets Detection from Airborne IR Imagery by Dynamic Gabor Filter and Dynamic Gaussian Detector

Fenghui Yao,[1] Guifeng Shao,[1] Ali Sekmen,[1] and Mohan Malkani[2]

[1] *Department of Computer Science, College of Engineering, Technology and Computer Science, Tennessee State University, 3500 John A Merritt Blvd, Nashville, TN 37209, USA*
[2] *Department of Electrical and Computer Engineering, Tennessee State University, Nashville, TN 37209, USA*

Correspondence should be addressed to Fenghui Yao, fyao@tnstate.edu

Received 1 February 2010; Revised 18 May 2010; Accepted 29 June 2010

Academic Editor: Jian Zhang

This paper presents a robust approach to detect multiple moving targets from aerial infrared (IR) image sequences. The proposed novel method is based on dynamic Gabor filter and dynamic Gaussian detector. First, the motion induced by the airborne platform is modeled by parametric affine transformation and the IR video is stabilized by eliminating the background motion. A set of feature points are extracted and they are categorized into inliers and outliers. The inliers are used to estimate affine transformation parameters, and the outliers are used to localize moving targets. Then, a dynamic Gabor filter is employed to enhance the difference images for more accurate detection and localization of moving targets. The Gabor filter's orientation is dynamically changed according to the orientation of optical flows. Next, the specular highlights generated by the dynamic Gabor filter are detected. The outliers and specular highlights are fused to indentify the moving targets. If a specular highlight lies in an outlier cluster, it corresponds to a target; otherwise, the dynamic Gaussian detector is employed to determine whether the specular highlight corresponds to a target. The detection speed is approximate 2 frames per second, which meets the real-time requirement of many target tracking systems.

1. Introduction

Detection of moving targets in infrared (IR) imagery is a challenging research topic in computer vision. Detecting and localizing a moving target accurately is important for automatic tracking system initialization and recovery from tracking failure. Although many methods have been developed on detecting and tracking targets in visual images (generated by daytime cameras), there exits limited amount of work on target detection and tracking from IR imagery in computer vision community [1]. IR images are obtained by sensing the radiation in IR spectrum, which is either emitted or reflected by the object in the scene. Due to this property, IR images can provide information which is not available in visual images. However, in comparison to the visual images, the images obtained from an IR camera have extremely low signal-to-noise ratio, which results in limited information for performing detection and tracking tasks. In addition, in airborne IR images, nonrepeatability of the target signature, competing background clutter, lack of a priori information, high ego-motion of the sensor, and the artifacts due to weather conditions make detection or tracking of targets even harder. To overcome the shortcomings of the nature of IR imagery, different approaches impose different constrains to provide solutions for a limited number of situations. For instance, several detection methods require that the targets are hot spots which appear as bright regions in the IR images [2–4]. Similarly, some other methods assume that target features do not drastically change over the course of tracking [4–7] or sensor platforms are stationary [5]. However, in realistic target detection scenarios, none of these assumptions are applicable, and a robust detection method must successfully deal with these problems.

This paper presents an approach for robust real-time target detection in airborne IR imagery. This approach has the following characteristics: (1) it is robust in presence of

high global motion and significant texture in background; (2) it does not require that targets have constant velocity or acceleration; (3) it does not assume that target features do not drastically change over the course of tracking. There are two contributions in our approach. The first contribution is the dynamic Gabor filter. In airborne IR video, the whole background appears to be moving because of the motion of the airborne platform. Hence, the motion of the targets must be distinguished from the motion of the background. To achieve this, the background motion is modeled by a global parametric transformation and then motion image is generated by frame differencing. However, the motion image generated by frame differencing using an IR camera is weaker compared to that of a daytime camera. Especially in the presence of significant texture in background, the small error in global motion model estimation accumulates large errors in motion image. This makes it impossible to detect the target from the motion image directly. To solve this problem, we employ a Gabor filter to enhance the motion image. The orientation of Gabor filter is changed from frame to frame and therefore we call it *dynamic Gabor filter*. The second contribution is dynamic Gaussian detector. After applying dynamic Gabor filter, the target detection problem becomes the detection of specular highlights. We employ both specular highlights and clusters of outliers (the feature points corresponding to the moving objects) to detect the target. If a specular highlight lies in a cluster of outliers, it is considered as a target. Otherwise, the Gaussian detector is applied to determine if a specular highlight corresponds to a target or not. The orientation of Gaussian detector is determined by the principal axis of the highlight. Therefore, we call it *dynamic Gaussian detector*.

The remainder of the paper is organized as follows. Section 2 provides a literature survey on detecting moving targets in airborne IR videos. In Section 3, the proposed algorithm is described in detail. Section 4 presents the experimental results. Section 5 gives the performance analysis of the proposed algorithm. Conclusions and future works are given in Section 6.

2. Related Work

For the detection of IR targets, many methods use the ithot spot technique, which assumes that the target IR radiation is much stronger than the radiation of the background and the noise. The goal of these target detectors is then to detect the center of the region with the highest intensity in image, which is called ithot spot [1]. The hot spot detectors use various spatial filters to detect the targets in the scene. Chen and Reed modeled the underlying clutter and noise after local demeaning as a whitened Gaussian random process and developed a constant false alarm rate detector using the generalized maximum likelihood ratio [2]. Longmire and Takken developed a spatial filter based on least mean square (LMS) to maximize the signal-to-clutter ratio for a known and fixed clutter environment [3]. Morin have presented a multistage infinite impulse response (IIR) filter for detecting dim point targets [8]. Tzannes and Brooks presented a generalized likelihood ratio test (GLRT) solution to detect small (point) targets in a cluttered background when both the target and clutter are moving through the image scene [9]. These methods do not work well in presence of significant texture in background because they employ the assumption that that the target IR radiation is much stronger than the radiation of the background and the noise. This assumption is not always satisfied. For instance, Figure 1 shows two IR images with significant texture in background, each contains three vehicles on a road. The IR radiation from asphalt concrete road and street lights is much stronger than that of vehicle bodies, and street lights appear in IR images as ithot spots but vehicles do not. Yilmaz et al. applied fuzzy clustering, edge fusion and local texture energy techniques to the input IR image directly, to detect the targets [1]. This method works well for IR videos with simple texture in background such as ocean or sky. For the IR videos as shown in Figure 1, this method will fail because the textures are complicated and edges are across the entire images. In addition, this algorithm requires an initialization of the target bounding box in the frame where the target first appears. Furthermore, this method can only detect and track a single target. Recently, Yin and Collins developed a method to detect and localize moving targets in IR imagery by forward-backward motion history images (MHI) [10]. Motion history images accumulate change detection results with a decay term over a short period of time, that is, motion history length L. This method can accurately detect location and shape of multiple moving objects in presence of significant texture in background. The drawback of this method is that it is difficult to determine the proper value for motion history length L. Even a well-tuned motion history length works well for one input video, it may not work for other input videos. In airborne IR imagery, the moving objects may be small, and intensity appearance may be camouflaged. To guarantee that the object shape can be detected well, a large L can be selected. But this will lengthen the lag of the target detection system. In this paper, we present a method for target detection in airborne IR imagery, which is motivated by the need to overcome some of the shortcomings of existing algorithms. Our method does not have any assumption on target velocity and acceleration, object intensity appearance, and camera motion. It can detect multiple moving targets in presence of significant texture in background. Section 3 describes this algorithm in detail.

3. Algorithm Description

The extensive literature survey indicates that moving target detection from stationary cameras has been well researched and various algorithms have been developed. When the camera is mounted on an airborne platform, the whole background of the scene appears to be moving and the actual motion of the targets must be distinguished from the background motion without any assumption on velocity and acceleration of the platform. Also, the algorithm must work in real-time, that is, the time-consuming algorithms that repeatedly employ the entire image pixels are not applicable for this problem.

(a) (b)

Figure 1: Two sample IR images with significant textures in background. (a) Frame 98 in dataset1; (b) Frame 0 in dataset 3.

To solve these problems, we propose an approach to perform the real-time multiple moving target detection in airborne IR imagery. This algorithm can be formulated in four steps as follows.

Step 1. Motion Compensation. It consists of the feature point detection, optical flow detection, estimation of the global transformation model parameter, and frame differencing.

Step 2. Dynamic Gabor Filtering. The frame difference image generated in Step 1 is weak, and it is difficult to detect targets from the frame difference image directly. We employ Gabor filter to enhance the frame difference image. The orientation of Gabor filter is dynamically controlled by using the orientation of the optical flows. Therefore, we call it *dynamic Gabor filter*.

Step 3. Specular Highlights Detection. After the dynamic Gabor filtering, the image changes appear as strong intensity in the dynamic Gabor filter response. We call these strong intensity *specular highlights*. The target detection problem then becomes the specular highlight detection. The detector employs the specular highlight point detection and clustering techniques to identify the center and size of the specular highlights.

Step 4. Target Localization. If a specular highlight lies in a cluster of outliers, it is considered as a target. Otherwise, the Gaussian detector is employed for further discrimination. The orientation of the specular highlight is used to control the orientation of the Gaussian detector. Therefore, we call it *dynamic Gaussian detector*.

The processing flow of this algorithm is shown in Figure 2. The following will describe above processing steps in detail.

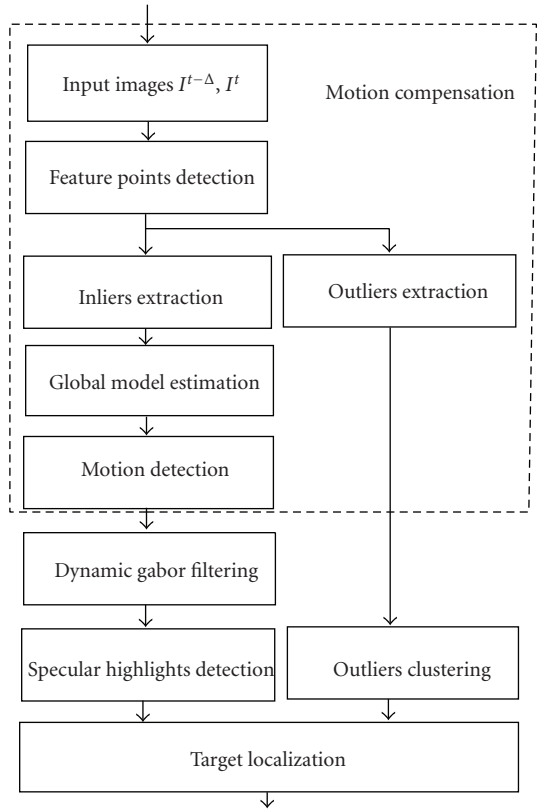


Figure 2: Processing flow of the proposed multiple moving IR targets detection algorithm.

3.1. Motion Compensation. The motion compensation is a technique for describing an image in terms of the transformation of a reference image to the current image. The reference image can be previous image in time. In airborne video, the background is moving over time due to the moving platform. The motion of the platform therefore must be compensated before generating the frame differencing. Two-frame background motion estimation is achieved by fitting a global parametric motion model based on optical flows. To determine optical flows, it needs feature points from two consecutive frames. The motion compensation contains the feature point extraction, optical flow detection, global parametric motion model estimation, and motion detection, which are described below.

3.1.1. Feature Point Extraction. The feature point extraction is used as the first step of many vision tasks such as tracking, localization, image mapping, and recognition. Hence, many feature point detectors exist in literature. Harris corner detector, Shi-Tomasi's corner detector, SUSAN, SIFT, SURF, and FAST are some representative feature point detection algorithms developed over past two decades. Harris corner detector [11] computes an approximation to the second derivative of the sum-of-squared-difference (SSD) between a patch around a candidate corner and patches shifted. The approximation is

$$\mathbf{H} = \begin{pmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{pmatrix}, \tag{1}$$

where angle brackets denote averaging performed over the image patch. The corner response is defined as

$$C = |\mathbf{H}| - k(\text{trace } \mathbf{H})^2, \tag{2}$$

where k is a tunable sensitivity parameter. A corner is characterized by a large variation of C in all directions of the vector (x, y). Shi and Tomasi [12] conclude that it is better to use the smallest eigenvalue of $\mathbf{H}$ as the corner strength function, that is,

$$C = \min(\lambda_1, \lambda_2). \tag{3}$$

SUSAN [13] computes self-similarity by looking at the proportion of pixels inside a disc whose intensity is within some threshold of the center (nucleus) value. Pixels closer in value to the nucleus receive a higher weighting. This measure is known as (the Univalue Segment Assimilating Nucleus) USAN. A low value for the USAN indicates a corner since the center pixel is very different from most of its surroundings. A set of rules is used to suppress qualitatively "bad" features, and then local minima of the SUSANs (Smallest USAN) are selected from the remaining candidates. SIFT (Scale Invariant Feature Transform) [14] obtains scale invariance by convolving the image with a Difference of Gaussians (DoG) kernel at multiple scales, retaining locations which are optima in scale as well as space. DoG is used because it is a good approximation for the Laplacian of a Gaussian (LoG) and much faster to compute. (Speed Up Robust Features) SURF [15] is based on the Hessian matrix, but uses a very basic approximation, just as DoG is a very basic Laplacian-based detector. It relies on integral images to reduce the computation time. (Features from Accelerated Segment Test) FAST feature detector [16] considers pixels in a Bresenham circle of radius r around the candidate point. If n contiguous pixels are all brighter than the nucleus by at least t or all darker than the nucleus by t, then the pixel under the nucleus is considered to be a feature. Although r can, in principle, take any value, only a value of 3 is used (corresponding to a circle of 16 pixels circumference), and tests show that the best value of n is 9.

For our real-time IR targets detection in airborne videos, it needs a fast and reliable feature point detection algorithm. However, the processing time depends on image contents. To investigate the processing time, we employ the synthesized test image as shown in Figure 3. The test image is 320×256 full color image which contains 252 (14 row, 18 column) 16×16 rectangles.

Table 1: Feature point detectors and their processing time for the synthesized image in Figure 3.

Feature point detector	Processing time (ms)	Number of feature points
Harris corner detector	47	82
Shi and Tomasi's corner detector	31	102
SUSAN corner detector	32	250
SIFT	655	714
SURF	344	355
FAST	<1	1424

The color of the rectangle is randomly determined. The number of ground truth corners in this image is 285 (15×19). The experiments are performed on a Windows Vista machine mounted with a 2.33 GHz Intel Core 2 CPU and 2 GB memory. The corners detected by above mentioned algorithms are marked by small red rectangles in Figures 3(a) to 3(f). The processing time and the detected corner number are listed in Table 1. According to the processing time in Table 1 and feature point detection results in Figure 3, we obtain the following conclusions. (i) SIFT and SURF need heavy computation, and they output many wrong corners (refer to Figures 3(d) and 3(e)). They are not suitable for real-time target detection algorithms. (ii) The processing time for FAST is less than 1 ms. This is really attractive. However, it generates many redundant feature points (refer to Figure 3(f)) in the local area of the real corner. The total number of the corners detected is 1424, which is much bigger than the number of the ground truth corners. And further, we tested this algorithm by using images from airborne IR camera. It fails to extract feature points for many images. FAST is not proper for feature point detection in airborne imagery. (iii) Harris corner detector is fast. But it missed many ground truth corners. It is not candidate for our algorithm. (iv) The processing time for SUSAN and Shi-Tomasi's corner detector are almost the same. SUSAN detects more ground truth corner than Shi-Tomasi's method for this synthesized image. Further, to investigate the robustness of SUSAN and Shi-Tomasi's corner detector, another 640×512 full color test image is synthesized. This test image contains 252 (14 row, 18 column) randomly colored triangles, which form 518 ($37 \times 13 + 18$ (top) + 19 (bottom)) ground truth corners. The experiment result is shown in Figure 4. Shi-Tomasi's method detected 265 corner points, as marked by small red rectangles in Figure 4(a), which are all ground truth corner points. SUSAN detected 598 corner points, as depicted by small red rectangles in Figure 4(b), which contain 80 false corner points (refer to the two close small rectangles at the top vertex of some triangles). These false

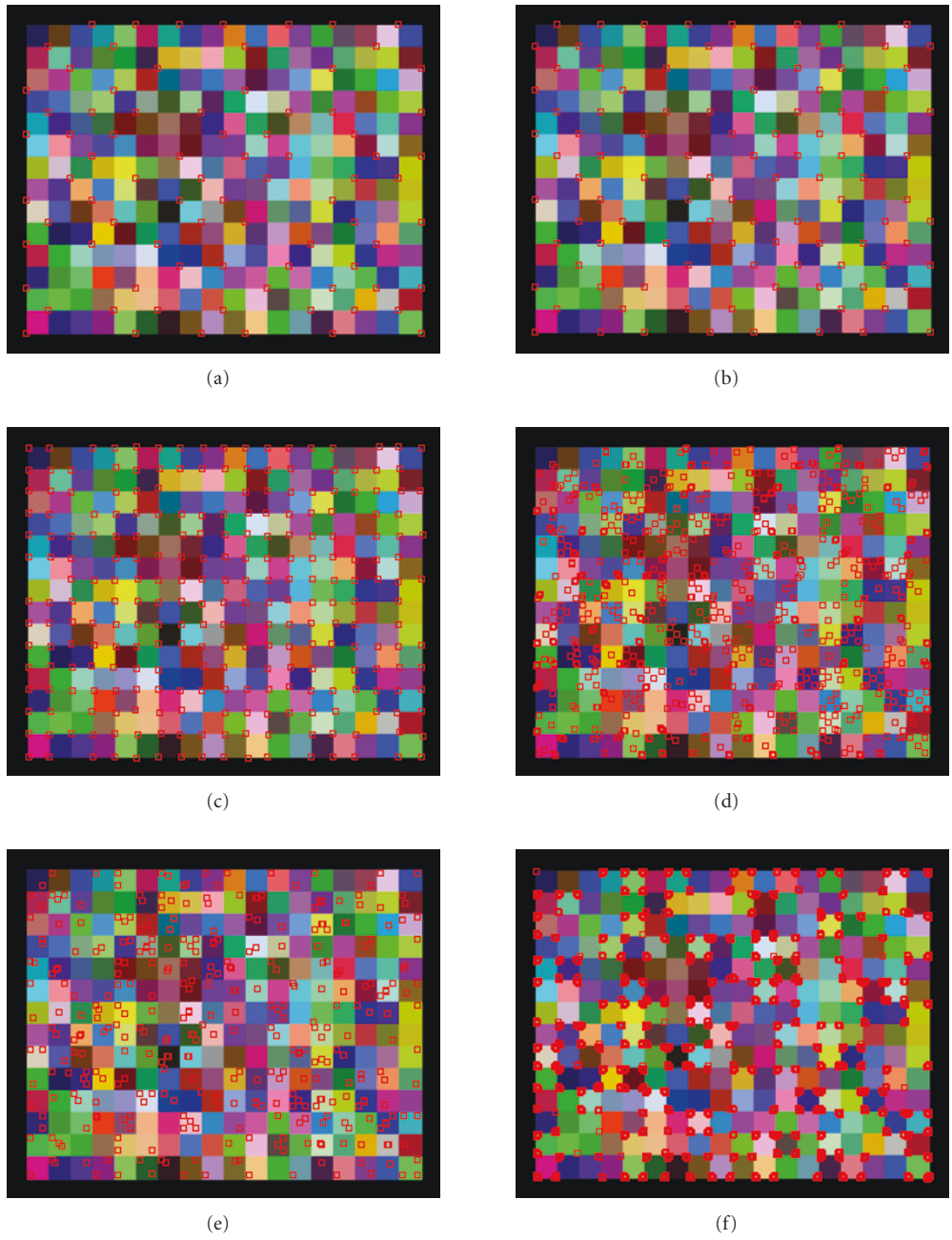
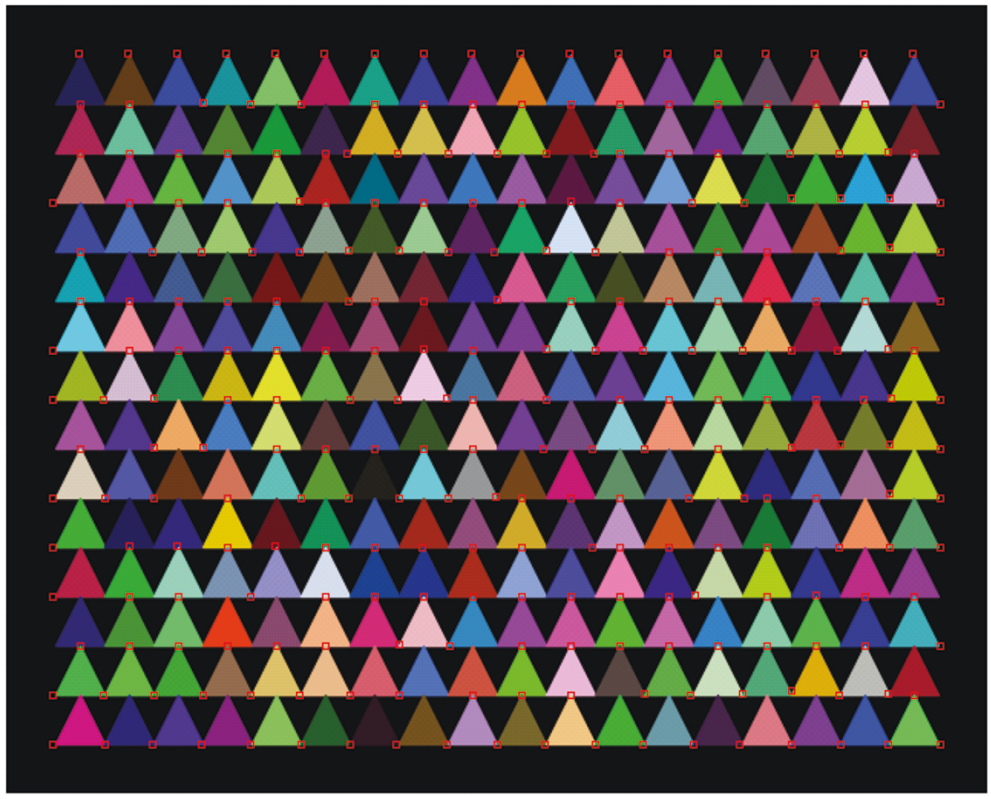


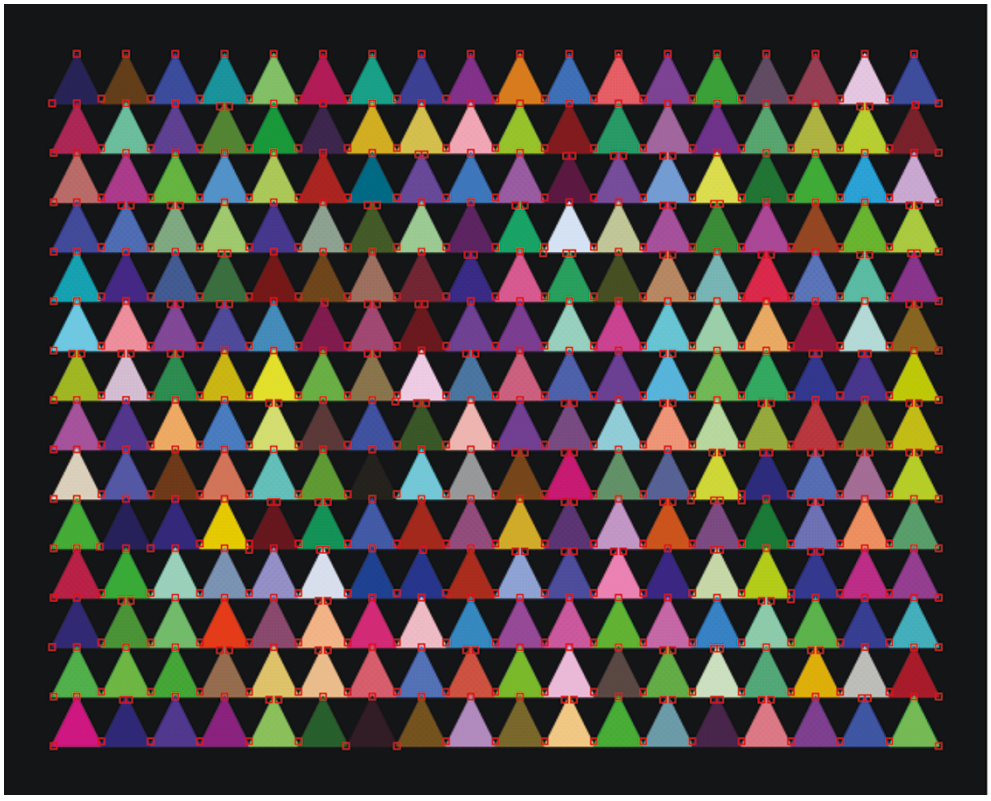
Figure 3: Feature point detection results for six algorithms; (a) Harris corner detector, (b) Shi and Tomasi's corner detector, (c) SUSAN corner detector, (d) SIFT feature point detector, (e) SURF feature point detector, and (f) FAST corner detector.

corner points will deteriorate the postprocessing. Furthermore, the robustness of these two detectors is investigated by using the IR images from airborne IR camera, as shown in Figure 1, in which (a) shows an IR image with complicated content, and (b) relatively simple contents. The experiment results are shown in Figure 5, in which (a) shows the corner points detected by Shi-Tomasi's method, and (b) by SUSAN. Although it is difficult to tell which ones are truth corner points in Figures 5(a) and 5(b), it is obvious that (b) contains many false corner points. From these results, it is clear that Shi-Tomasi's method is more robust than SUSAN. For more details about performance evaluation of corner detection algorithms, readers are referred to [17].

From above results and discussion, this paper employs Shi-Tomasi's method to detect feature points. For two input images, let $P^{t'} = \{p_1^{t'}, \ldots, p_M^{t'}\}$ and $P^t = \{p_1^t, \ldots, p_N^t\}$ denote the feature points detected from $I^{t'}$ and I^t, respectively, where $t' = t - \Delta$, $p_i^{t'} = (x_i^{t'}, y_i^{t'})$, $p_j^t = (x_j^t, y_j^t)$, $i = 1, 2, \ldots, M$ and $j = 1, 2, \ldots, N$. In the following, $I^{t'}$ is called previous image, I^t is called current image or reference image. These feature points are used for optical flow detection.

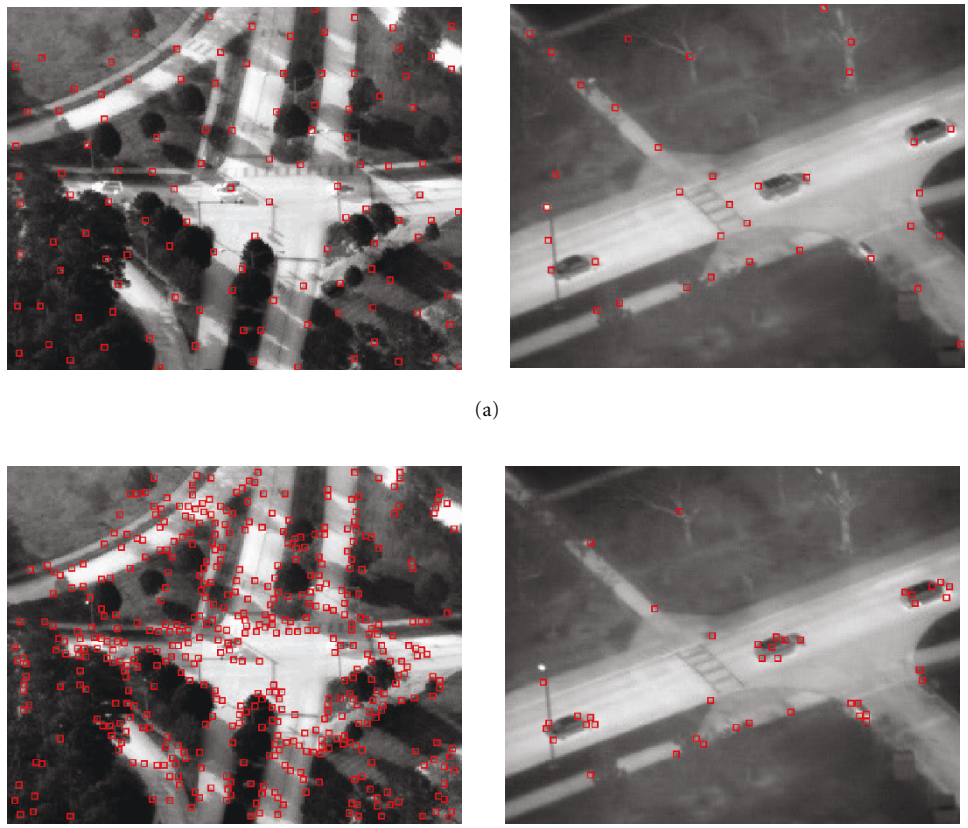


(a)



(b)

Figure 4: Feature points detected by (a) Shi and Tomasi's corner detector, and (b) SUSAN corner detector, for 640×512 color image.



(a)

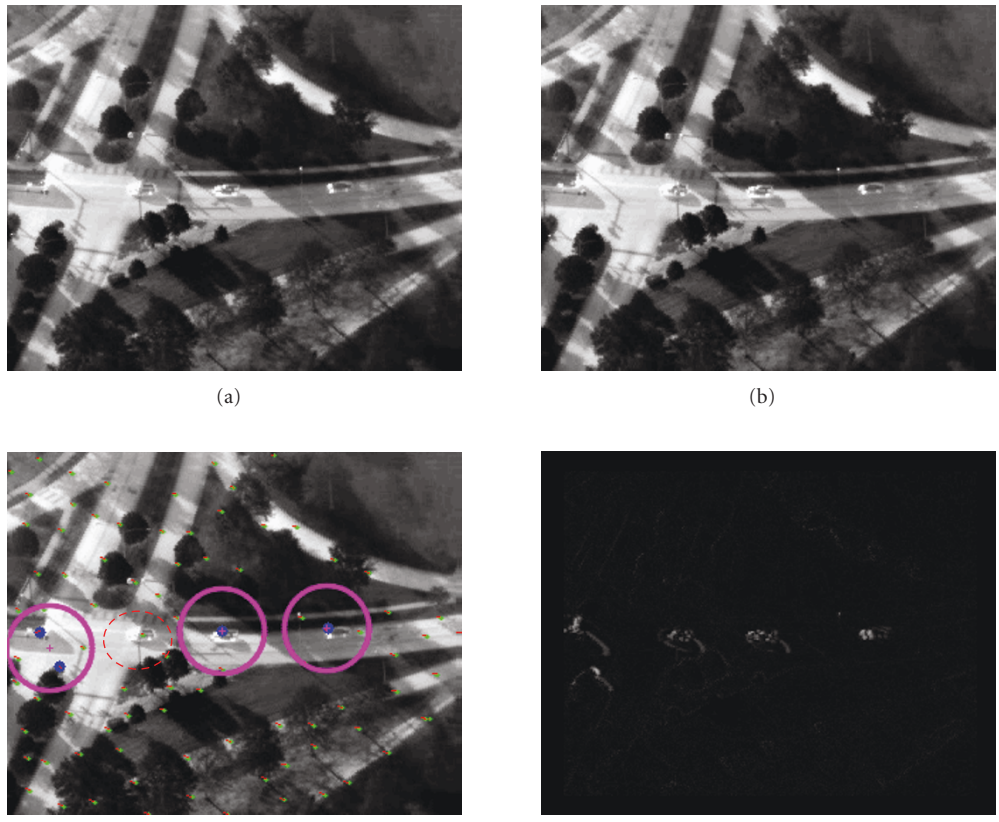
(b)

Figure 5: Feature points detected by (a) Shi and Tomasi's corner detector and, (b) SUSAN corner detector, for 640×512 color image.

3.1.2. Optical Flow Detection. The optical flow is the apparent motion of the brightness patterns in the image [18]. In our algorithm, the feature points obtained in previous section are used as the brightness patterns in the definition of optical flow [18]. That is, the task for optical detection is to find the corresponding feature point p_j^t in frame I^t, for the feature point $p_i^{t'}$ in frame $I^{t'}$, where $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, N$.

There are many optical flow detection algorithms. Recently there are several new developments on this topic. Black and Anandan [19] proposed a framework based on robust estimation that addresses violations of the brightness constancy, and spatial smoothness assumptions caused by multiple motions. Bruhn et al. [20] developed a differential method that combines local methods such as the Lucas-Kanade's technique and global methods such as the Horn-Schunck's approach. Zitnick et al.'s method is based on statistical modeling of an image pair using constraints on appearance and motion [21]. Bouguet's method is the pyramidal implementation of the Lucas-Kanade's technique [22]. The evaluation results of these four algorithms show that Bouguet's method is the best for the interpolation task [23]. As measured by average rank, the best performing algorithms for the ground truth motion are Bruhn et al. and Black and Anandan.

In our algorithm, we employed Bouguet's method for optical flow detection. Figures 6(a) and 6(b) show two input images, $I^{t'}$ and I^t. The frame interval, Δ, is an important parameter that affects the quality of the optical flow. If it is too small, the displacement between two consecutive frames is also too small (close to zero). In this case, the optical flow cannot be precisely detected. If it is too large, the error in the process of finding the corresponding feature points in the consecutive frame increases. In this case, the optical flow also cannot be precisely detected. In our airborne videos, the helicopter flew at very high altitude, and the displacement between consecutive image frames is relatively small. To speed up the algorithm, Δ is set at 3. The experiments show our algorithm works well for $\Delta = 1, \ldots, 4$. Figure 6(c) shows the optical flows detected from the feature points $\{p_1^{t'}, \ldots, p_M^{t'}\}$ and $\{p_1^t, \ldots, p_N^t\}$, where the optical flow are marked by red line segments, and the endpoints of the optical flows are marked by green dots. Let $F^{t't} = \{\vec{F}_1^{t't}, \vec{F}_2^{t't}, \ldots, \vec{F}_K^{t't}\}$ denote the detected optical flows. Note that the start point of ith optical flow, $\vec{F}_i^{t't}$, belongs to set $P^{t'}$, and the endpoint belongs to set P^t. For the feature points in set $P^{t'}$ and P^t, from which no optical flow is detected, they are filtered out. Therefore, after this filtering operation, the number of feature points in two sets, $P^{t'}$ and P^t, becomes the same with the number of optical flows in optical flow set $F^{t't}$, that is,



(a) (b) (c) (d)

Figure 6: Optical flow detection and frame differencing results. (a) and (b) Two input images; (c) Detected optical flows; (d) Frame difference.

K. They are rewritten as $P^{t'} = \{p_1^{t'}, \ldots, p_K^{t'}\}$, and $P^t = \{p_1^t, \ldots, p_K^t\}$, accordingly. In the following, in order to make the description easier, we consider that the feature points in P^t is sorted so that the start point and endpoint of $\vec{F}_i^{t't}$ are consequently $p_i^{t'} \in P^{t'}$ and $p_i^t \in P^t$, respectively. That is, $\vec{F}_i^{t't}$ means $\vec{p}_i^{t'}\, p_i^t$. Note that there is no need to perform this sorting in the implementation because the optical flow $\vec{F}_i^{t't}$ holds the index information for the feature points in set $P^{t'}$ and P^t.

3.1.3. Global Parametric Motion Model Estimation

(A) Transformation Model Selection. Motion compensation requires finding the coordinate transformation between two consecutive images. It is important to have a precise description of the coordinate transformation between a pair of images. By applying the appropriate transformations via a warping operation and subtracting the warped images from the reference image, it is possible to construct the frame difference that contains image changes (motion image).

There exist many publications about motion parameter estimation which can be used for motion compensation. A coordinate transformation maps the image coordinates, $\mathbf{x}' = (x', y')^T$, to a new set of coordinates, $\mathbf{x} = (x, y)^T$. Generally, the approach to finding the coordinate transformation relies on assuming that it will take one of the following six models, (1) translation, (2) affine, (3) bilinear, (4) projective, (5) pseudo perspective, and (6) biquadratic, and then estimating the two to twelve parameters in the chosen models.

The translation model is based on the assumption that the coordinate transformation between frames is only translation. Although it is easy to implement, it is very poor to handle large changes due to camera rotation, panning, and tilting. This model is not suitable for our purpose. On the other hand, the parameter estimation in 8-parameter projective model and 12-parameter biquadratic model becomes complicated. Time-consuming models are not suitable for the real-time applications. Therefore, our algorithm does not employ these two models, neither. The following investigates affine, bilinear, and pseudo perspective models. Let (x', y') denote the feature point coordinates in previous image, and (x, y) the coordinates in the current image. Affine model is given by

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} a_5 \\ a_6 \end{pmatrix}. \tag{4}$$

The bilinear model is defined as

$$\begin{aligned} x &= a_1 x' + a_2 y' + a_3 + a_4 x' y,' \\ y &= a_5 x' + a_6 y' + a_7 + a_8 x' y'. \end{aligned} \tag{5}$$

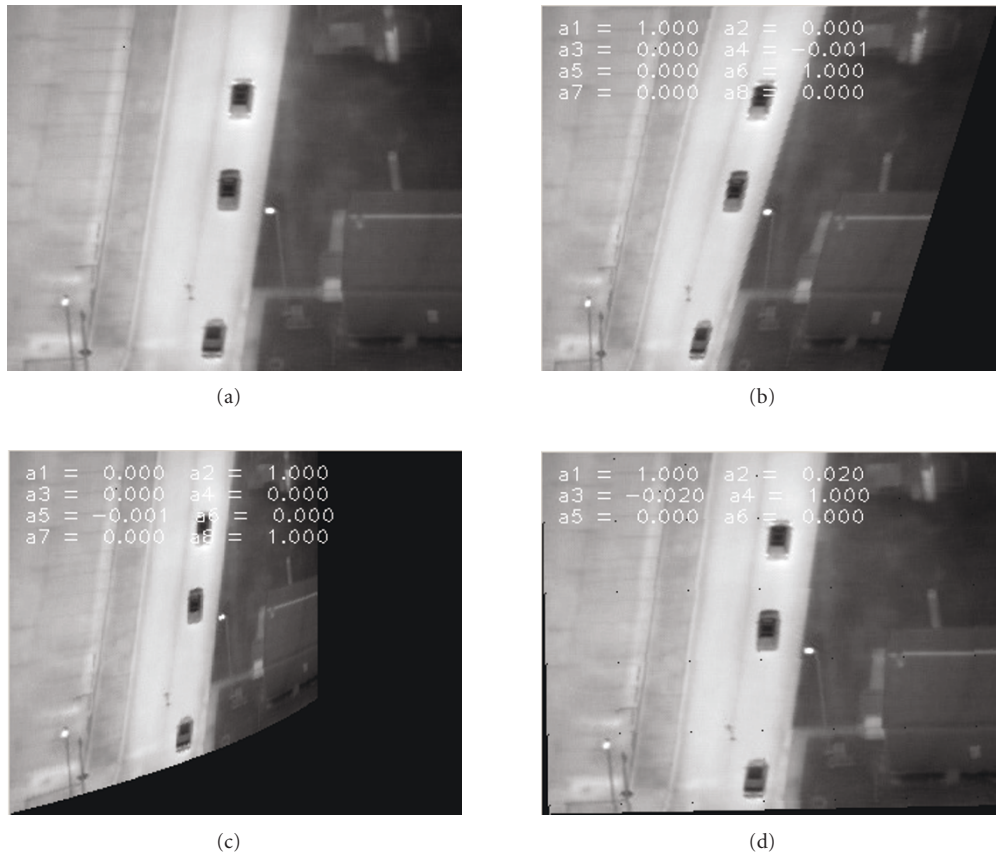


Figure 7: Image transformation results; (a) Original image, (b) Image generated by bilinear model, (c) Image generated by pseudo perspective model, and (d) Image generated by Affine model.

The pseudo perspective model is given by

$$x = a_1 + a_2x' + a_3y' + a_4x'y' + a_5x'^2,$$
$$y = a_4y'^2 + a_5x'y' + a_6 + a_7x' + a_8y'. \quad (6)$$

Figure 7(b) shows the transformed image for the image in Figure 7(a), by applying the bilinear transformation with parameters of $a_1 = a_6 = 1.0$, $a_4 = -0.001$, and others (a_2, a_3, a_5, a_7, a_8) equal to 0.0. For this set of parameters, if a_4 is also set to 0.0, no transformation is applied to the original image. However, if a_4 is set at -0.001, which corresponds to the fact that a_4 contains 1‰ error, the output image is greatly deformed. Similarly, Figure 7(c) shows the transformed image for the image in (a), by applying pseudo perspective transformation with parameters of $a_2 = a_8 = 1.0$, $a_5 = -0.001$, and others (a_1, a_3, a_4, a_6, a_7) equal to 0.0. For this set of parameters, if a_5 is also set to 0.0, no transformation is applied to the original image. However, if a_5 is set at -0.001, which corresponds to the fact that a_5 contains 1‰ error, the output image is greatly deformed. These results show that bilinear model and pseudo perspective model are sensitive to parameter errors. A small error in parameter estimation may cause huge difference in the transformed images. We used the images from airborne IR camera to test the frame difference based on these two models, the results are poor. In contrast, the affine transformation contains translation, rotation, and scale although it cannot capture camera pan and tilt motion. However, in the system to generate airborne videos, cameras are usually mounted on the moving platform such as a helicopter or an UAV (unmanned aerial vehicle). In this case, there is no camera pan and tilt motion. Figure 7(d) shows the transformed image for the image in (a), by applying affine transformation with parameters of $a_1 = a_4 = 1.0$, $a_2 = 0.02$, $a_3 = -0.02$, and $a_5 = a_6 = 1.0$. This setting is corresponding to that a_2 and $a_3$contain 2% error, respectively. Comparing the results in Figures 7(b), 7(c), and 7(d), we can say that even the parameter estimation error in affine transformation is 20 times larger than the error in bilinear transformation or pseudo perspective transformation (2% in affine transform, to 1‰ in bilinear transformation and pseudo perspective transformation), the image deformation is still tolerable (see Figure 7(d)). This result shows that the affine model is robust to the parameter errors. Therefore, in our algorithm, we employ affine model for motion detection.

(B) Inliers/Outliers Separation. The feature points $P^{t'} = \{p_1^{t'}, \ldots, p_K^{t'}\}$ and $P^t = \{p_1^t, \ldots, p_K^t\}$, obtained in Section 3.1.2, are used to estimate six parameters in (4). The corresponding relations for the feature points in set $P^{t'}$ and P^t are determined by the optical flows, $F^{t't} = \{\vec{F}_1^{t't}, \vec{F}_2^{t't}, \ldots, \vec{F}_K^{t't}\}$. For the feature points in set $P^{t'}$ and P^t, some of them are associated with the background,

and some with the moving targets. The feature points associated with the moving targets are called *outliers*. Those associated with the background are called *inliers*. To detect the motion image (that is, image changes) for two consecutive images, the previous image is wrapped to the current image by performing affine transformation, and then the frame difference image can be obtained by image subtraction. This operation needs precise transformation model. To estimate the transformation model precisely, the outliers must be excluded. That is, the feature points need to be categorized into outliers and inliers, and only the inliers are used to estimate the affine transformation parameters. The inliers/outliers are separated automatically by the following algorithm.

Inliers/Outliers Separation Algorithm. (i) Using all feature points in set $P^{t'}$ and P^t, 6-paramers in affine model are primarily estimated by least-square method [24]. That is, $a_1, \ldots, a_6$ are obtained by solving the equation below.

$$\begin{pmatrix} \sum x_i^{t'} x_i^{t'} & \sum x_i^{t'} y_i^{t'} & 0 & 0 & \sum x_i^{t'} & 0 \\ \sum x_i^{t'} y_i^{t'} & \sum y_i^{t'} y_i^{t'} & 0 & 0 & \sum y_i^{t'} & 0 \\ \sum x_i^{t'} & \sum y_i^{t'} & 0 & 0 & K & 0 \\ 0 & 0 & \sum x_i^{t'} x_i^{t'} & \sum x_i^{t'} y_i^{t'} & 0 & \sum x_i^{t'} \\ 0 & 0 & \sum x_i^{t'} y_i^{t'} & \sum y_i^{t'} y_i^{t'} & 0 & \sum y_i^{t'} \\ 0 & 0 & \sum x_i^{t'} & \sum y_i^{t'} & 0 & K \end{pmatrix} \times \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} \sum x_i^t x_i^{t'} \\ \sum x_i^t y_i^{t'} \\ \sum x_i^t \\ \sum y_i^t x_i^{t'} \\ \sum y_i^t y_i^{t'} \\ \sum y_i^t \end{pmatrix}, \tag{7}$$

where the summation $\sum$ represents $\sum_{i=1}^{K}$, $(x_i^{t'}, y_i^{t'}) \in P^{t'}$, and $(x_j^t, y_j^t) \in P^t$. Let $\mathbf{A}'$ denote the affine model obtained from (7).

(ii) Applying $\mathbf{A}'$ to the feature points in set $P^{t'}$, the transformed feature points are obtained, which are denoted by $\widetilde{P}^{t'} = \{\widetilde{p}_1^{t'}, \ldots, \widetilde{p}_K^{t'}\}$. The error between the transformed feature points and their corresponding feature points in P^t is defined as

$$E_i = \left\| \widetilde{p}_i^{t'} - p_i^t \right\|, \tag{8}$$

where $\|\cdot\|$ means norm operation, $i = 1, \ldots, K$.

(iii) Inliers/outliers are discriminated according to the following criteria:

$$\widetilde{p}_i^{t'} \text{ and } p_i^t \text{ are} \begin{cases} \text{inliers}, & \text{if } E_i \le E_T, \\ \text{outliers}, & \text{if } E_i > E_T, \end{cases} \tag{9}$$

$$E_T = \frac{\lambda_E}{K} \sum_{i=1}^{K} E_i,$$

where λ_E is the weighting coefficient. The value of λ_E depends on the size of the moving target. The larger the moving target is, the smaller the value of λ_E needs to be. In airborne IR videos, the moving target is relatively small because the observer is at high altitude, λ_E can be relatively large. Experiments show that the value of λ_E can be in the range of 1.0 to 1.4, currently is set at 1.3.

The algorithm described above is based on the fact that for the feature points belonging to the moving target, the error defined in (8) is large because the corresponding feature points are moving accompanied with the moving target. Figure 6(c) shows the inliers/outliers separation for the feature points detected from the input images in Figures 6(a) and 6(b). The outliers are marked by blue dots. After this operation, $P^{t'}$ is separated to inliers set $P_{\text{in}}^{t'} = \{p_1^{t'}, \ldots, p_{K_{\text{in}}}^{t'}\}$, and outliers set $P_{\text{out}}^{t'} = \{p_1^{t'}, \ldots, p_{K_{\text{out}}}^{t'}\}$, P^t is separated to inliers set $P_{\text{in}}^t = \{p_1^t, \ldots, p_{K_{\text{in}}}^t\}$ and outliers set $P_{\text{out}}^t = \{p_1^t, \ldots, p_{K_{\text{out}}}^t\}$, and $F^{t't}$ is separated to optical flows $F_{\text{in}}^{t't} = \{\vec{F}_1^{t't}, \vec{F}_2^{t't}, \ldots, \vec{F}_{K_{\text{in}}}^{t't}\}$ corresponding to inliers, and optical flows $F_{\text{out}}^{t't} = \{\vec{F}_1^{t't}, \vec{F}_2^{t't}, \ldots, \vec{F}_{K_{\text{out}}}^{t't}\}$ corresponding to outliers. And the following relations hold

$$\begin{aligned} P^{t'} &= P_{\text{in}}^{t'} + P_{\text{out}}^{t'}, \\ P^t &= P_{\text{in}}^t + P_{\text{out}}^t, \\ F^{t't} &= F_{\text{in}}^{t't} + F_{\text{out}}^{t't}. \end{aligned} \tag{10}$$

That is, the first and second formula in (10) show that the total feature points detected from the previous image frame and current image frame are separated into inliers and outliers, respectively. Correspondingly, the optical flows are also separated into two classes, optical flows belonging to inliers and those belonging to outliers, as indicated by the third formula in (10).

Again, in the following, to make the description easier, let us assume $p_1^{t'} \in P_{\text{in}}^{t'}$ corresponds to $p_1^t \in P_{\text{in}}^t$, and $p_1^{t'} \in P_{\text{out}}^{t'}$ to $p_1^t \in P_{\text{out}}^t$ and so on. The actual implementation does not need this assumption because the optical flows hold the feature point correspondence (refer to Section 3.1.2). Inliers are used in the affine model parameter estimation below, and in dynamic Gabor filter (refer to Section 3.2.2). Outliers are used in target localization (refer to Section 3.4.2).

(C) Affine Transformation Parameter Estimation. There are the six parameters in affine transformation. It needs three pairs of feature points in $P_{\text{in}}^{t'}$ and P_{in}^t to estimate these six parameters. However, affine model determined only by using three pairs of feature points might not be accurate. To determine these parameters efficiently and precisely, our method employs the following algorithm.

Affine Model Estimation Algorithm. (1) Randomly choose L triplet inliers pairs from $P_{\text{in}}^{t'}$ and P_{in}^t, respectively. For a triplet

inliers pair $(p_i^{t'}, p_{i+1}^{t'}, p_{i+2}^{t'}) \in P_{\text{in}}^{t'}$ and $(p_i^t, p_{i+1}^t, p_{i+2}^t) \in P_{\text{in}}^t$, an affine model determined by solving the following equation:

$$\begin{pmatrix} x_i^{t'} & y_i^{t'} & 0 & 0 & 1 & 0 \\ x_{i+1}^{t'} & y_{i+1}^{t'} & 0 & 0 & 1 & 0 \\ x_{i+2}^{t'} & y_{i+2}^{t'} & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i^{t'} & y_i^{t'} & 0 & 1 \\ 0 & 0 & x_{i+1}^{t'} & y_{i+1}^{t'} & 0 & 1 \\ 0 & 0 & x_{i+2}^{t'} & y_{i+2}^{t'} & 0 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \begin{pmatrix} x_i^t \\ x_{i+1}^t \\ x_{i+2}^t \\ y_i^t \\ y_{i+1}^t \\ y_{i+2}^t \end{pmatrix}, \tag{11}$$

where $(x_i^{t'}, y_i^{t'}) \in P_{\text{in}}^{t'}$, and $(x_j^t, y_j^t) \in P_{\text{in}}^t$, and $i = 1, 3, 6, \ldots, 3L$. Let $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_L)$ represent these L Affine models. They are used to determine the best affine model below.

(2) Apply $\mathbf{A}_s \in \mathbf{A}$ to the previous image $I^{t'}$ and feature points in $P_{\text{in}}^{t'}$. It generates the transformed image $I_A^{t'}$ and transformed feature points $\widetilde{P}_{\text{in}}^{t'} = \{\widetilde{p}_1^{t'}, \ldots, \widetilde{p}_{K_{\text{in}}}^{t'}\}$. The local area correlation coefficient (LACC) is used to determine whether two feature points are matched. The LACC is given by

$$c_{ij} = \sum_{k=-n}^{n} \sum_{l=-m}^{m} \times \frac{\left[I_A^{t'}\left(\widetilde{x}_i^{t'} + k, \widetilde{y}_i^{t'} + l\right) - \overline{I}_A^{t'}\left(\widetilde{x}_i^{t'}, \widetilde{y}_i^{t'}\right)\right]}{(2n+1)(2m+1)\sqrt{\sigma_i\left(I_A^{t'}\right) \times \sigma_j(I^t)}} \times \frac{\left[I^t\left(x_j^t + k, y_j^t + l\right) - \overline{I}^t(x_i^t, y_i^t)\right]}{(2n+1)(2m+1)\sqrt{\sigma_i\left(I_A^{t'}\right) \times \sigma_j(I^t)}}, \tag{12}$$

where $I_A^{t'}$ and I^t are the intensities of the two images, $(\widetilde{x}_i^{t'}, \widetilde{y}_i^{t'})$ and (x_i^t, y_i^t) the ith and jth feature points to be matched, m and n the half-width and half-length of the matching window,

$$\overline{I}(x, y) = \frac{\sum_{k=-n}^{n} \sum_{l=-m}^{m} [I(x + k, y + l)]}{(2n+1)(2m+1)} \tag{13}$$

is the average intensity of the window, and

$$\sigma = \sqrt{\frac{\sum_{k=-n}^{n} \sum_{l=-m}^{m} I^2(x + k, y + l)}{(2n+1)(2m+1)} - \overline{I}^2(x, y)} \tag{14}$$

is the standard variance of the image in matching window. c_{ij} ranges from -1 to 1, indicating the similarity from smallest to largest. Once again, as mentioned in Section 3.1.3.(B), the optical flows keep the corresponding relation for ith feature point in $P_{\text{in}}^{t'}$ and jth feature point in P_{in}^t. For simplifying description, we just say the feature points in $\widetilde{P}_{\text{in}}^{t'}$ are matched to those in P_{in}^t one to one, starting from 1 to K_{in}. Therefore, c_{ij} can be rewritten as c_{ii}. The evaluation function for affine model $\mathbf{A_s}$ is defined by

$$E_s = \sum_{i=1}^{K_{\text{in}}} c_{ii}, \tag{15}$$

where $s = 1, 2, \ldots, L$.

(3) The affine model $\mathbf{A}_b \in \mathbf{A}$, whose evaluation value is maximal, that is, $E_b = \max(E_1, E_2, \ldots, E_L)$, is selected as the best affine model in our algorithm. $\mathbf{A}_b$ is used for image change detection below.

In above framework, there are two affine transformation estimations. The first one is to choose L sets of affine transformations by employing the inliers detected in Section 3.1.3.(B). The second one is to estimate the best affine transformation by calculating the matching measure according to (15). In this framework, the influence of outliers can be determined as follows [25]. The probability p that at least one data set of three points belongs to the inliers, is derived from,

$$p(\varepsilon, q, L) = 1 - \left\{1 - [(1 - \varepsilon)q]^3\right\}^L, \tag{16}$$

where $\varepsilon(<0.5)$ is the ratio of moving target regions to the whole image, q is the probability that the corresponding points are inliers. The probability that this algorithm picks up the outliers is $1 - p$. For example, $p \approx 0.993$ when $\varepsilon = 0.3$, $q = 0.7$, and $L = 40$, then $1 - p = 0.007$. That is, the probability that the outliers will influence the affine transformation estimation is very low, if the moving targets constitute a small area (i.e., less than 50%). In airborne video camera, this requirement can be easily satisfied.

(D) Image Changes Detection. Here, in airborne imagery, the image changes mean changes caused by the moving targets. We call image changes *motion images*. The previous image is transformed by the best affine model $\mathbf{A}_b$, and subtract from the current image. That is, the frame difference is generated as follows:

$$I_{\text{diff}} = \left| I^t - \mathbf{A}_b \times I^{t'} \right|, \tag{17}$$

where $I^{t'}$ and I^t is previous image and current image, respectively. Figure 8(d) shows the frame difference image generated by (17) from two input images in Figures 6(a) and 6(b).

3.2. Dynamic Gabor Filter

3.2.1. Problems of the Thresholding Algorithms. To detect the targets, the motion image needs to be binarized. Figure 8 shows the binarization results for the frame difference image in Figure 6(d) by employing three binarization algorithms. Figures 8(a) and 8(b) show the results for a fixed threshold at 10 and 30, respectively. Figure 8(c) shows the output of the adaptive thresholding algorithm based on *mean C*, where the window size is 5×5 and the constant C is set at 5.0. Figure 8(d) shows the output of Gaussian adaptive thresholding algorithm, where the window size is 5×5 and the constant C is set at 10.0. From these binary images, it is difficult to detect targets. Although by applying some morphological operations such as dilation and erosion techniques, it is possible to detect targets from some frame difference images. However for video sequence processing, this method is not stable. To solve this problem, we need some technique to enhance the frame difference image.

Image enhancement is the improvement of digital image quality (e.g., for visual inspection or for machine analysis),

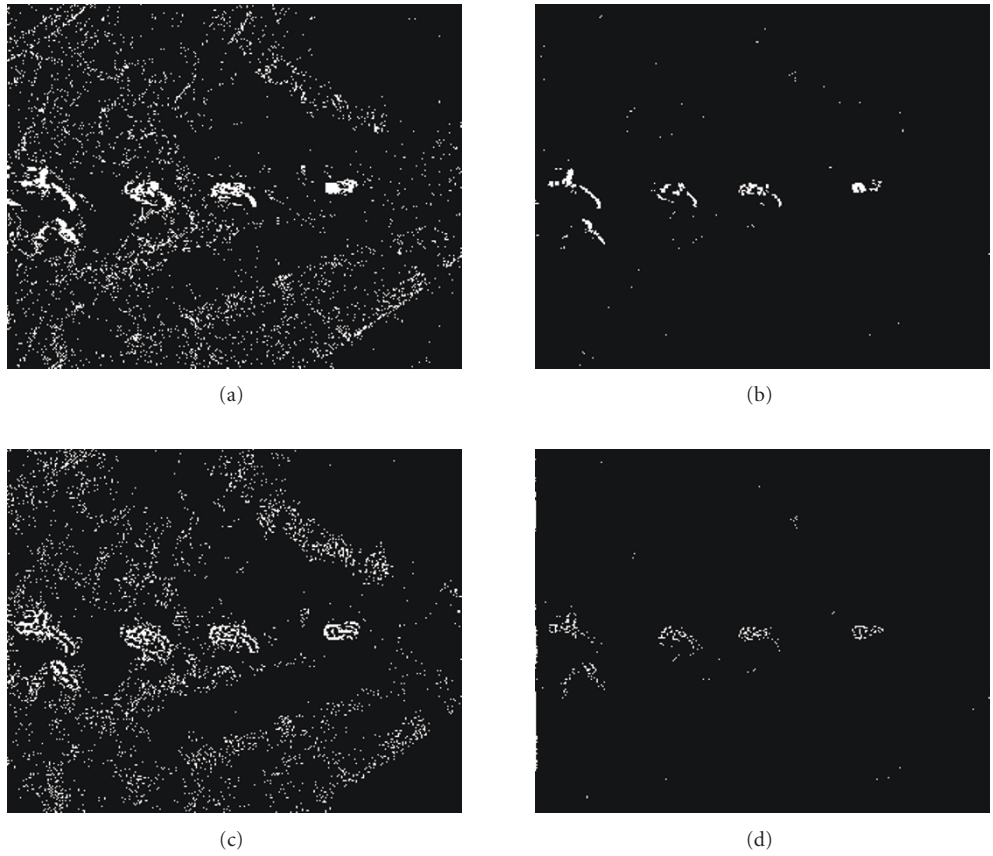


Figure 8: Binary images generated by applying different thresholding algorithms to frame difference image in Figure 6(d); (a) Fixed threshold = 10; (b) Fixed threshold = 30; (c) Adaptive thresholding by using *mean-C* with a 5×5 window and C is set at 5.0; (d) Gaussian adaptive thresholding with a 5×5 window and C is set at 10.0.

without knowledge about the source of degradation. Many different, often elementary and heuristic methods are used to improve images in some sense. A literature survey is given in [26]. Advanced image enhancement algorithms employ spatial filter, neural network, cellular neural network, and fuzzy filter. However, these methods are computationally heavy. They are not suitable for real-time target detection. In our algorithm, we employ dynamic Gabor filter.

3.2.2. Dynamic Gabor Filter. Gabor function has been recognized as a very useful tool in computer vision and image processing, especially for texture analysis, due to its optimal localization properties in both spatial and frequency domain. There are many publications on its applications since Gabor proposed the 1D Gabor function [27]. The family of 2D Gabor filters was originally presented by Daugman [28] as a framework for understanding the orientation-selective and spatial–frequency-selective receptive field properties of neurons in the brain's visual cortex, and then was further mathematically elaborated [29]. The 2D Gabor function is a harmonic oscillator, composed of a sinusoidal plane wave of a particular frequency and orientation, within a Gaussian envelope. Gabor wavelets are hierarchically arranged, Gaussian-modulated sinusoids. The Gabor-wavelet transform of a two-dimensional visual field generates a four-dimensional field: two of the dimensions are spatial, the other two represent spatial frequency and orientation. A Gabor wavelet is defined as

$$\psi_{\mu,\nu}(z) = \frac{\left\|k_{\mu,\nu}\right\|^2}{\sigma^2} e^{-\|k_{\mu,\nu}\|^2 \times \|z\|^2 / 2\sigma^2} \left[e^{ik_{\mu,\nu}z} - e^{-\sigma^2/2}\right], \quad (18)$$

where $z = (x, y)$ is the point with the horizontal coordinate x and the vertical coordinate y. The parameters μ and ν define the orientation and scale of the Gabor kernel, $\|\cdot\|$ denotes the norm operator, and σ is related to the standard derivation of the Gaussian window in the kernel and determines the ratio of the Gaussian window width to the wavelength. The wave vector $k_{\mu,\nu}$ is defined as follows

$$k_{\mu,\nu} = k_\nu e^{i\phi_\mu}, \quad (19)$$

where $k_\nu = k_{\max}/f^\nu$ and $\phi_\mu = \pi\mu/8$, $k_{\max}$ the maximum frequency, and f^ν is the spatial frequency between kernels in frequency domain.

The Gabor kernels in (18) are all self-similar since they can be generated from one kernel (a mother wavelet) by dilation and rotation via the wave vector $k_{\mu,\nu}$. Each kernel is a product of a Gaussian envelope and a complex plane wave. The first term $e^{ik_{\mu,\nu}z}$ in the square bracket in (18) controls the oscillatory part of the kernel and the second term $e^{-\sigma^2/2}$

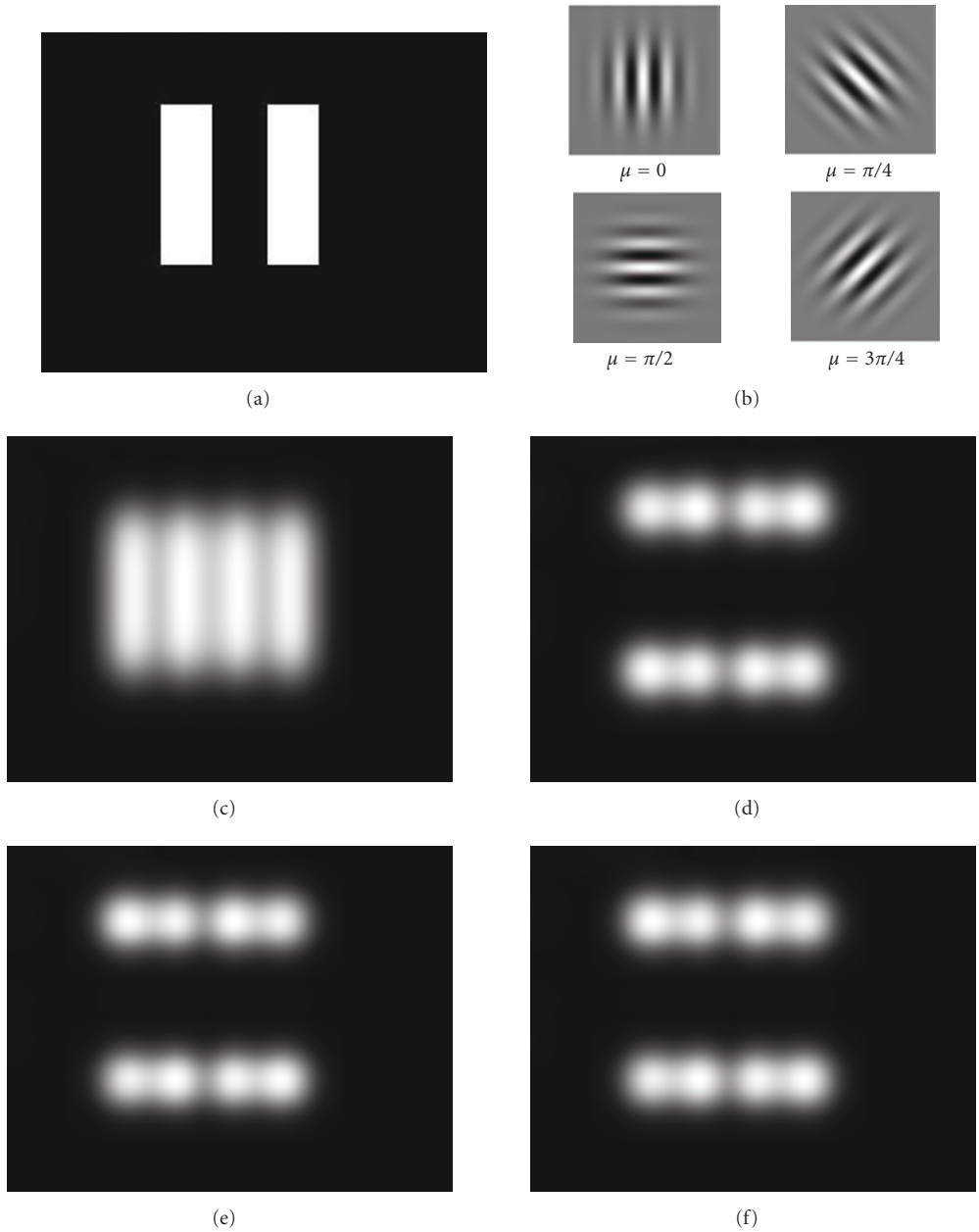


Figure 9: Gabor kernels and Gabor filter responses. (a) Input image; (b) 4 Gabor kernels with $\nu = 3$ and $\sigma = 2\pi$; (c), (d), (e), and (f) Gabor filter response with a Gabor kernel at orientation $\mu = 0$, $\pi/4$, $\pi/2$, and $3\pi/4$, respectively.

compensates for the DC value, thus making the kernel DC-free, that is, the integral $\int \psi_{\mu,\nu}(z)d^2z$ vanishes. Therefore, it is not necessary to consider the DC effect, when the parameter σ is large enough.

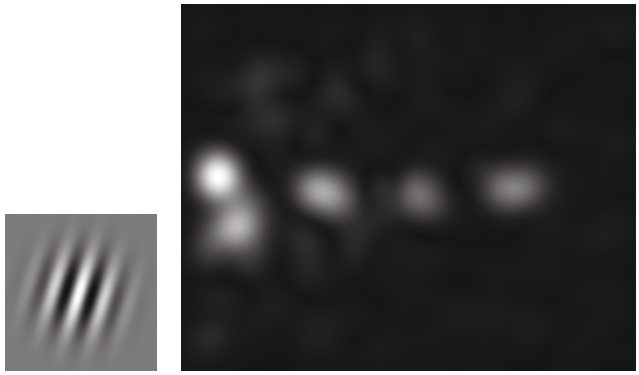
The Gabor filtering of an image I is the convolution of the image I with a Gabor kernel as defined by (18). The convolution image is defined as

$$O_{\mu,\nu}(z) = I(z) * \psi_{\mu,\nu}(z). \tag{20}$$

The response $O_{\mu,\nu}(z)$ to the Gabor kernel $\psi_{\mu,\nu}(z)$ is a complex function with a real part $\mathrm{Re}\{O_{\mu,\nu}(z)\}$ and an imaginary part $\mathrm{Im}\{O_{\mu,\nu}(z)\}$. The magnitude response $\|O_{\mu,\nu}(z)\|$ is expressed as

$$\left\|O_{\mu,\nu}(z)\right\| = \sqrt{\mathrm{Re}\left\{O_{\mu,\nu}(z)\right\}^2 + \mathrm{Im}\left\{O_{\mu,\nu}(z)\right\}^2}. \tag{21}$$

Figure 9(a) shows a synthesized binary image. Figure 9(b) shows four Gabor kernels with $\nu = 3$ and $\sigma = 2\pi$, at orientation $\mu = 0$, $\pi/4$, $\pi/2$, and $3\pi/4$, respectively. The Gabor filter responses are shown in (c), (d), (e), and (f), corresponding to the Gabor kernel at orientation 0, $\pi/4$, $\pi/2$, and $3\pi/4$, accordingly. Here, the interesting result is shown in (c), where the disconnected blobs in (a) are merged into one blob after Gabor filtering. The similar phenomenon



(a) (b)

Figure 10: (a) Dynamic Gabor kernel determined by the optical flow in Figure 6(c); (b) Gabor filter response for the frame difference image in Figure 6(d).

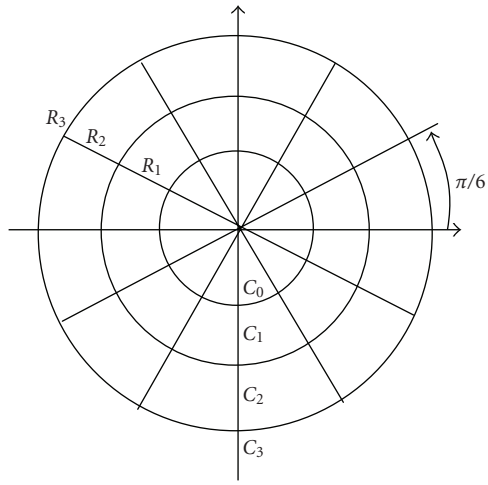


Figure 11: Specular highlight detector.

happens in target detection by frame differencing technique. If the interval between two consecutive frames is too large or if the targets move too fast, the moving targets appear as separate blobs in frame difference image. By carefully choosing the orientation of Gabor filter, separated blobs can be detected as a connected blob from Gabor response. Our algorithm employs this experiment result.

In our algorithm, we fix the following parameters, $k_{\max} = \pi/2$, $\sigma = 2\pi$, $f = \sqrt{2}$, and $\nu = 3$. The orientation μ is dynamically changed according to optical flows from inliers. We call it *dynamic Gabor filter*. The orientation μ is defined as

$$\mu = \frac{1}{K_{\text{in}}} \sum_{i=1}^{K_{\text{in}}} \theta\left(\vec{F}_i^{t't}\right), \tag{22}$$

where $\theta(\vec{F}_i^{t't})$ is the orientation of the optical flow $\vec{F}_i^{t't} \in F_{\text{in}}^{t't}$, and is given by

$$\theta\left(\vec{F}_i^{t't}\right) = \arctan \frac{y_i^{t'} - y_i^t}{x_i^{t'} - x_i^t}. \tag{23}$$

Figure 10(a) shows the dynamic Gabor kernel determined by the optical flows in $F_{\text{in}}^{t't}$ as shown in Figure 6(c). Figure 10(b) shows the Gabor filter response by performing convolution for the frame difference image in Figure 6(d) and the dynamic Gabor kernel in Figure 10(a).

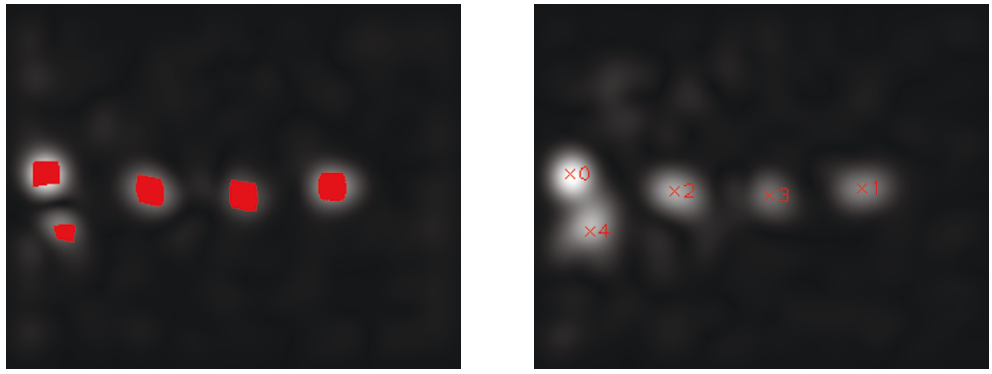
3.3. Specular Highlights Detection. As can be seen in Figure 10(b), the image changes appear as high intensity in the dynamic Gabor filter response. They look like spotlights. The center of the spotlight is brightest, and the brightness on the circular points around the center becomes dim gradually when the circle becomes larger. We call these high intensity *specular highlights*. Therefore, the target detection problem becomes the specular highlight detection problem. Because the intensity of highlights changes for the moving targets (some specular highlights are dimmer than others), the thresholding algorithms cannot detect all specular highlights successfully. Here, we employ the specular highlight detector as shown in Figure 11. The C_0 is the pixel under examination. This detector compares the intensity at C_0 and the intensity of pixels on the circular circles C_1, C_2, and C_3, with radius R_1, R_2, and R_3, respectively. C_1, C_2, and C_3 are sampled at $\pi/6$ interval, hence the detector will only compare the intensity at C_0 and 12 sample points, $C_{j,1}$, $C_{j,2}, \ldots, C_{j,12}$, from each circular circle. Let $G(z)$ denote the dynamic Gabor filter response at z, the discrimination of specular highlights is as follows

$$C_0 \text{ is} \begin{cases} \text{a specular highlight,} \\ \quad \text{iff } G(C_0) \geq G(C_{1,i}) \text{ and } G\left(C_{j,i}\right) \geq G\left(C_{j+1,i}\right), \\ \text{not a specular highlight,} \\ \quad \text{otherwise,} \end{cases} \tag{24}$$

where $j = 1, 2$, and $i = 1, 2, \ldots, 12$.

The specular highlight points detected from the dynamic Gabor filter response in Figure 10(b) are shown in Figure 12(a) by red dots. Note that red dots form several red regions in Figure 12(a). This is caused by the loose condition, " if and only if $G(C_0) \geq G(C_{1,i})$ and $G(C_{j,i}) \geq G(C_{j+1,i})$", in (24). The loose condition is chosen in attempt not to miss the possible specular highlights. These specular highlight points are denoted by $P^h = \{p_1^h, \ldots, p_{K_h}^h\}$, where K_h is the number of specular highlight points. In our algorithm, it is convenient to use the center and radius to represent the location and size of the specular highlights. To obtain the location and the size of specular highlights, $\{p_1^h, \ldots, p_{K_h}^h\}$ are clustered. Let $H_i(c, r)$ denote ith specular highlight, where r is the radius, c the center, and c contains x-coordinate, x_c, and y-coordinate, y_c.

The specular highlights generated above need to be clustered to determine the precise center of the specular spot. Among the clustering algorithms, k-NN (k nearest neighbor) algorithm needs a user predetermined constant k the number of the clusters [30]. It is not applicable to



(a) (b)

Figure 12: (a) Specular highlight points; (b) Specular highlight clustering.

our problem. On the other hand, the mean shift algorithm is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters [31]. However, the computation is complicated. Similarly, (support vector machine) SVM is another powerful clustering algorithm [32], but it is computationally heavy. In this work, $H_i(c,r)$ is obtained according to the following algorithm.

Specular Highlight Point Clustering Algorithm. The summary of this algorithm is as follows. For a specular highlight point, create a new cluster and consider this point is the center of the newly created cluster. Then, check whether there are other specular highlight points that are close to the current one, according to the predetermined threshold T_h. If yes, those points are also added to the newly created cluster, and the center of the cluster is updated after adding a specular highlight point to the newly created cluster. This process is repeated for all specular highlight points. After this processing, it forms a cluster. Then it chooses the next specular highlight point that is not clustered so far, and repeats the above processing. This processing is repeated until all specular highlight points are clustered. The details are given below.

(1) For $p_j^h \in P^h$, it is considered as the center of $H_i(c,r)$ and it is removed from P^h, added to $H_i(c,r)$, and set $c = p_j^h$ and $M_i = 1$, where M_i is the number of the specular highlight points in $H_i(c,r)$, and both i and j begin from 0, and $H_i(c,r)$ is an empty set initially.

(2) For $p_k^h \in P^h (k \neq j)$, if $\|p_k^h - c\| \leq T_h$, p_k^h is removed from P^h, added to $H_i(c,r)$, and update M_i and the center c according to

$$M_i = M_i + 1, \qquad x_c = \frac{1}{M_i}\sum_{m=1}^{M_i} x_m, \qquad y_c = \frac{1}{M_i}\sum_{m=1}^{M_i} y_m, \tag{25}$$

where T_h is a predetermined threshold value, $(x_m, y_m) \in H_i(c,r)$, (x_c, y_c) is the coordinates of the center c, and $\|p_k^h - c\|$ means the Euclidean distance between the specular highlight point, p_k^h, and the center c.

(3) Repeat step (2) for all specular highlight points in P^h. When this step finishes, $H_i(c,r)$ is obtained, and the radius r is given by

$$r = \max\left\|p_k^h - c\right\|, \tag{26}$$

where $k = 1, 2, \ldots, M_i$,

(4) Update i, and repeat steps (1) to (3) for the left specular highlight points in P^h to search for the next cluster.

(5) Repeat steps (1) to (4) until P^h becomes an empty set.

Let $H_S = \{H_1(c,r), H_2(c,r), \ldots, H_{Ks}(c,r)\}$ represent the detected specular highlights, where K_s is the number of specular highlights. Figure 12(b) shows the clustering result for the specular highlight points in Figure 12(a), where each cluster means a specular highlight. The specular highlights are numbered from 0 to 4, and the centers are marked by a small "x".

3.4. Moving Target Localization

3.4.1. Outlier Clustering. Because outliers are caused by the moving targets, they can be used for moving target localization. Here we employ the observation result that if outliers belong to the same moving targets, they are located closely, in optical flow field. Therefore, the outliers are clustered first. The clustering algorithm for outliers is the same one as described in Section 3.3, but with different clustering threshold T_o. Let $C_{\text{out}} = \{C_1(c,r), C_2(c,r), \ldots, C_{K_o}(c,r)\}$ represent the outliers clusters, where K_o is the number of the clusters. The outlier clustering result for the outliers detected from input images in Figures 6(a) and 6(b) is shown in Figure 6(c) by the purple circles, and the center of each cluster is marked by small "+" in purple. If all outliers are separated correctly, we can say that each cluster corresponds to one or multiple targets. However, this assumption is not always correct. Some moving target may not generate outliers

because outlier separation algorithm may fail or because the displacement of moving target is too small. This case is indicated in Figure 6(c) by the dotted circle in red, where a moving target exists. In the following, we combine both outlier clustering result and specular highlight detection result for moving target localization.

3.4.2. Moving Target Localization Based on Outlier Clustering and Specular Highlights. The discrimination rule for moving target localization based on outlier clustering and specular highlight detection is as follows. For a specular highlight $H_i(c,r) \in H_S$, if its center lies in a outlier cluster $C_k(c,r) \in C_{\text{out}}$ $(i = 1,\ldots,K_s, k = 1,\ldots,K_o)$, it is considered as a target. If its center does not lie in any outlier cluster, the dynamic Gaussian detector is employed, which is described in Section 3.4.3. According to this rule, the specular highlight numbers 0, 1, 3, and 4 in Figure 12(b) are identified as moving targets, and are marked by red circles in Figure 13. The localized targets are represented by its center and radius which is set at T_o (the thresholding for outliers clustering).

3.4.3. Moving Target Localization Based on Dynamic Gaussian Detector. As shown in Figure 12(b), a specular highlight is similar to a two-dimensional (2-D) Gaussian distribution. The moving target localization method described in Section 3.4.2 may fail if the feature point detector, described in Section 3.1.1, does not detect the enough outliers belonging to a moving target. To make the moving target localization robust, we further employ 2-D Gaussian function as a target detector to conduct the secondary moving target localization. (Correspondingly, the method used in Section 3.4.2 is called primary moving target localization.) A general 2-D Gaussian function is given by

$$G(x,y) = Ae^{-[a(x-x_0)^2+b(x-x_0)(y-y_0)+c(y-y_0)^2]}, \tag{27}$$

where

$$\begin{aligned} a &= \left(\frac{\cos\theta}{\sigma_x}\right)^2 + \left(\frac{\sin\theta}{\sigma_y}\right)^2, \\ b &= -\frac{\sin 2\theta}{\sigma_x^2} + \frac{\sin 2\theta}{\sigma_y^2}, \\ c &= \left(\frac{\sin\theta}{\sigma_x}\right)^2 + \left(\frac{\cos\theta}{\sigma_y}\right)^2 \end{aligned} \tag{28}$$

and the coefficient A is the amplitude, (x_0, y_0) is the center, σ_x, σ_y are the x and y spreads of the Gaussian function, and θ is the orientation. Figure 14 shows 2D Gaussian function distribution at orientation $\theta = 0, \pi/6, \pi/3, \pi/2, 2\pi/3, 5\pi/6$, respectively.

In our algorithm, the detector compares the specular highlight with 2D Gaussian kernel generated according to (27) and (28), and calculates the similarity. The orientation θ of 2-D Gaussian function is determined by the orientation of the specular highlight. Here we call it *dynamic Gaussian detector*. This detector algorithm is as follows.

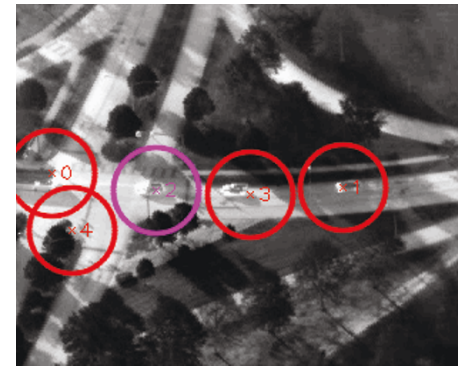


Figure 13: Target localization result.

Figure 14: Gaussian kernel at orientation $\theta = 0, \pi/6, \pi/3, \pi/2, 2\pi/3$, and $5\pi/6$, respectively.

Target Localization Algorithm based on Dynamic Gaussian Detector. (1) For $H_i(c,r) \in H_S$ which does not lie in any outlier cluster in C_{out}, extract $W \times W$ image I_{sub} centered at c for this specular highlight, where W is determined by r, and currently is set at $2 \times 1.2 \times r + 1$.

(2) I_{sub} is binarized by fixed threshold, $0.7v_{\max}$, where $v_{\max}$ is the maximal intensity in I_{sub}.

(3) The first principal axis of the binarized image I_{sub} is calculated according to

$$\alpha = \frac{1}{2}\arctan\frac{2m_{11}}{m_{20}-m_{02}}, \tag{29}$$

where

$$m_{pq} = \sum_{x=1}^{W}\sum_{y=1}^{W} I_{\text{sub}}(x,y)(x-x_c)^p(y-y_c)^q \quad (p,q = 1,1;\ 2,0;\ 0,2) \tag{30}$$

is the moment around the centroid (x_c, y_c). x_c and y_c are given by

$$x_c = \frac{m_{10}}{m_{00}}, \qquad y_c = \frac{m_{01}}{m_{00}}, \tag{31}$$

where

$$m_{p'q'} = \sum_{x=1}^{W}\sum_{y=1}^{W} I_{\text{sub}}(x,y)x^{p'}y^{q'} \quad (p',q' = 0,0;\ 1,0;\ 0,12). \tag{32}$$

Note that m_{pq} and $m_{p'q'}$ are the *moment* of order $(p+q)$ for the image I_{sub}, around the center c and origin, respectively. Equations (30) and (31) are the digital expression of the moment. Generally, for a 2D continuous function $f(x,y)$ the moment (sometimes called "raw moment") of

Table 2: Correct detection rate, miss detection rate, and hit rate for the 4 datasets.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Total number of targets	381	266	287	297
Detected targets	326	221	249	270
Missed targets	55	45	38	27
Correct detection rate	85.6%	83.1%	86.8%	90.90%
Miss detection rate	14.4%	16.9%	13.2%	9.10%
Hit rate	85.9%	81.3%	70.7%	76.60%

(a) (b) (c)

Figure 15: (a) A specular highlight; (b) Principal axis for the specular highlight in (a); (c) Generated Gaussian pattern.

order $(p + q)$ is defined by $m_{pq} = \iint_{-\infty}^{\infty} x^p y^q f(x, y)dxdy$, where $p, q = 0, 1, 2, \ldots$.

(4) α is used as the orientation to generate Gaussian kernel I_G, according to (27), where the Gaussian pattern size is W.

(5) The similarity between I_{sub} and I_G is calculated according to (12) [33], which is rewritten as

$$s = \sum_{k=0}^{W-1} \sum_{k=0}^{W-1} \frac{\left[I_{\text{sub}}(k,l) - \bar{I}_{\text{sub}}\right] \times \left[I_G(k,l) - \bar{I}_G\right]}{W^2 \sqrt{\sigma(I_{\text{sub}}) \times \sigma(I_G)}}. \quad (33)$$

If $s \geq T_G$, $H_i(c, r)$ is considered as a target, where T_G is the predetermined threshold.

(6) Repeat steps (1) to (5) for all specular highlights in H_S, which do not lie in any cluster in C_{out}.

Figure 15(a) shows the image I_{sub} for the specular highlight number 2 in Figure 12(b), which does not lie in any outlier cluster in Figure 6(c). Figure 15(b) shows the binarized specular highlight and the first principal axis by a long black line segment, and the second principal axis by short, and (c) shows the generated Gaussian kernel according to (27).

4. Experiment Results

The entire algorithm described in Section 3 is implemented by using C++ and OpenCV on windows platform. The input image size is 320×256, Δ is set at 2, the outlier clustering threshold T_o at H/6 (H is the image height), the specular highlight point clustering threshold T_h at $2T_o/3$, the similarity threshold T_G at 0.93, and A, σ_x, and σ_y are set at 1, 25.0, and 15.0, respectively. The IR video data from the VIVID datasets provided by the Air Force Research Laboratory is used. Figures 16, 17, and 18 show some experiment results. Figures 16(a) and 16(b) show two consecutive input images, (c) shows the detected optical flows (marked by red line segments) and outlier clustering (marked by purple circles), (d) the generated frame difference, (e) the detected specular highlights, and (f) the detected moving targets marked by red circles.

Figure 17 shows the target detection results at frame 29, 32, 37, 69, 78, and 82, for an input image sequence. Green circles mark the ground truth target positions, labeled manually, red circles means targets detected based on outlier clustering and specular highlights, and purple circles marks the output of the dynamic Gaussian detector. In frame 32, the target number 3 in (a) is missed. In frame 37, the target number 3 in (a) is also missed, and the dynamic Gaussian detector mistakenly detected a specular highlight (marked by purple circle) caused by tree leaves. In frame 69, the system also mistakenly detected a specular highlight caused by tree leaves. However, the system detected a moving target (number 2 in (d)) that was not marked by the human operator. In Frame 78, the system also detected a moving target (number 2 in (e)) which is the ground truth target but is not marked by the human operator. This is a human operator's mistake. In frame 81, the system mistakenly detected a target (number 0 in (f)) and lost one target.

Figure 18 shows target detection results at frame 44, 50, 53, 73, 81, and 84 for another input image sequence. Green circles mark the ground truth target positions, labeled manually, red circles means targets detected based on outlier clustering and specular highlights, and purple circles marks the output of the dynamic Gaussian detector. In frame 44, the dynamic Gaussian detector identified two targets, number 2 and 3, in (a). However, the target number 3 is a false target. In frame 53, the target in the middle was detected as two separated targets. In frame 81 and 84, the system lost one target.

5. Performance Analysis

To evaluate the performance of this algorithm, we selected four image sequences with the significant background as the test data. Each sequence contains 100 frames, and each frame contains two to four moving targets. The ground truth targets are labelled manually. The total number of targets in these 4 datasets is 1231. We examined the correct detection rate, hit rate, and processing time. The hit rate is defined as the ratio for the intersected area of detected target and ground truth target and the area of the ground truth target. The experiments are conducted on a Windows Vista machine mounted with a 2.33 GHz Intel Core 2 CPU and 2 GB main memory. The total average correct detection rate is 86.6%,

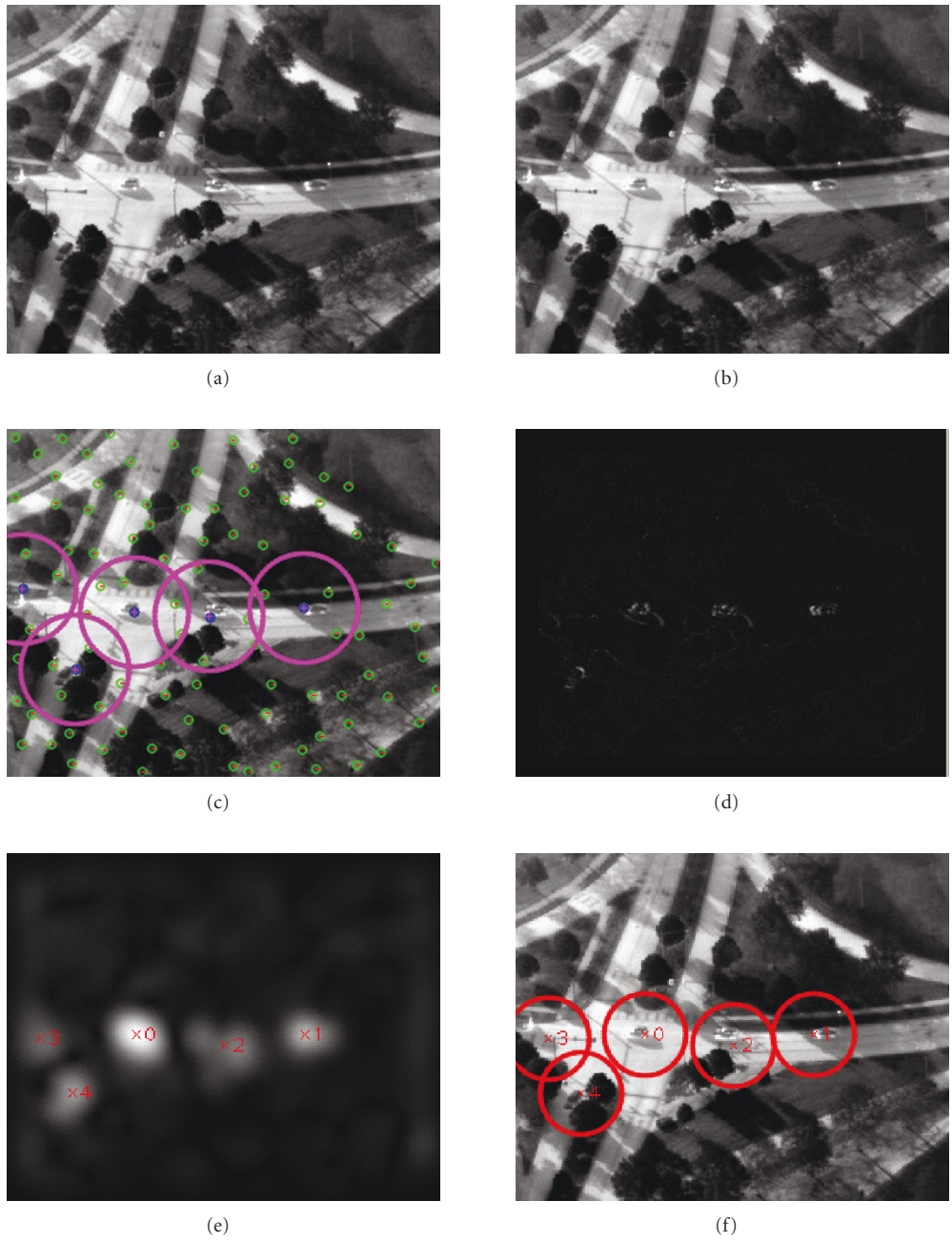
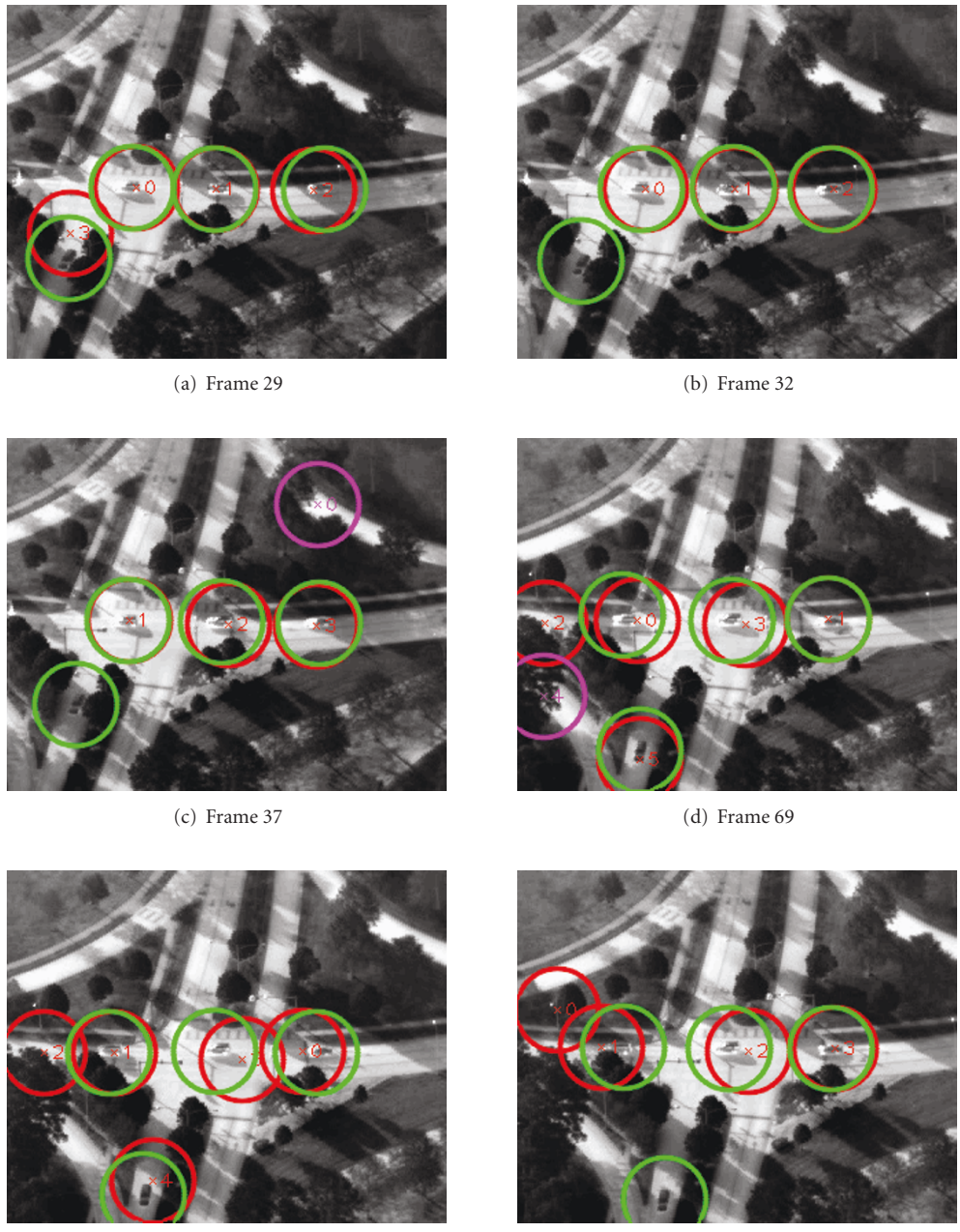


Figure 16: (a) and (b) Two input images; (c) Detected optical flows (marked by red line segments) and outliers clustering (marked by purple circles); (d) Frame difference; (e) Detected specular highlights; (f) Detected moving targets marked by red circles.

and hit rate is 78.6%, respectively. The detail detection results are shown in Table 2. The average processing time is 581 ms/frame. The detailed processing time are shown in Figure 19.

6. Conclusions and Future Works

This paper described a method for multiple moving target detection from airborne IR imagery. It consists of motion compensation, dynamic Gabor filtering, specular highlights detection, and target localization. In motion compensation, the optical flows for two consecutive images are detected from the feature points. The feature points are separated into inliers and outliers, accordingly, the optical flows are also separated into two classes, optical flows belonging to inliers and optical flows belonging to outliers. The optical flows belonging to inliers are used to calculate the global motion model parameters. Here, the Affine model is employed. After the motion model estimation, the frame difference image is generated. Because of difficulties to detect the targets from the frame difference image, we introduce the dynamic Gabor filter. In this step, we use the orientation of the optical



(a) Frame 29 (b) Frame 32

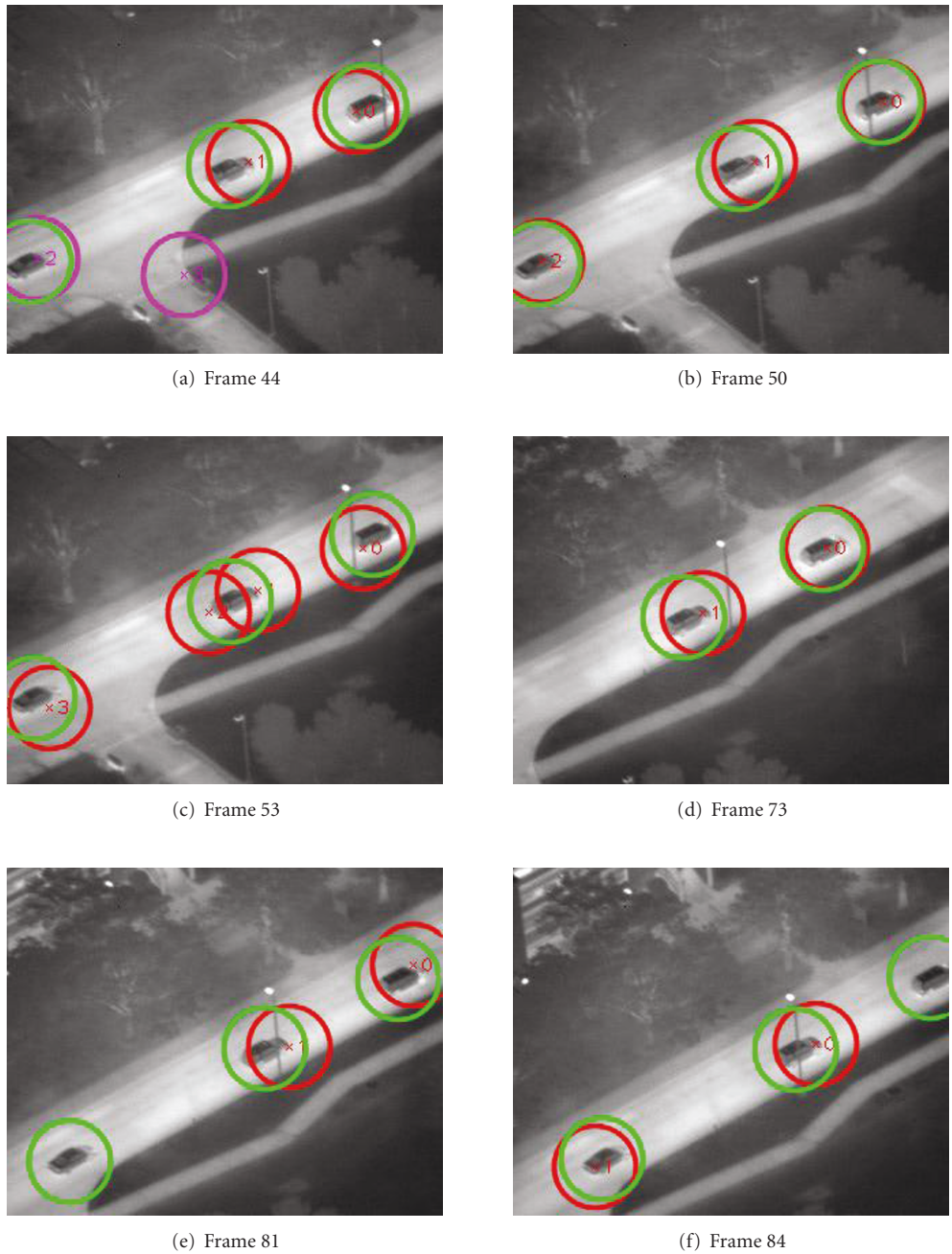
(c) Frame 37 (d) Frame 69

(e) Frame 78 (f) Frame 82

Figure 17: Target detection results in frame 29, 32, 37, 69, 78, and 82. Green circles mark the ground truth target positions, labeled manually. Red circles means targets detected based on outliers clustering and specular highlights. Purple circles mark the output of the dynamic Gaussian detector.

flows belonging to inliers to control the orientation of the Gabor filter. We call it *dynamic Gabor filter*. This is the first contribution of this paper. After the dynamic Gabor filtering, the image changes appear as high intensity in dynamic Gabor filter response. We call these high intensity *specular highlights*. In specular highlight detection, we use a simple but efficient detector to extract the specular highlight points. These specular highlight points are clustered to indentify the specular highlight center and its size. In the last step, it employs the outlier clustering and specular highlights to localize the targets. If a specular highlight lies in an outlier cluster, it is considered as a target. If a specular highlight does not lie in any outlier cluster, it employs the Gaussian detector to identify the target. The orientation of the specular highlight is used to control the orientation of Gaussian kernel. We call this detector *dynamic Gaussian detector*. This is the second contribution of this paper.

This algorithm was implemented in C++ and OpenCV. We tested the algorithm by using the airborne IR videos from AFRL VIVID datasets. The correct detection rate is



(a) Frame 44 (b) Frame 50

(c) Frame 53 (d) Frame 73

(e) Frame 81 (f) Frame 84

Figure 18: Target detection results in frame 44, 50, 53, 73, 81, and 84. Green circles mark the ground truth target positions, labeled manually. Red circles means targets detected based on outliers clustering and specular highlights. Purple circles mark the output of the dynamic Gaussian detector.

86.6%, and the hit rate for the correct detection is 78.6%. The processing rate is 581 ms/frame, that is, approximate 2 frames per second. This speed meets the requirement for many real-time target detection and tracking systems. As seen in Figures 17 and 18, in some cases the system fail to detect the targets or it mistakenly detects the image changes caused by the background significant features such as tree leaves or building corners. This can be improved by two efforts. The first one is to improve the inliers/outliers separation algorithm so that it maximally recognizes the feature points belonging to the background as the inliers. The second effort is to improve the dynamic Gaussian detector. Currently, the threshold for the dynamic Gaussian detector is set at a high value. This rejects some specular highlights to be recognized as targets. However, if this threshold is set at a low value, it will bring about false detection. And σ_x and σ_y in dynamic Gaussian detector are fixed. These can be dynamically changed according to the detection results of the dynamic Gabor filter. As shown in Section 3.1.1, six feature point detectors have been evaluated by employing

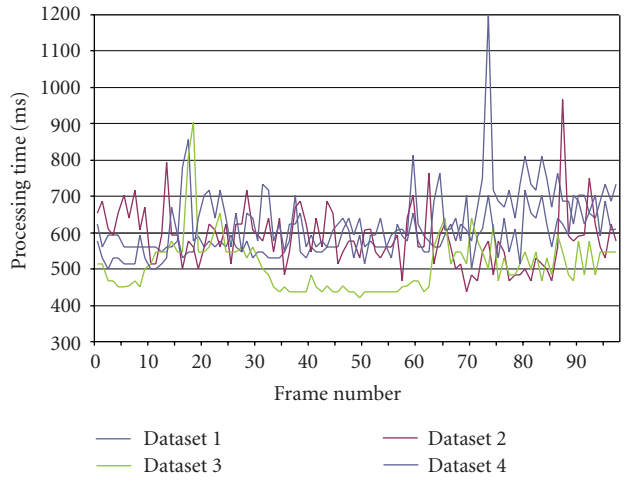


Figure 19: Processing time for multiple moving target detection.

the synthesized images and IR images. The Shi-Tomasi's method shows the best performance experimentally. The detailed performance analysis of these feature point detectors needs the theoretical investigation of these six detectors. The theoretical comparison of them will be detailed in our next paper. As shown in Section 3.1.3, this paper evaluated three transformation models between image frames. The experiment result shows the affine transformation model has best performance. This is because that, for the airborne-based IR image, the camera is far away from the object and the panning and tiling are not distinguished. The further theoretical study of these transformation models is our future work. Furthermore, since the target detection is a part of target tracking system, we will apply this algorithm to the target tracking system. This is also our future works.

Acknowledgments

This paper was partially supported by a Grant from AFRL under Minority Leaders Program, contract No. TENN 06-S567-07-C2. The authors would like to thank AFRL for providing the datasets used in this research. The authors also would like to thank anonymous reviewers for their careful review and valuable comments.

References

[1] A. Yilmaz, K. Shafique, and M. Shah, "Target tracking in airborne forward looking infrared imagery," *Image and Vision Computing*, vol. 21, no. 7, pp. 623–635, 2003.

[2] J. Y. Chen and I. S. Reed, "A detection algorithm for optical targets in clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 23, no. 1, pp. 46–59, 1987.

[3] M. S. Longmire and E. H. Takken, "Lms and matched digital filters for optical clutter suppression," *Applied Optics*, vol. 27, no. 6, pp. 1141–1159, 1988.

[4] H. Shekarforoush and R. Chellappa, "A multi-fractal formalism for stabilization, object detection and tracking in FLIR sequences," in *Proceedings of the International Conference on Image Processing (ICIP '00)*, pp. 78–81, September 2000.

[5] D. Davies, P. Palmer, and Mirmehdi, "Detection and tracking of very small low contrast objects," in *Proceedings of the 9th British Machine Vision Conference*, pp. 599–608, September 1998.

[6] A. Strehl and J. K. Aggarwal, "Detecting moving objects in airborne forward looking infra-red sequences," *Machine Vision Applications Journal*, vol. 11, pp. 267–276, 2000.

[7] U. Braga-Neto, M. Choudhary, and J. Goutsias, "Automatic target detection and tracking in forward-looking infrared image sequences using morphological connected operators," *Journal of Electronic Imaging*, vol. 13, no. 4, pp. 802–813, 2004.

[8] A. Morin, "Adaptive spatial filtering techniques for the detection of targets in infrared imaging seekers," in *Acquisition, Tracking, and Pointing XIV*, vol. 4025 of *Proceedings of SPIE*, pp. 182–193, 2000.

[9] A. P. Tzannes and D. H. Brooks, "Detection of point targets in image sequences by hypothesis testing: a temporal test first approach," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '99)*, pp. 3377–3380, March 1999.

[10] Z. Yin and R. Collins, "Moving object localization in thermal imagery by forward-backward MHI," in *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (OTCBVS '06)*, New York, NY, USA, June 2006.

[11] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Alvey Vision Conference*, pp. 147–151, 1988.

[12] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the 9th IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 593–600, Springer, June 1994.

[13] S. M. Smith and J. M. Brady, "SUSAN—a new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, no. 1, pp. 45–78, 1997.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[16] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of the 9th European Conference on Computer Vision (ECCV '06)*, vol. 3951 of *Lecture Notes in Computer Science*, pp. 430–443, 2006.

[17] F. Mohanna and F. Mokhtarian, "Performance evaluation of corner detection algorithms under similarity and affine transforms," in *Proceedings of the British Machine Vision Conference*, pp. 353–362, 2001.

[18] B. K. P. Horn, *Robot Vision*, MIT Press, Cambridge, Mass, USA, 1986.

[19] M. J. Black and P. Anandan, "The robust estimation of multiple motions: parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[20] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods," *International Journal of Computer Vision*, vol. 61, no. 3, pp. 211–231, 2005.

[21] C. L. Zitnick, N. Jojic, and S. B. Kang, "Consistent segmentation for optical flow estimation," in *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV '05)*, vol. 2, pp. 1308–1315, October 2005.

[22] J. Y. Bouguet, *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the Algorithm*, Intel Corporation, 2003.

[23] S. Baker, S. Roth, D. Scharstein, M. J. Black, J. P. Lewis, and R. Szeliski, "A database and evaluation methodology for optical flow," in *Proceedings of the IEEE 11th International Conference on Computer Vision (ICCV '07)*, October 2007.

[24] C. Moler, "Least squares," in *Numerical Computing with MATLAB*, chapter 5, pp. 141–159, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa, USA, 2008.

[25] S. Araki, T. Matsuoka, N. Yokoya, and H. Takemura, "Real-time tracking of multiple moving object contours in a moving camera image sequence," *IEICE Transactions on Information and Systems*, vol. 83, no. 7, pp. 1583–1591, 2000.

[26] D. H. Rao and P. P. Panduranga, "A survey on image enhancement techniques: classical spatial filter, neural network, cellular neural network and fuzzy filter," in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT '06)*, pp. 2821–2826, December 2006.

[27] D. Gabor, "Theory of communication," *Journal of IEE*, vol. 93, no. 26, pp. 429–457, 1946.

[28] J. G. Daugman, "Two-dimensional spectral analysis of cortical receptive field profiles," *Vision Research*, vol. 20, no. 10, pp. 847–856, 1980.

[29] J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journal of the Optical Society of America A*, vol. 2, no. 7, pp. 1160–1169, 1985.

[30] R. O. Duda, P. E. Hart, and D. G. Stork, "Section 4.4: *kn*—nearest-neighbor estimation," in *Pattern Classification*, Wiley InterScience, Malden, Mass, USA, 2004.

[31] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[32] J. Li, X. Gao, and L. Jiao, "A novel clustering method based on SVM," in *Advances in Neural Networks*, Lecture Notes in Computer Science, pp. 57–62, Springer, Berlin, Germany, 2005.

[33] C. Balletti and F. Guerra, "Image matching for historical maps comparison," *e-Perimetron*, vol. 4, no. 3, pp. 180–186, 2009.