Moving Object Detection on a Runway Prior to Landing Using an Onboard Infrared Camera

Cheng-Hua Pai, Yu-Ping Lin and Gerard G. Medioni
Institute for Robotics and Intelligence Systems
University of Southern California
Los Angeles CA 90089-0273
{chenghup,yupingli,medioni}@usc.edu

Ray Rida Hamza
Displays, Interaction & Decision Systems
Aerospace Advanced Technology
3660 Technology Drive MN65-2600
Minneapolis, MN 55418
rida.hamza@honeywell.com

Abstract

Determining the status of a runway prior to landing is essential for any aircraft, whether manned or unmanned. In this paper, we present a method that can detect moving objects on the runway from an onboard infrared camera prior to the landing phase. Since the runway is a planar surface, we first locally stabilize the sequence to automatically selected reference frames using feature points in the neighborhood of the runway. Next, we normalize the stabilized sequence to compensate for the global intensity variation caused by the gain control of the infrared camera. We then create a background model to learn an appearance model of the runway. Finally, we identify moving objects by comparing the image sequence with the background model. We have tested our system with both synthetic and real world data and show that it can detect distant moving objects on the runway. We also provide a quantitative analysis of the performance with respect to variations in size, direction and speed of the target.

1. Introduction

A system for detecting moving objects on the runway prior to landing using an onboard camera is helpful not only for unmanned air vehicles (UAVs), but also for pilots to determine whether it is safe to land.

Such system should have the following properties:

1. Ability to detect distant moving objects, in order to give enough response time for UAVs or pilots.

2. Robustness to plane motion.

3. Robustness to illumination, as to provide around-the-clock functionality.

In this paper, we present a method designed to detect moving objects from an infrared runway sequence taken by an airplane prior to landing.

Previous methods for detecting motion on a moving camera include optical flow based approach [1, 7, 8] and background subtraction based approach [9]. Our system belongs to the latter group.

Optical flow approaches require the availability of camera motion parameters (position and velocity) to estimate object range [1, 7, 8]. In [1], the optical flow is first calculated for extracted features. A Kalman filter then uses the optical flow to calculate the range of those features. Finally, the range map is used to detect obstacles. In [8], the model flow field and residual flow field are first initialized with the camera motion parameters. Obstacles are then detected by comparing the expected residual flow with the observed residual flow field. Instead of calculating optical flow for the whole image as in [8], [7] only calculates optical flow for extracted features since full optical flow is unnecessary and unreliable.

In contrast to the optical flow approaches, the background subtraction approach [9] does not need camera motion parameters. Camera motion is compensated by estimating the transformation between two images from feature points. Moving objects are detected by finding the frame differences between the motion-compensated pairs.

Comparing to previous methods, the scale of our moving objects is considerably smaller. Therefore, the optical flow method may not be able to detect moving objects in the scale of our interest.

Our work is different from [9] in that, instead of stabilizing consecutive frames and comparing them to find the changing parts in the sequence, we stabilize multiple frames to a local reference frame and use a background model to detect changes caused by moving objects.

1-4244-1180-7/07/$25.00 ©2007 IEEE

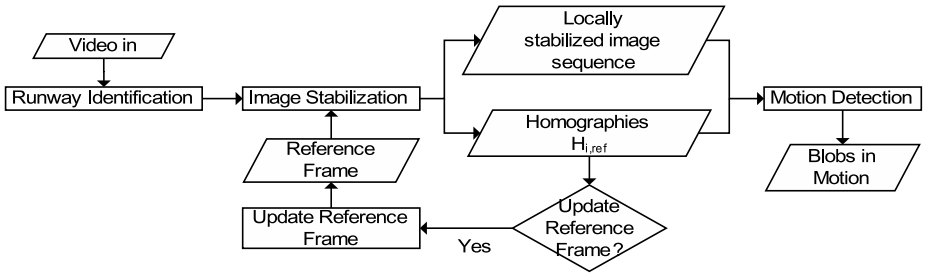


Figure 1. Flow chart of the system.

2. Overview

Background modeling is a method for detecting moving objects in sequences captured by static cameras [2]. However, this method is not appropriate if the camera moves or the background intensity changes. Therefore, we divide the problem into two steps: stabilization and motion detection. The first step compensates for the camera movement by stabilizing the runway. Once the stabilized runway is obtained, we use a background model to segment moving blobs on the runway.

Our approach is summarized in figure 1. First, a *Runway Identification* process identifies a 4-sided polygon which contains the runway location in the first image. Once this 4-sided polygon region is selected, an *Image Stabilization* process estimates the homographies between the selected region in each pair of consecutive images.

Images are then warped to automatically selected reference frames to form a locally stabilized image sequence. The reference frame is updated when necessary by the *Update Reference Frame* process. The homographies along with the locally stabilized image sequence are then given to the *Motion Detection* process where global intensity variation is compensated and moving objects on the image are identified.

3. Stabilization

Since it is very hard to directly detect moving objects when the camera also moves, our first goal is to stabilize the image sequence. We assume that the ground is a planar surface, which is a reasonable assumption in the neighborhood of a runway. With this assumption, the change of viewpoint between two adjacent frames can be represented by a homography.

3.1. Runway identification

Because the stabilization process requires a planar surface, and non-planar areas in the image sequence do not fit the transformation and may invalidate it, we need to restrict the region of interest. We first tried a vanishing line method to stabilize the image below the vanishing line. However, since the bottom of the image is closer to the airplane, the height of the buildings and trees make the process unstable. Knowing that, we select the planar region around the runway, and the stabilization process is applied only to this region. We hand-picked the polygons for our test sequences; however, with the help of onboard devices such as a GPS, the vertices' locations can be calculated automatically in an operational scenario.

3.2. Image stabilization

Let I_i be the i^{th} frame in the video sequence starting at 0, R_i be a planar region in I_i containing the runway and marked by its vertices, and $H_{i,i-1}$ be the homography between R_i and R_{i-1}. We have $R_{i-1} = H_{i,i-1}R_i$ and $R_i = H_{i-1,i}R_{i-1}$. A reference frame can be any frame in the image sequence. We use Ref to represent the index of the reference frame which the current image registers to. It is initialized to 0, and is automatically updated. Hence we have

$$R_{Ref} = \begin{cases} R_n, & \text{if } n = Ref \\ H_{m,Ref}R_m, & \text{if } m > Ref \end{cases} \quad (1)$$

The homography of the current region R_m with respect to its (local) reference frame R_{Ref} is derived as

$$H_{m,Ref} = \prod_{i=Ref+1}^{m} H_{i,i-1} \quad (2)$$

Feature points are used to find a robust estimation of the perspective transformation $H_{i,i-1}$ between two consecutive frames.

We first tried Harris corner features [5] for this task. The number of Harris points can be varied by a threshold; however, they are not stable enough for our test sequence to approximate the correct transformation between frames. We then used SIFT (Scale Invariant Feature Transform) features [6]. The number of SIFT points in a 720-by-480 test sequence is about 800 for each frame, and we obtained a much better result with SIFT feature points.

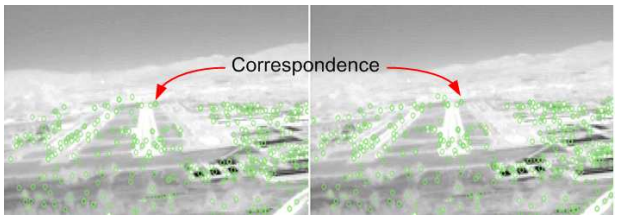


Figure 2. The corresponding SIFT features in consecutive images.

SIFT feature points within the runway in each pair of consecutive images are extracted and matched as shown in figure 2. Then a RANSAC [3] process is applied to estimate the best perspective transformation for the pair of images from the matched feature points. The RANSAC process chooses 8 random correspondences from the matched feature pairs (2 pairs from each quadrant), and calculates a perspective transformation from it. The RANSAC process is applied 2000 times and the result is evaluated by applying it to the matched pairs. The transformation that is correct for the largest number of matched pairs is chosen as the best approximation.

3.3. Reference frame update

In our early implementation, a single reference frame was chosen for the whole sequence. However, it is not a good idea for a long sequence since small errors are inevitable when doing registration and these errors may accumulate to affect the stabilization results of later frames. Moreover, the distance to the runway varies a lot in the sequence, and some detail will be lost if we warp every frame to a fixed reference frame. To resolve this problem, we allow updating of reference frame for the stabilization process.

On option is to swap reference frame on a fixed interval. The problem with this approach is that the interval is too short when the airplane is very far from the runway, since the scale of the runway does not change much during the interval; but the interval is too long when the airplane is close to the runway as the aspect changes quickly.

We need to define a measure that gives a longer interval when the runway is far, and a shorter interval when the runway is near.

We then examine the tilting angle of the warped image to determine when to update the reference frame. It turns out that we could not find this angle because the depth information is missing in the transformation matrix.

We noticed that after warping, the image corners changes its angle. We tried to threshold on those angles, but we found it unstable. As the plane turns by rotating toward one direction, the angles of the warped image corners also changes, which affects the decision.

Finally, we base our decision on the ratio of the lower edge length before and after the warping as in figure 3 to decide whether to swap a reference frame. The thresholding ratio we use is $(length\ after)/(length\ before) = 0.8$. Since this ratio is related to the tilting angle, thresholding on this measure has the same effect as thresholding on the tilting angle. This measure gives the most stable result of the three measures we've tried.

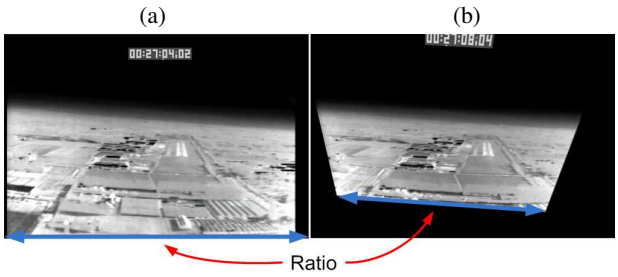


Figure 3. Ratio of lower edge length. Reference frame (a) and a stabilized image (b).

4. Motion detection

Now that we have an online stabilization process, the next problem is to find the moving objects in the sequence. The method we choose for this task is background modeling [2]. The approach for this module can be summarized by the flow chart in figure 4.

4.1. Runway mask

Since the area of interest is strictly the runway, other areas can be filtered out. We define a runway filter f to be a binary mask in the shape of the runway. The vertices of the runway in the mask is hand-picked for the test sequences. The process simply applies an "and" operation on the image and the binary mask to single out the area of interest.

When the reference frame changes from I_i to I_j, the following equation is applied to the runway mask vertices:

$$f_i = H_{j,i} f_j \tag{3}$$

4.2. Gain compensation

According to [4], the intensity between any two images with different gains can be modeled by an affine transformation.

$$\forall (x, y): \quad I_j(x, y) = m_{i,j} I_i(x, y) + b_{i,j} + \epsilon_{i,j} \tag{4}$$

By ignoring the saturated pixels, the transformation can be estimated by LMSE (Least Mean Square Estimation) and the gain can be compensated figure 5.

Because LMSE is used, small errors can be introduced in the compensation. If the whole sequence is compensated

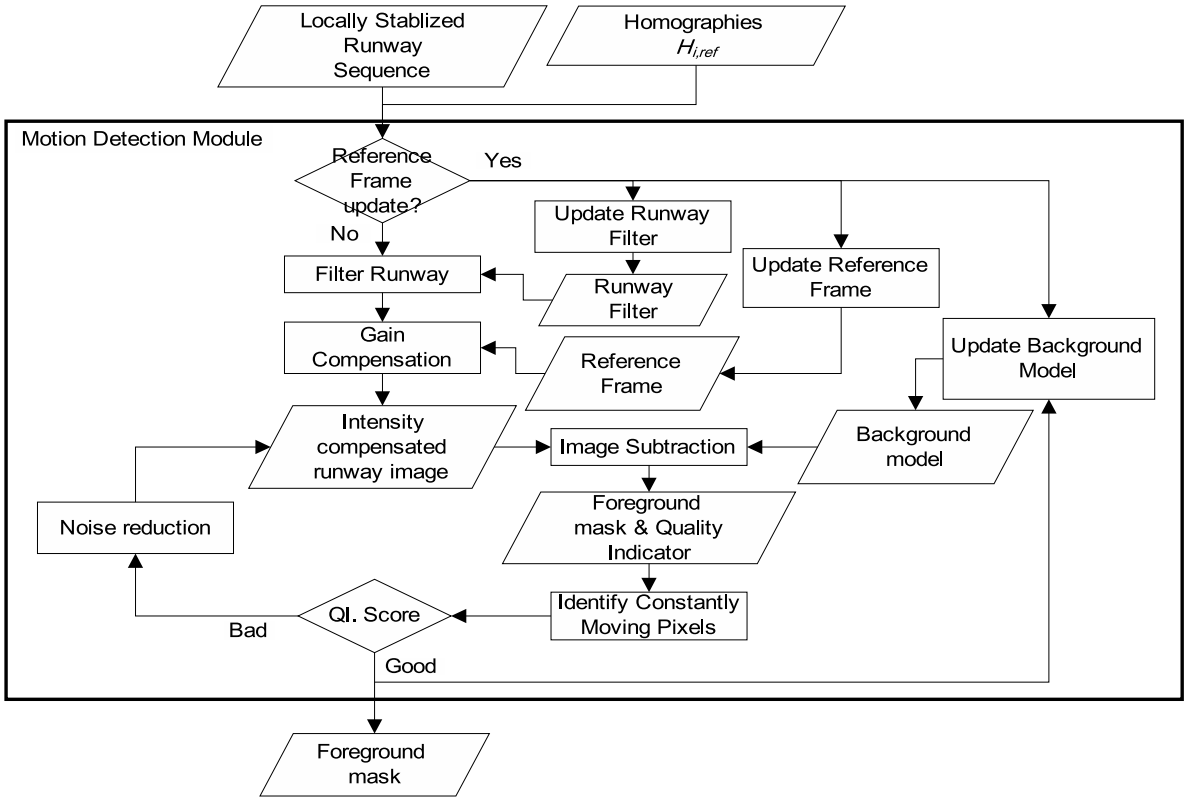


Figure 4. Flow chart of the motion detection module.

(a) (b)

Figure 5. A runway before gain compensation (a) and after gain compensation (b).

recursively, the errors accumulate to affect the global intensity of later frames. To increase the accuracy of the gain compensation, we correct the global intensity with respect to the reference frame of the current frame rather than between two adjacent frames. As a result, errors no longer accumulate and the intensity of the sequence is stable.

Since we have multiple reference frames, we have to adjust the intensity of **all** reference frames to a specific intensity before compensating other frames against them. We first tried to adjust the intensity of the reference frames against the mean intensity of the background model, but found this method not stable.

Even if the learning rate is increased, we still get very dark or very bright images at the end of some sequences. The reason is that, by normalizing the reference frame against the background model and the rest of the frames against the reference frame, we force every frame to behave like the initial background. As a result, the intensity range of the initial background greatly affects the compensation quality.

We also tried to linearly scale the initial background to increase its intensity range, but this produced similar results.

Knowing that using the mean intensity of the background model as the specific intensity does not work, we tried to use the intensity of the new reference frames as reference. We adjust the mean intensity of our background model against the new reference frame using equation 5.

$$\mu_{Ref}(x,y) = m_{Ref} I_{Ref}(x,y) + b_{Ref} + \epsilon_{Ref} \quad (5)$$

By doing so, we not only reduce the effect of the initial background, but also increase the quality of our background model.

4.3. Background modeling

A Gaussian distribution is used to model the intensity of each pixel [2]. The mean (μ) of the Gaussian model is initialized to the value of the first image.

The variance σ^2 is initialized to a constant value (currently 5). Both mean and variance are updated for each new frame according to the following formula where ρ is the learning rate (currently 0.02).

$$\mu_i(x,y) = (1-\rho)\mu_{i-1}(x,y) + \rho I_i(x,y) \quad (6)$$

$$\sigma_i^2(x,y) = (1-\rho)\sigma_{i-1}^2(x,y) + \rho(I_i(x,y) - \mu_i(x,y)) \quad (7)$$

Pixels having intensity difference greater than $4\sigma_i^2$ from μ_i are marked as foreground pixels.

When the reference frame updates, we need to adjust the orientation of our background model. Since the background model models the previous reference frame, by treating the mean and variance as images, we can warp them to the coordinates of the new reference frame. By applying the homographies between the reference frames to the mean and the variance images (equation 8, 9, 10), we obtain a background model for the new reference frame.

$$[u,v,1]' = H_{j,i}[x,y,1]' \quad (8)$$

$$\mu_i(u,v) = \mu_j(x,y) \quad (9)$$

$$\sigma_i^2(u,v) = \sigma_j^2(x,y) \quad (10)$$

Where x and y are coordinates in the old reference frame and u and v are the coordinates in the new reference frame.

However, each reference frame update also brings more details. These details may be classified as foreground because they were not present in our background model. To reduce the effect of details, we update the background model with the new reference frame by applying equation 6 and equation 7 right after the transformation.

4.4. Identify constant moving pixels

Since the scale of our moving objects is small, we cannot apply morphological operations to each foreground mask to reduce noise, as it may remove our targets, too. However, since real moving objects move more smoothly comparing to random noise, we can detect random noise by enforcing smoothness constraint in consecutive foreground masks.

After the binary foreground mask is produced, we compare it with the previous foreground mask to find constant moving pixels. In order to suppress the most number of noise without removing the object, we define constant moving pixels to be pixels that are marked as foreground in both the previous and the current foreground mask plus the foreground pixels that have moved one pixel in any direction.

To find those pixels, the previous foreground mask is first dilated, and then a binary "and" operation is applied to the dilated mask and the current mask.

4.5. Evaluating foreground mask quality

To evaluate the quality of a resulting foreground mask, we need a quality indicator. One such indicator is the total number of foreground pixels in the foreground mask. If the number is greater than a threshold (currently 250), the result is considered poor.

4.6. Noise reduction

If we know that a foreground mask is poor, we can go back and try to restabilize the image in order to improve the result. Since we used feature points during the stabilization step, we need to select a different source of information for the refinement. We considered using linear features, such as edges, which can be detected in the image, to improve the stabilization. By aligning edges of a poorly stabilized image with those of the base image, we hope to find a better transformation. The results, however, were inconclusive.

Instead, we found that we could apply the same stabilization technique on the gradient of the two poorly stabilized images to refine the transformation. Since the SIFT features [6] are extracted from gradient maps, they contain edge information. By using these features to estimate transformation, we take edge information into account.

One modification that dramatically improves the stabilization quality can be achieved by making sure that correspondences from each of the 4 quadrants are selected to build the homography model in each iteration of RANSAC. The previously mentioned restabilization process is no longer needed because after the modification, it can't improve stabilization anymore.

Even though random noise is removed when we check for constant moving pixels, there is still some noise caused by local intensity variation. For example, sunlight penetrating through clouds creates a bright area in our image.

After studying the intensity of the foreground pixels in noisy images, we found that there exist a linear relationship between the intensity of the foreground pixels in the runway image and the intensity of the same pixels in the mean of the background model, as shown on figure 6.

By applying a normalization process on the foreground pixels (equation 11), we were able to reduce the noise by 75% in some cases, as shown in figure 7. Notice that the scale in the vertical axis is different.

$$I_i^{fg}(x,y) = m_i \mu_i^{fg}(x,y) + b_i + \epsilon_i \quad (11)$$

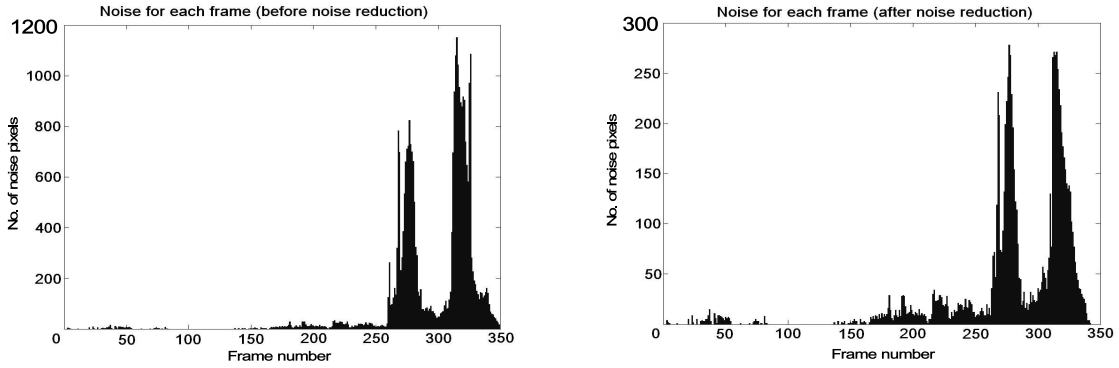
Where I_i^{fg} denotes the intensity of foreground pixels at index i.

5. Experimental results

We used both synthesized and real world runway sequences to test our program. For the synthetic experiments,

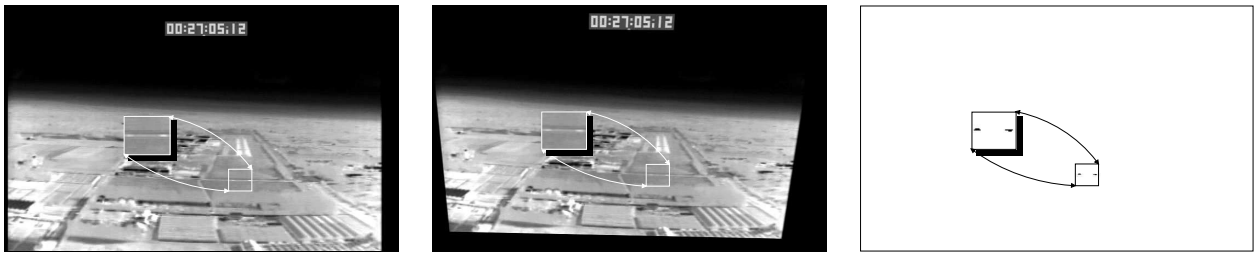
(a) (b) (c)

Figure 6. Foreground pixels' intensity in runway image vs. foreground pixels' intensity in background model (a). Foreground mask before noise reduction (b) and after noise reduction (c).



(a) (b)

Figure 7. Noise for each frame before noise reduction (a) and after noise reduction (b).



(a) (b) (c)

Figure 8. A real world runway image before stabilization (a), after stabilization (b) and the extracted moving objects (c).

we produced 150 runway sequences from a test sequence. The synthetic sequences have 351 frames and for each sequence, a different simulated object is added. The added objects are of different sizes, moving in different directions at different speeds. For the real world test, we ran the program on 18 real world runway sequences with moving objects.

The performance of our system is about 5 seconds per frame on a 2.8 Ghz Pentium 4 processor. Due to the calculation of SIFT features, the stabilization process is the most computationally complex part of our system. The motion detection process can generate foreground masks at about 10 frames per second.

5.1. Synthetic data

The three variables for the simulated objects are size, direction and speed. For the size variable, we vary from 2-by-2 to 6-by-6 (pixel). For the direction variable, diagonal, vertical, and horizontal directions are considered. For the speed variable, the range is from 0.1 pixels per frame to 2.8 pixels per frame with 0.3 pixels per frame increment. By mixing and matching the three variables, 150 test sequences were generated.

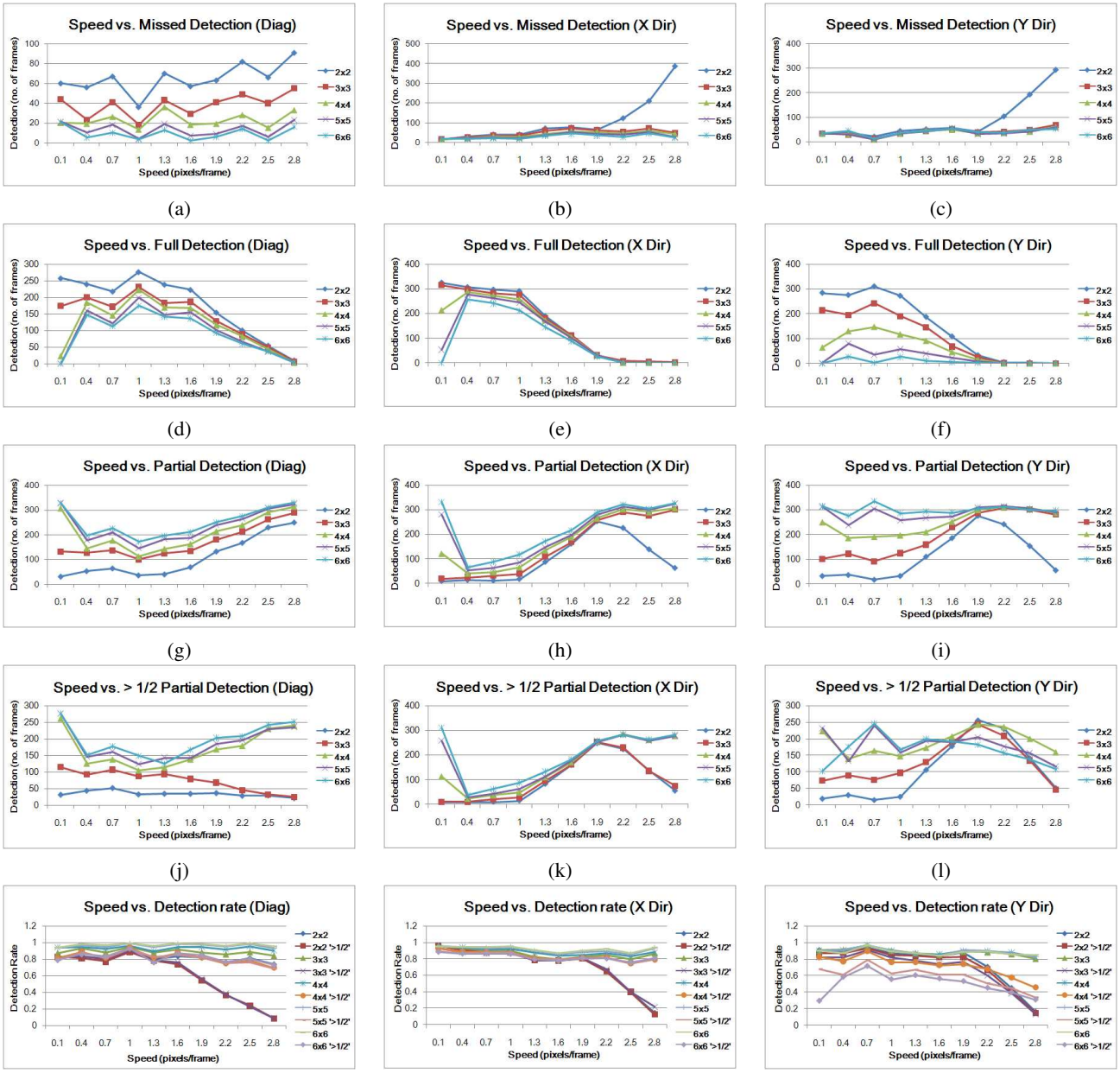


Figure 9. Result for synthetic data. Speed vs. number of missed detection in three directions (a), (b) and (c). Speed vs. number of full detection in three directions (d), (e), and (f). Speed vs. number of partial detection (g), (h) and (i). Speed vs. number of detecting more than half of the object (j), (k) and (l). The detection rate for objects moving in three directions (m), (n) and (o).

Since we know the object positions in the simulated sequences, we were able to collect statistical data for analysis. We count the number of frames in a sequence that missed the object completely, detected the object fully, detected the object partially, and detected partially but more than half of the object. The collected data is summarized in figure 9.

As for the detection rate, it is greater than 73% in diagonal direction and greater than 77% in horizontal and vertical directions except objects of size 2-by-2 moving at a speed higher than 1.9 pixels per frame.

For the detection rate of detecting more than half of the object, it is greater than 69% in diagonal direction except objects of size 2-by-2 and 3-by-3 moving faster than 1.6 pixels per frame. In horizontal direction, it is greater than 73% except object of size 2-by-2 and 3-by-3 moving at more than 1.9 pixels per frame. In vertical direction it is

greater than 60%, except for object of size 6-by-6 and object of all sizes with speed greater than 1.9 pixels per frame.

5.2. Real data

We applied our system to 18 sequences of the same runway taken in 4 different days. Since no moving objects can be observed in the runway, we included the road just before the runway for the test (figure 8). The system successfully detects 35 of the 37 vehicles moving on the road. One of the car was not detected because the background model has adapted to a slow moving truck just before the car. This case however, rarely occurs on the runway. The other misdetected car was removed by the noise reduction process because it appeared near the end of the sequence where noise is high.

6. Discussion

We found that when the contrast in a sequence is low, fewer features can be extracted for the stabilization. As a result, the stabilized runway slowly slides. Fortunately, the slide is slow and our background model can adapt to the changes.

In the simulated test, we noticed a small increase of noise as the size of the object increases. One reason for this behavior is that since we use a dilation with a 3-by-3 kernel and an "and" operation on adjacent masks to filter the constant moving pixels, larger objects will allow more noise around the object to pass through this filter.

Since we only allow constant moving pixels to move one pixel distance in adjacent frames, moving objects with speed greater than 1 pixel per frame in horizontal and vertical directions and $\sqrt{2}$ pixels per frame in diagonal direction will not be fully detected. Also, a moving object with speed greater than half of its dilated size will lose more than half of its size in the foreground mask.

When we compare the direction variable, while diagonal and horizontal directions behave similarly, vertical direction has noticeably fewer detections of more than half of the object. One explanation is that in a runway sequence, many background objects are aligned vertically. Therefore, if an object moving vertically was between two vertically aligned objects (or edges), the detection rate is affected by them.

Even though the chosen threshold parameters work well for our synthetic tests, they may need fine-tuning in real world sequences for some cases such as slow moving vehicles and low contrast sequences.

Finally, the noise reduction method we use seems to work very well when the stabilization quality is good. However, near the end of a sequence, there are not enough features to stabilize the runway and the runway slides, some background edges may be classified as foreground pixels. In this case, our program applies the noise reduction method and remove those edges, which is an unwanted behavior.

7. Conclusion

We have presented a two step method to detect distant moving objects on the runway with an onboard infrared camera. The system is able to detect distant moving objects thus give enough response time for UAVs or pilots, is indifferent to plane motion and since it utilizes infrared, it is also unaffected by illumination. We have performed extensive validation on both synthetic and real image sequences. Our next steps are further validation on a larger data set, and speed improvement so that the system can run in real time.

Acknowledgment

This work was supported by a grant from Honeywell Corporation. The infrared runway sequences were provided by Randy Bailey of NASA Langley Research Center.

References

[1] R. S. B. Sridhar and B. Hussien. Passive range estimation for rotor-craft low-altitude flight. *Machine Vision and Applications*, 6(1):10–24, 1993.

[2] W. G. Chris Stauffer. Adaptive background mixture models for real-time tracking. *1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)*, 2:2246, 1999.

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

[4] R. C. H. Yalcin and M. Hebert. Background estimation under rapid gain change in thermal imagery. *Second IEEE Workshop on Object Tracking and Classification in and Beyond the Visible Spectrum (OTCBVS'05)*, 2005.

[5] C. Harris and M. Stephens. A combined corner and edge detector. *Proceedings of The Fourth Alvey Vision Conference*, (1):147–152, 1988.

[6] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[7] T. G. R. Kasturi, O. Camps and S. Devadiga. Detection of obstacles on runway using ego-motion compensation and tracking of signicant features. *Proceedings 3rd IEEE Workshop on Applications of Computer Vision, 1996 (WACV'96)*, pages 168–173, 1996.

[8] S. Sull and B. Sridhar. Runway obstacle detection by controlled spatiotemporal image flow disparity. *IEEE Transactions on Robotics and Automation*, 15(3):537–547, 1999.

[9] Q. Zheng and R. Chellappa. Motion detection in image sequences acquired from a moving platform. *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Minneapolis*, 5:201–204, 1993.