Moving Object Tracking using Single Camera on Aerial Robot

JiaLei Wang, jw865@cornell.edu

***Abstract*— Robust detection of moving objects from an aerial robot is required for safe outdoor navigation, but is not easily achieved because the motion is two fold: motion of the moving object and motion of the robot itself. Our goal is to have the robot identify the actual moving objects from the dynamic camera view space, track the moving objects and give the right command to avoid any potential impact with the moving objects.**

I. Introduction

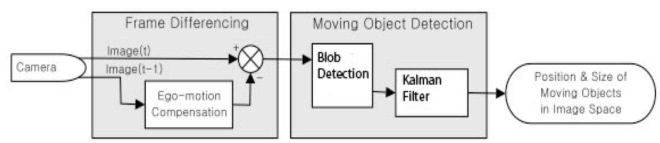
It is particularly important for aerial robot to detect and avoid obstacles because any minor collision with the obstacle could spell disaster in the air. While extensive research has conducted on detecting stationary obstacles, tracking and avoiding moving objects can be more challenging. There are two independent motions involved: the motion of the robot and the motion of moving objects in the environment. Unfortunately, these two motions are blended together when measured through a sensor such as a camera. In order to detect moving objects robustly, it should be able to decompose these two independent motions from sensor readings. Once motion has been identified, moving objects needed to be tracked, and appropriate action would be sent to robot in order to avoid any potential impact with the moving objects.

II. Related Work

There are various methods to stabilize camera when it is moving, such as by computing optical flow [1,2] and by tracking features [3,4]. The basic idea is to estimate the transformation between two image coordinate systems. Work focusing on robust multiple target tracking includes [5] uses Kalman filter to detect and track human activity with the combination of a static camera and a moving camera, and [6] which uses a particle filter to track multiple objects using a stationary camera, and [7] which also uses particle filter to track people indoors using a laser rangefinder.

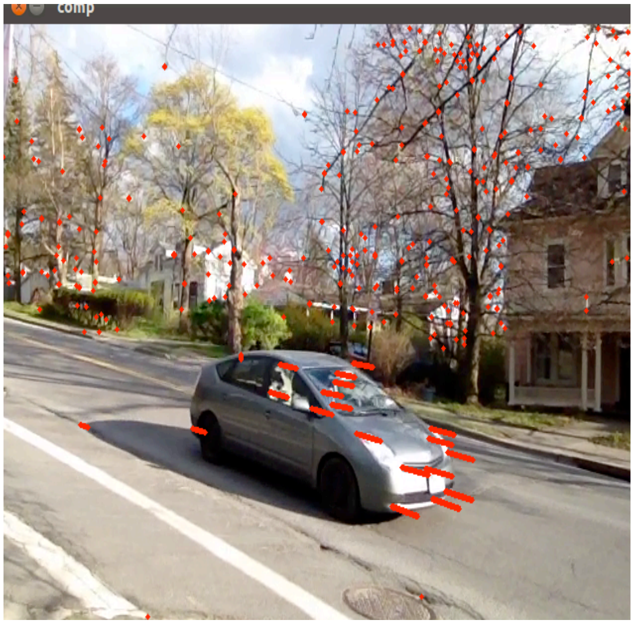
III. Our Approach

We use the single onboard camera (mounted on the aerial robot) to detect and track any moving objects. The motion detection process is in two steps: the stabilization of camera images, and the tracking of moving objects. We propose to use some ego-motion compensation [8] to stabilize the motion of camera, and then detect foreground object and segment from a video stream. After segmentation, we group the connected foreground points as blob, and implement Kalman filter to track the blobs, namely the moving objects.Figure below gives an execution flow of the program.



A. Ego-motion compensation

The ego-motion compensation is a transformation from the image coordinates of previous image to that of current image, so that these the effect of ego-motion of camera can be eliminated. We adopt good feature selection algorithm from the opencv library to select a number of "good" features from the given image. Here "good" feature means the features having complex textures, like corners of bricks and cars, and leaves. Then we apply Lucas-Kanade method to track those features in the subsequent image. The following image marks out the good feature to track with red points, and track the corresponding features in the subsequent image. We connect line between the feature pairs to show the sense of optical flow.



When we find feature pairs <f_{t-1},f_t>, where (t-1) is the last image and t denotes the current image, the ego-motion of camera can be estimated by using a transformation model. The effective and simple transformation model is given below:

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0\, f_x^{t-1} + a_1\, f_y^{t-1} + a_2 + a_3\, f_x^{t-1}\, f_y^{t-1} \\ a_4\, f_x^{t-1} + a_5\, f_y^{t-1} + a_6 + a_7\, f_x^{t-1}\, f_y^{t-1} \end{bmatrix}$$

Then we simply apply linear regression to train the constants.

The next procedure is to eliminate the bad features and refine the transformation modal. The algorithm can be stated as below:

a. Compute the initial transformation matrix using the full features.

b. Take out the bad features as outlier using the criteria:

$$\begin{cases} f_i \in F_{in} & \text{if } |f_i^t - T_{0\,t-1}^{\;t}(f_i^{t-1})| < \epsilon \\ f_i \in F_{out} & \text{otherwise} \end{cases}$$

c. Re-compute the refined transformation matrix using only 'good' features.

d. Repeat until transformation matrix converges.

Now we can use the transformation model obtained in previous to manipulate the whole image pixels, in order to eliminate the effect of ego-motion of camera. For each pixel (x,y):

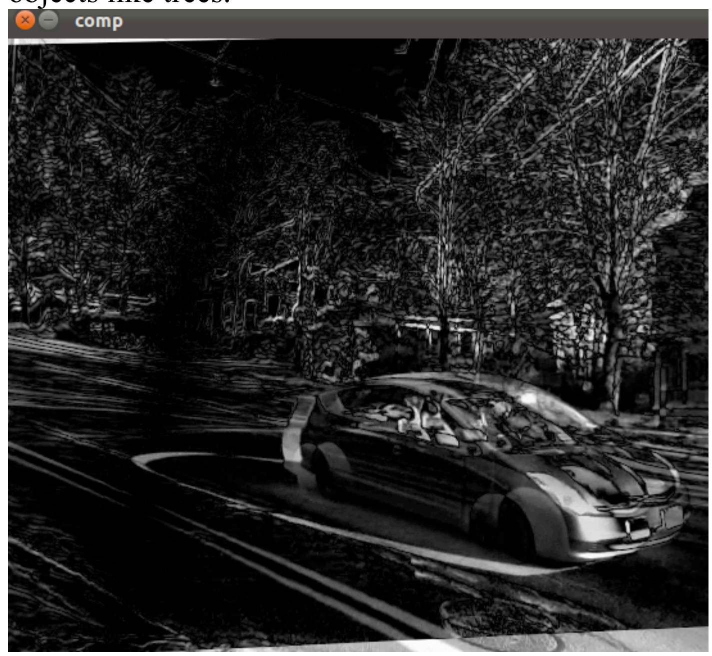
$$I_{comp}(x,y) = I_{t-1}\left({T_{t-1}^{t}}^{-1}(x,y)\right)$$

where Icomp is the transformed image.

The difference image between two consecutive image is computed as

$$I_{diff}(x,y) = |\,(I_{comp}(x,y) - I_t(x,y))\,|$$

The following picture shows one of the difference images. Which is not quite good since it suppose to suppress the stationary objects like trees.



B. *Foreground segmentation*

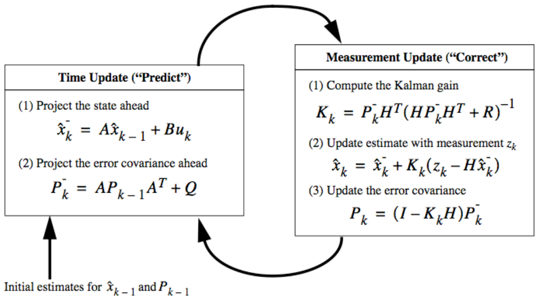
We can use the difference images obtained to foreground the actual moving points. We use mixture of Gaussians function and apply Expectation-Maximization algorithm to model each pixel, and the pixels in the region of moving objects can be foregrounded. Lastly we use blob library for opencv to detect connected foregrounded region, namely, the moving object.

C. *Shape Classification*

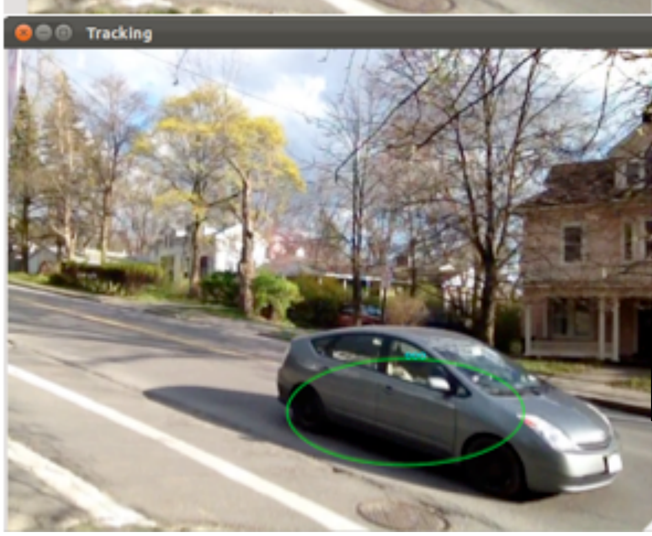
In this project we are mainly to track and avoid road vehicles. Thus to improve the accuracy of foreground segmentation, we use Support Vector Machine to classify moving objects as vehicles and non vehicles, and feed position data of vehicles to the tracking algorithm in the later stage.

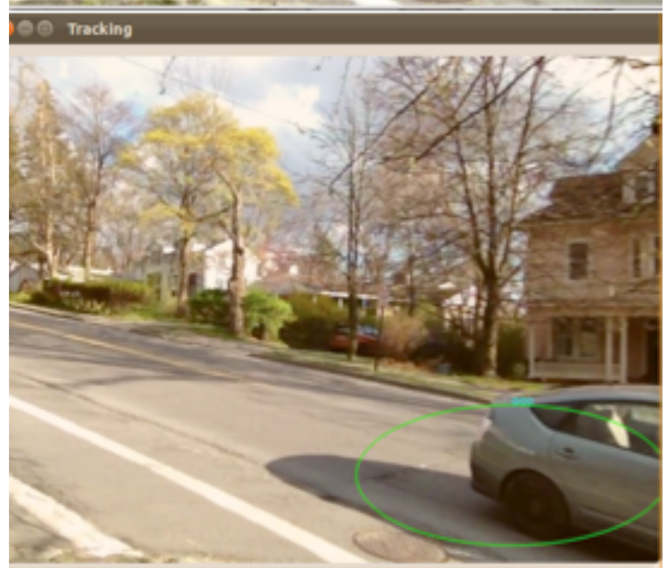
D. *Object tracking*

This task can be achieved by implementing Kalman filter [9]. The algorithm is shown in the graph below.



The following image sequences show the effectiveness of tracking algorithm.





IV. Results

A. *Performance of classifier*

We use a dataset including 80 images with vehicles presented and 100 images without vehicles. We used half of the dataset for training the classifier and the remaining for testing. The feature vector components include size, aspect ratio, width and solidity of the object. The accuracy is shown in the table below.

True Positive	True Negative	Average
87.5%	90%	88.9%

B. *Performance of the tracker*

We performed a series of tests to determine the accuracy of moving object detection when camera is stationary and when the camera is moving. We also implement the classifier to see if there is any improvement. To determine the accuracy, I used 3 videos with stationary camera and 3 movies with moving camera. The result is shown in the following table.

Track without classifier using stationary camera				
	Video 1	Video 2	Video 3	summary
success	6	12	8	26
fail	0	1	1	2
total	6	13	9	28
percentage	100.00	92.31	88.89	92.85

Track without classifier using moving camera				
	Video 1	Video 2	Video 3	summary
success	8	9	8	25
fail	2	3	3	8
total	10	12	11	33
percentage	80.00	75.00	72.73	75.75

Track with classifier using stationary camera				
	Video 1	Video 2	Video 3	summary
success	6	11	9	26
fail	0	2	0	2
total	6	13	9	28
percentage	100.00	84.61	100.00	92.85

Track with classifier using moving camera				
	Video 1	Video 2	Video 3	summary
success	9	11	9	29
fail	1	1	2	4
total	10	12	11	33
percentage	90.00	91.67	81.81	87.87

V. CONCLUSION

We were able to successfully implement the tracking algorithm to detect moving objects. In stationary camera scenario, the tracker works good, with accuracy 92.85%, while in the situation of moving camera, which is always the case for aerial robot, the performance of tracking algorithm falls significantly, although much effort has been put to stabilize the image. By adding the classifier, we will see improvement from the table. Except the offset that the imposed restriction to track some specific moving objects, in this case the road vehicles, and the classifier requires offline training.

ACKNOWLEDGMENT

We would like to thank Prof. Ashutosh Saxena and the course staff of CS4758, in particular, Ian Lenz, for their invaluable guidance and feedback throughout the project.

REFERENCES

[1] Sridhar Srinivasan and Rama Chellappa. Image stabilization and mosaicking using the overlapped basis optical flow field. In *Proceedings of IEEE International Conference on Image Processing,* 1997.

[2] MichalIrani,RennyRousso,andShmuelPeleg.Recoveryof ego-motionusingimagestabilization.InProceedingsof *the IEEE Computer Vision and Pattern Recognition,* pages 454–460, March 1994.

[3] Alberto Censi, Andrea Fusiello, and Vito Roberto. Image stabilization by features tracking. In *Proceedings of the 10th International Conference on Image Analysis and Processing,* pages 665–667, Venice, Italy, September 1999.

[4] I. Zoghlami, O. Faugeras, and R. Deriche. Using geometric corners to build a 2d mosaic from a set of images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pages 420–425, 1997.

[5] JinmanKang,IsaacCohen,andGerardMedioni.Continuou smulti-viewstrackingusingtensorvoting.InProceedings *of the IEEE Workshop on Motion and Video Computing,* pages 181–186, Orlando, Florida, December 2002.

[6] CarineHue,Jean-PierreLeCadre,andPatrickPe´rez.Aparti clefiltertotrackmultipleobjects.InIEEEWorkshopon *Multi-Object Tracking,* pages 61–68, Vancouver, Canada, July 2001.

[7] Dirk Schultz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proceedings of the 2001 IEEE*

International Conference on Robotics and Automation, pages 1165–1170, 2001.

[8] Boyoon Jung and Gaurav S. Sukhatme. Detecting Moving Objects using a Single Camera on a Mobile Robot in an Outdoor Environment. In the 8th Conference on Intelligent Autonomous System, pages 980—987, 2004

[9] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. SIGGRAPH 2001