Tracking of Small Unmanned Aerial Vehicles

Steven Krukowski
Aeronautics and Astronautics
Stanford University
Stanford, CA 94305
Email: spk170@stanford.edu

Adrien Perkins
Aeronautics and Astronautics
Stanford University
Stanford, CA 94305
Email: adrienp@stanford.edu

***Abstract*—Visual tracking of small unmanned aerial vehicles (UAVs) on a smart phone can be quite a daunting task. In this paper an image processing algorithm is outlined in order to be able to assist a user in visually tracking small UAVs. Due to the small nature of the target, simple template matching or optical flow provided to be too process intensive and unreliable for real time tracking. To improve accuracy and speed, a motion model is used to limit the search area used, resulting in performance levels required for real time processing.**

I. Introduction

A. Motivation

As the use of small unmanned aerial vehicles (UAVs) increase today, so does the desire to use them as a platform for photography and filming. One area of interest is the ability to use these vehicles as a platform for filming other UAVs that are in flight. Unfortunately one of the most challenging parts of this process is the task of visually tracking a target in the live stream from one of these cameras. Many commercial systems today use a smartphone as the medium to display the live stream which adds to the challenge of finding the desired target in the video frame. Figure 1 shows an example frame from the display on a smartphone which shows how difficult it can be to locate the actual UAV in the frame. In an effort

Fig. 1. Flying Small Unmanned Aerial Vehicle

to assist a user that desires to film a small UAV in flight, this project sets out to be able to track a small UAV in real time. Template matching and optical flow techniques are explored in order to be able to reliable track a very small object in a live video frame. In order to meet the requirements of a live video stream, these techniques have been augmented to be able to speed up the long computation time required in order to localize the target within a frame.

Beyond the assistance of a user, there are a handful of potential applications for the ability to track small targets in the frame. The end goal is to be able to wrap a navigation loop around the information from the image tracking in order to be able to autonomously pan/tilt the camera and move the vehicle in order to keep the target centered in the frame.

B. Background

In searching for algorithms used for feature tracking, two different methods came up often: template matching and feature tracking with optical flow.

1) Template Matching: Template matching is used to detect and locate a portion of an image within the larger image. Template matching can be improved if a frequency response of the noise is known by a process known as matched filter, but if the noise is assumed to be white noise, then matched filtering and template matching are identical. The goal of template matching is find a location in the image, at which the mean squared error between the template and the image is minimized.

$$E[i,j] = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} [s[x,y] - t[x-i, y-j]]^2 \quad (1)$$

. Minimizing the mean squared error also equates to maximizing the area correlation between the image and the template,

$$r[i,j] = s[i,j] * t[-i,-j] \quad (2)$$

[1]Template matching works best when the portion of the image, or template, is an exactly duplicate of what is found in the larger search-space image; however, this method is also produces acceptable results in the presence of noise. Template matching is not robust to large changes in scale and rotation between the template and its location in the image. For our project, the metric of mean squared error was used and the noise was assumed to be white noise. More details on our implementation of template can be found in Section II.

2) Optical Flow: Simply put, optical flow is the measure of how far a pixel move from one frame to another in a video create. The knowledge of the travel of a set of pixels between subsequent frames can then be used in order to determine crucial velocity information of a specific target in a video frame.

Optical flow is a commonly used technique for feature tracking in a video frame and has many different possible

implementations. There are many different optical flow algorithms available but they almost all rely on the comparison of key features from one frame to another in order to determine the motion information of those features. One of the more widely used implementations found was the combination of Shi-Tomasi good features and a Lucas-Kanade optical flow algorithm.

The Lucas Kanade optical flow has a couple key assumptions that have to be valid in order for it to yield reliable results: 1. Brightness constancy between frames, 2. Small motion between points in ubsequent images, and 3. Points will move like their neighbors [5]. In designing the algorithm used in this project to be able to track a small UAV, it was critical tobe able to satisfy these assumptions in order to be able to use this optical flow technique.

C. Related Work

This project started as a follow on to Padial and Hammonds EE 368 Project from Spring 2011 [2]. Their focus is more on the detection and classification of aircraft rather than tracking. In order to successfully classify an object, they relied on the use of SIFT features and RANSAC. We explored similar techniques, but found them to be inappropriate for our specification application and goal. First, Hammond and Padial note that this procedure ran too slow to work in real-time, which would also not be appropriate for our projects goal. Also, in order to have enough SIFT features to be used for classification, many of their test cases they were trying to detect were fairly large compared to the image. This is also not appropriate for our project because we are trying to track objects that are relatively small in the images. Other work focused on detection was Obstacle Detection for Small Autonomous Aircraft Using Sky Segmentation by McGee, Sengupta, and Hedrick [3]. The goal of this work was to detect an aerial object, but segmenting portions of the image as sky and non-sky regions. Since the regions of sky were relatively uniform, an aerial object would stand out in the sky region. Although this would be appropriate for the small scale objects we are trying to track, our goal is to track an object on a potentially changing background (sky and ground). An aircraft may stand out on a blue sky background, but not stand out when viewed from above with land in the background. In their paper, Tracking-Learning-Detection, Kala, Mikolajczyk, and Matas describe an algorithm which is able to track a moving object based on an initial template [4]. They detect all possible appearances of the object in a frame and model the system as a discrete dynamical system to determine whether the appearance is the object being tracked. Also, they use a learning model to accumulate different templates of the same object as it changes in a video. The also use the learning model to accumulate false positive templates, to help reject background objects that look similar to the tracked object to improve performance. The work done in this paper goes far beyond the scope of a class final project. Some of the techniques used in this paper were explored and implemented, but our goal was not to replicate or implement their algorithm for a specific application.

II. Method

A. General Approach

The first step in the algorithm designed is to get user input as to the object to track. This provides the initial template for the template matching and the initial good features for the optical flow.

The first options explored were the use of solely template matching or optical flow in order to do the live tracking of the small UAV in a video frame. Unfortunately neither of these by themselves were able to process quickly enough in order to make them effective for live tracking. As a result an alternative approach was taken which was heavily influenced by experience in navigation.

Instead of simply using just template matching or just optical flow on the entire image, the goal was to leverage known information about the target being tracked. Tracking a small UAV means that the object will be constrained to a certain set of physically realizable motions that can help reduce the search space for template matching and optical flow. The idea then was to use a motion model of the vehicle in order to minimize the amount of the image being searched. In the process developed, shown in figure 2, for each new frame, the motion model is used in order to update the search area for template matching. Once the template is matched and the target is found, optical flow can be used to provide updated velocity information to feed back into the motion model to be used in the next frame. Using a motion model to reduce the search area also provided for the added benefit of reducing the number of false positive detections of elements in other parts of the video frame. In this process, the motion

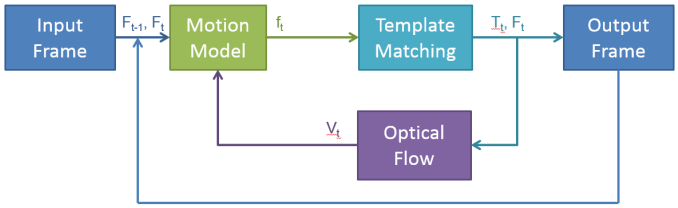

Fig. 2. General Tracking Approach

model is key to be able to reduce the search space to a reasonable size in order for both the template matching and optical flow to run smoothly on a live video. The motion model, shown in equation 4, is deceptively simple.

$$i' = i + \delta i + \epsilon \quad (3)$$

$$j' = i + \delta j + \epsilon \quad (4)$$

For each new frame, the corners of a new search area (i, j) are determined by updating the corners of the previous search area (i, j) by the motion δI and δJ.

B. Template Matching and Updating

Our approach to template matching involves four steps as show in figure 3. First, a motion update is performed to estimate the location of the aircraft in the new frame using

Fig. 3. Template Matching Steps

our motion model. This reduces the search area for template matching to a box around this location along with giving us a probabilistic estimate of where the aircraft is in the new frame. The motion is assumed to be normally distributed around the expected value as shown in equation 5.

$$\mathcal{P}\left(i', j' | i, j, \delta i, \delta j\right) \propto e^{\frac{-\left((i'-i)^2+(j'-j)^2\right)}{\sigma_v^2}} \tag{5}$$

Next, template matching is performed within this limited search space using mean removed, normalized grayscale images. The results of the template matching are assumed to also be normally distributed as shown in equation 6.

$$P\left(i'', j'' | i', j', \mathcal{T}_{t-1}, \mathcal{F}_t\right) \propto e^{\frac{\sum_{i,j}\left(\mathcal{F}_t(i+i'', j+j'') - \mathcal{T}_{t-1}(i,j)\right)^2}{\sigma}} \tag{6}$$

The next step is to multiply both the motion probability and the template matching probability for a total probability. The updated position estimate is chosen to be the max likelihood position of the total probability. The final step is to update the template by selecting the pixels in the appropriately sized region around the update aircraft position estimate. The updated template is used for the matching in the next frame of the video. The initial template to be used is chosen by the user.

The template is updated from frame to frame to account for changing perspectives on the vehicle throughout time. At a high enough frame rate, the perspective on the vehicle does not change significantly between two adjacent frames as shown in figure 4; however, the perspective of the vehicle is significantly different between two temporally separated frames. Another reason for updating the template is due to

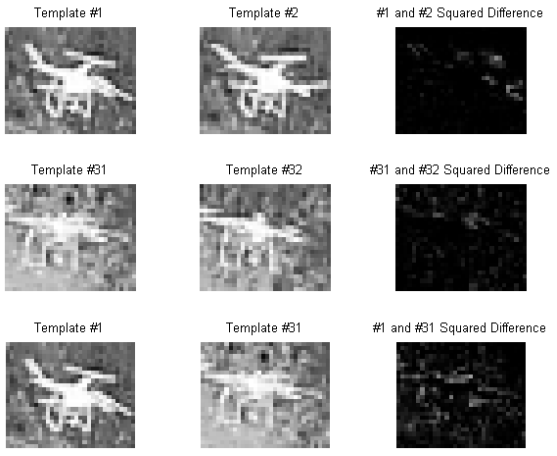

Fig. 4. Vehicle Perspective Changes

changes in the background in the template. Two adjacent frames will have similar backgrounds, but as shown in figure 5, the relative intensity of the background (compared to the vehicle intensity) changes significantly throughout time. Due to some of the template including background pixels, not changing the template could result in background correlation outweighing the correlation of the object to be tracked.

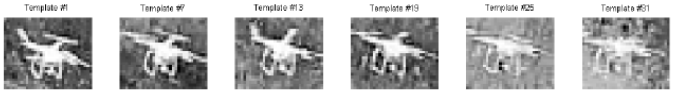

Fig. 5. Template Background Changes Over Time

In order to account for scaling changes over time, template matching is also done with a template 10 percent larger and 10 percent smaller than the original template. The size template which performs best out of the three on the total probability is chosen as the next template size. This allows the template to grow and shrink as the scale of the target aircraft increases and decreases. The test cases used were quadcopters, so many of the templates were near-square in shape. Similar consideration could be given to rotation to tracking an aircraft, where at times a more rectangular shaped template is appropriate. In such a case, frame to frame the template would not change shape very much, but over time the template would need to change shape.

C. Optical Flow

In order for optical flow to yield reliable results there are three key assumptions made about the frames of the video: 1. Brightness constancy between frames, 2. Small motion between points in subsequent images, and 3. Points will move like their neighbors. Some considerations needed to be made in order to make all three of these assumptions valid.

Working backwards, the third assumption may not be globally valid across the entire frame, but since the user is selecting the initial template to be used, by limiting the points to points within that template we can ensure that points will move like their neighbors for the most part. The small motion assumption is satisfied if the frame rate of the video is fast enough to be greater than the travel motion of the vehicle which brings the requirement of being able to do fast computation of the tracking algorithm. Finally the brightness constancy is definitely not valid across the entire video, especially if the object is moving in very different parts of the screen. However an assumption is made that between local frames that holds true and therefore if good features to track are updated periodically on the new template, it will ensure that the brightness constancy requirement is met.

Updating the good features periodically with the new template also helps to ensure the ability to track a small UAV even under different orientation changes throughout the flight.

Using the available functionality within openCV for optical flow, the overall process is straightforward. First a set of good features is required in order to do the tracking and is determined using openCV to get a set of Shi-Tomasi

good features. With these features, the optical flow can be calculated using openCVs pyramid implementation of the Lucas-Kanade method. From here the average velocity of all the good features can be determined in order to update the motion model.

For the motion model update it was determined that δI and δJ are best not as simply the difference between two frames. It was found that the motion model was much more reliable by using a moving average of the velocities of the objects, as velocity can be a very noisy measurement.

III. Experimental Results

A. Matlab

The template matching was tested using pre-recorded videos in Matlab of a quadcopter flying over two backgrounds. In the first, a quadcopter taking off was recorded from a stabilized camera on another station quadcopter. The second video is taken of a quadcopter with a sky background. Neither incorporates optical flow into the motion model because that was only implemented on Android so the observer is assumed to be stationary. Template matching was implemented but not tested on Android. Matlab testing of the template matching does not incorporate changes in scale or rotation to the template (Android implementation incorporates scale changes).

The results from the first video can be seen in figures 6 and 7. A color version of the template is shown in the first

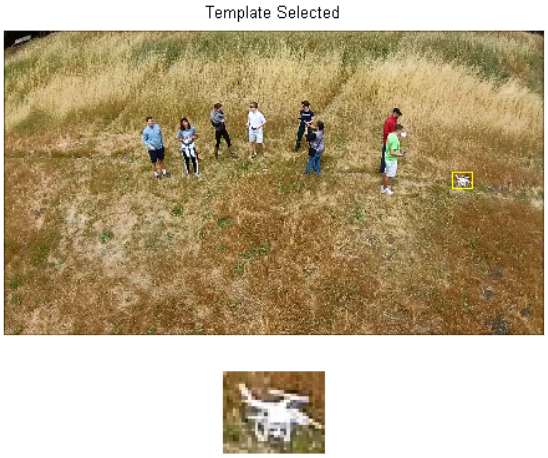

Fig. 6. Test Case #1 Template

frame. In figure 7the vehicle path (blue line) with the final template (yellow box) and the final search area box (green box). The bottom of the figure shows the template used for the searching and the resulting template found along with the complete search area. On the right hand side of the results, the template matching mean squared difference results, the template matching probability, the motion probability, and the resulting total probability is shown for the entire search area. This particular frame is significant because it shows the need for the motion model. The results of the template

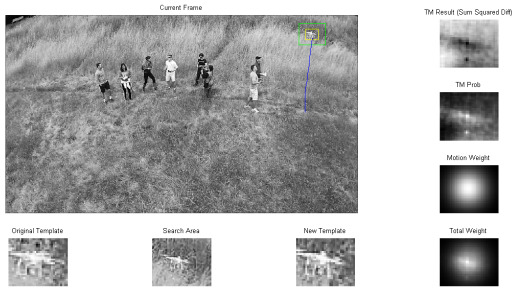

Fig. 7. Test Case #1 Results

matching by itself shows a high probability area centered near the bottom of the search area. Including the motion model in the total weight, lowers the probability of this false positive result, which results in the correctly chosen vehicle position. Qualitatively, this example performed better when the initial template size was chosen tight around the vehicle to reduce the amount of background pixels in the template. If the template was chosen too large, the vehicle would drift from the template, especially in the lighter intensity background where the vehicle is close in intensity to the grass.

The results from the second video can be seen in figures 8 and 9.

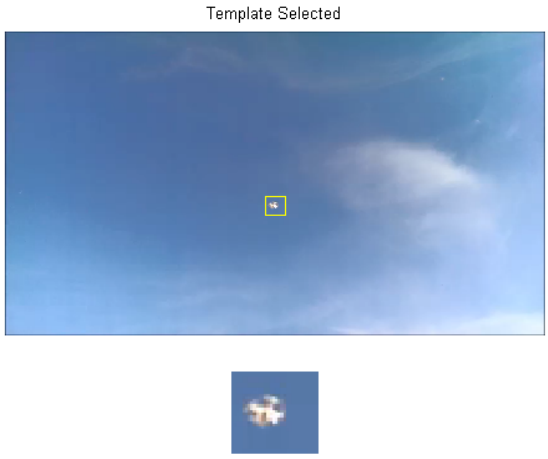

Fig. 8. Test Case #2 Template

The observer also pans the camera during this example. Due to uncertainty in the camera motion, the variance in the motion noise to better fit the problem. The performance of this example was not as robust as the first example because of this higher uncertainty in the motion and due to a smaller scale on the quadcopter. Since the background was uniform, this example was found to perform best when the template was not as tight around the tracked aircraft as the first example. Otherwise, the tracked aircraft would drift from the template due to the observer motion not accounted for in our motion model. With an appropriately sized template and well-tuned parameters, the aircraft could be tracked for a while before

Fig. 9. Test Case #1 Results

TABLE I. APPROXIMATE FRAME RATE ON MOBILE DEVICE

Method	Search Area	Approximate Frame Rate (fps)
Optical Flow	Entire image	3-4
Optical Flow	Motion limited	13-15
Template Matching	Entire image	3-4
Template Matching	Motion limited	10-12

drifting out of the template. Future consideration should be given to ensure that this drift is minimalized, possibly using the originally chosen template or a subset of past templates.

B. Android

The end goal was to be able to run the image processing algorithm on an android phone in real time. Frame rate of the video was an important metric in being able to make sure that the algorithm used was able to truly run in real time. As seen in table I, neither optical flow by itself or template matching on the entire image would be suitable for real time processing due to the very low frame rates provided. Using the motion model to reduce the search area allowed for drastically reducing computation time and therefore increasing the frame rate. Unfortunately the entire algorithm together has not been implemented on the Android phone yet, but we are confident in being able to achieve the desired frame rate of 10 fps.

The optical flow portion of the algorithm was successfully implemented on the Android phone and a screenshot can be seen in figure 10. In the testing done with the phone, the optical flow algorithm seemed to be extremely sensitive to the parameters used to determine the good features to track. In some cases where the UAV was too small in the frame it required very loose parameters in order to determine any good features, but this would also lead to tracking undesired background features that resulted in very poor performance of the optical flow. When the UAV was a bit larger in the frame and good features were able to be found it was able to provide very reliable velocity information.

It was also seen that the velocity information was rather noisy

Fig. 10. Test Case #1 Results

and therefore would require some filtering and smoothing before being able to be used in the motion model reliably.

IV. CONCLUSIONS AND FUTURE WORK

Overall the initial phases of the algorithm developed here performed fairly well in the Matlab testing done. The use of a motion model to reduce the search area significantly reduced both processing time (and therefore increased frame rate) and false detection rate (by only searching the area around the template). A lot of challenges were faced with the integration of the algorithm into Android, but that is still in active development as we hope to get this process successfully running on the Android phone.

An area of continuing work on this would be to be able to integrate the tracking of a target in a video to the control and navigation of another UAV in order to keep the target centered in the frame. In order for this to happen, the algorithm will have to be able to take into account the motion of the camera in order to successfully keep tracking the UAV.

Eventually, instead of just having the optical flow in the feedback path of the algorithm, it can also be used in the feed forward path in order to cross check the results of the template matching to help make the algorithm more robust. This can potentially be done with kmeans clustering, using 2 clusters, one for background velocity and one for object velocity, and training on the previous frame and categorizing on the next frame.

APPENDIX

DIVISION OF RESPONSIBILITY

Most of the work was done collaboratively, but Adrien Perkin's area of responsibility was the Optical Flow and Steven Krukowski's area of responsibility was the Template Matching

REFERENCES

[1] B. Girod, *Template Matching 2*, EE 368 Digital Image Processing, Stanford University, 2013.

[2] M. Hammond and J. Padial *Automatic Aircraft Detection and Classification in Cluttered Environments*, EE 368 Digital Image Process, Stanford University, 2011.

[3] T. McGee, R. Sengupta, and K. Hedrick. *Obstacle detection for small autonomous aircraft using sky segmentation.* Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE, 2005.

[4] K. Kalal, K. Mikolajczyk, and J. Matas. *Tracking-learning-detection.* Pattern Analysis and Machine Intelligence, IEEE Transactions on 34.7 (2012): 1409-1422.

[5] J.Y. Bouguet *Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm*, Intel Corporation 5, 2001.