Qiuwei Shou
Alan Wu
5/4/16
B657 Comp Vision
Final Project Report

# Object Detection through Forward Movement

## Introduction

Our project is inspired by its applications to autonomous robotics on an embedded platform. We envision an unmanned aerial vehicle navigating through a dense environment, such as a forest, for a search and rescue operation or an area survey. The ability to detect and avoid obstacles is paramount to a successful mission. Our approach is motivated by Mori and Scherer's paper published at the 2013 ICCV conference where they use relative size of SURF features to detect approaching objects. Their algorithm is based on using a monocular camera, which has advantages in a smaller and more affordable payload compared to a stereo (or two) camera(s). Relative size is one of several approaches in using a monocular camera, which the paper's authors outline in their related works section, and briefly summarized here.

Depth calculation is central to obstacle detection and avoidance. Optical flow and structure from motion (SfM) are popular methods, but they are computationally intensive – difficult for an embedded processor to handle – and do not detect obstacles well that lie directly ahead. [Mori] While optical flow showed promising results in outdoor settings [Campbell], it has had difficulty with homogenous textures such as corridor walls. Perspective cues, on the other hand, is "only useful in structured environments" and other methods relying on a priori knowledge such as texture gradient are limited to specific objects. [Mori] Lastly, depth from focus relies on large aperture cameras, which is non-ideal for mounting on small aerial vehicles.

Mori and Scherer leverage fast computation of SURF as the method uses integral images to calculate Hessian-matrix approximations. [Bay] Not only does using SURF over SIFT for computational savings seem appropriate, but a comparative study found SURF to be more robust to "significant changes in scale, viewpoint, and lighting" than SIFT. [Fuentes-Pacheco] It was only appropriate for us to proceed with our project also using SURF.

## Dataset

We derived our dataset from hobbyist drone videos posted on YouTube[www.youtube.com]. These videos are taken from a first-person viewpoint (FPV) of the aerial vehicle and therefore yield the perspective that would see oncoming obstacles, making them suitable for our project. Our dataset consists of nearly 500 image pairs, derived from 18 videos, each of which is taken from a different location. Of these, nearly 400 image pairs (from 13 videos) are taken in dense forest settings and nearly 100 (from 5 videos) are taken in open field settings. We run our algorithm over these two sets of images to obtain our accuracy and false positive measurements. Ideally, our algorithm would report obstacles from all 400 dense forest image pairs and report no obstacles from the 100 open field image pairs.

We manually selected portions (mostly 10-sec clips) of each video that show continuous footage of forward navigation (at a relatively constant velocity) through a dense forest (> 1 tree per 10 meters). From these clips, we used ffmpeg[www.ffmpeg.org] to produce frames for comparing "previous" and "current" images to detect obstacles. Ideally, the physical travel rate is known and the frame interval can be chosen to reflect the desired travel distance of the vehicle, from which one can determine the expected increase in scale of an object to see if it were an obstacle. However, because we did not have this information, we set the frame interval for each clip so that objects would increase in size from "previous" image to "current" image large enough to be deemed an obstacle. This approximation would likely yield a lower accuracy rate than what our algorithm yielded since some frame intervals might not actually produce objects that have increased enough in size. In [Mori], obstacle poles are clearly marked with "X" posters whereas we do not mark any of the obstacles in our image dataset. Figure 1 shows a sample of our forest dataset and Figure 2 shows a sample of our open field dataset.

Figure 1. Sample Images from Forest Dataset

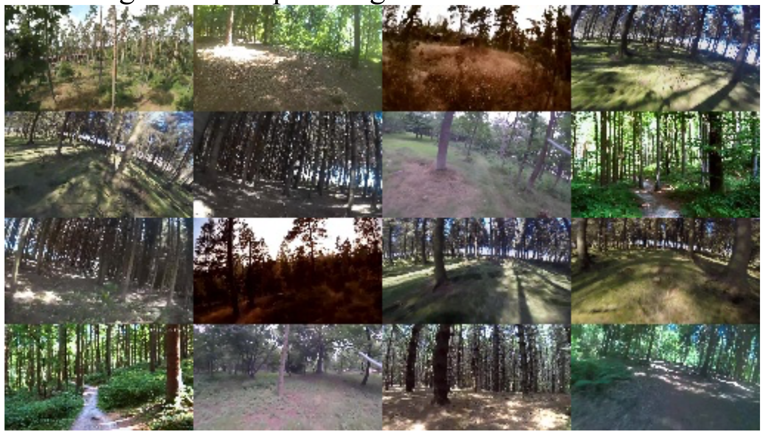
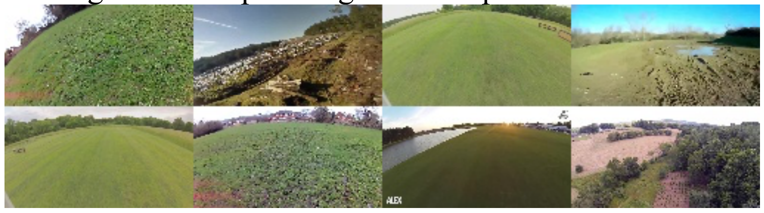


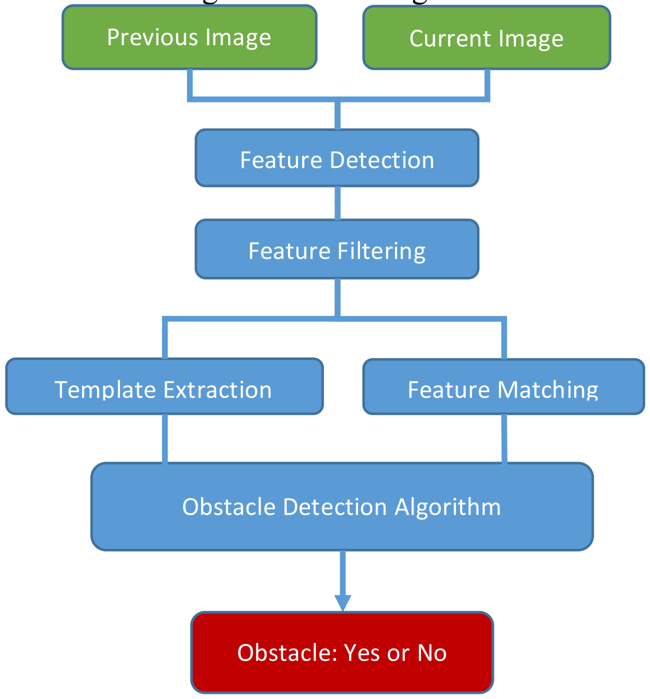
Figure 2. Sample Images from Open Field Dataset



## Methodology

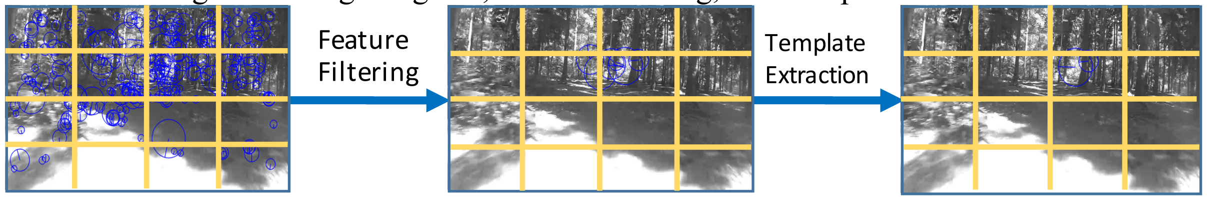
We use Speeded-Up Robust Features (SURF) method to quickly extract features from the "previous" and "current" images. [Bay] Figure 3 shows the flow of our approach.

Figure 3. Flow diagram



With each image divided into 4x4 regions (Figure 4), we discard features detected on the bottom row regions (reflecting objects on the ground that would not be contacted by an aerial vehicle) and side regions of the "previous" image only, but we keep all features found in the "current" image as features move in the radially outward motion. We want to be able to identify when an object moves radially outward, especially if it ends up in an outer region. If we prematurely remove a feature in "current" frame, the feature may erroneously be matched to another feature that is within the central regions just because "a" nearest neighbor exists, but not "the" nearest neighbor. If we find that "the" nearest neighbor resides in one of the outer regions in the "current" frame, we can confidently discard this feature (as we would be passing it and not run into it).

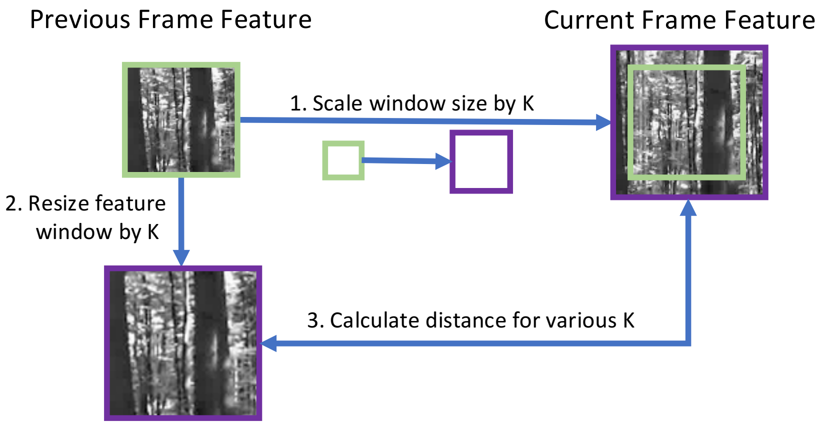
Figure 4. Image Regions, Feature Filtering, and Template Extraction



Because SURF is a blob detector [Bay], the tool reports the size of each feature. We also discard any features whose size does not increase from “previous” image to “current” image. Furthermore, we use template matching for increased accuracy in measuring scale increases of object features. Mori and Scherer reported that using template matching yield more accurate results than SURF blob size measurements. In addition to relative size, the absolute size of the features in the “current” image must be greater than a threshold to indicate that it is of an impending obstacle, as opposed to something that grew in size but also far away, which would not be considered as an obstacle.

After filtering, we draw a small window of initial size W around each remaining feature in the “previous” frame. A similar window is drawn around the matching feature (via nearest neighbor) in the “current” frame, scaled up by K, where K is a parameter ranging from 1.1 to 1.6 in increments of 0.1, reflecting the size increase of the object. We resize the same feature (as opposed to rescaling the window size) in the “previous” frame also by K. Scalar K is determined such that the correlation distance (SSD) between the “previous” and the “current” windows is minimized. We discard any matches with distance $D > D_{max}$ and $K < K_{min}$. For our experiments, we set $D_{max} = 0.8$ and $K_{min} = 1.1$. Figure 5 illustrates our template matching approach.

Figure 5. Template Matching


Previous Frame Feature
Current Frame Feature
1. Scale window size by K
2. Resize feature window by K
3. Calculate distance for various K

Although it is part of our algorithm to determine K to achieve the minimum correlation distance, we do not specifically use this value of K other than seeing if it is within our parameter range (1.1 to 1.5) to qualify as an obstacle. However, K is useful in dealing with real-time applications in that the ability to pinpoint K will yield accurate time to collision approximations that can be inferred from object size growth rate and vehicle velocity. Figure 6 shows that K=1.3 yields the minimum distance between the templates shown in Figure 5.

Figure 6. Template Correlation Distance vs. Scale Factor K

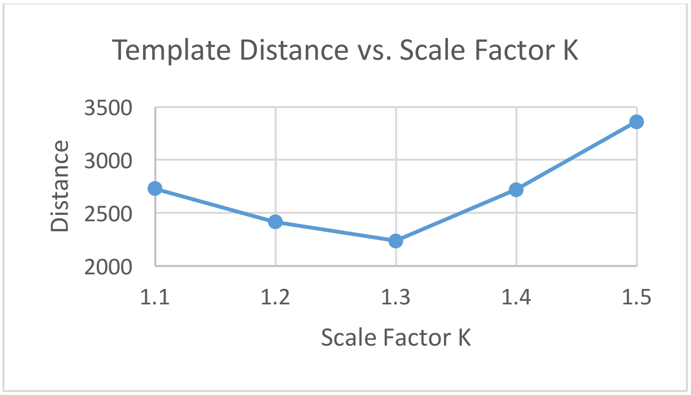
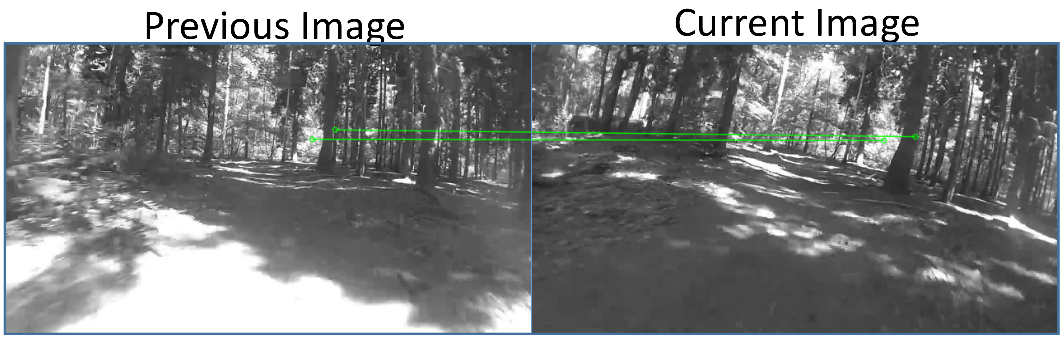


Figure 7 shows an example of the remaining features after template matching. Note that the left feature in the both images is that of the background. This points to the weakness of SURF that it does not work well with noisy backgrounds and picks up extraneous features often difficult to filter out without a priori knowledge of the object one is attempting to detect.

Figure 7. Example Feature Match after Template Matching.



It is worth noting that the selection of the initial window size is important as it affects accuracy and false positive calculations. If the initial window size is too small, the feature will be too vague and even with increases in K, almost all NN-matched in the "current" frame features would pass. If too large, then we have over-fitting. Either case would defeat the purpose of using template matching. We select the initial W value based on data run on a baseline set of images derived from a subset of the dense forest and open field datasets. The effects of initial W are shown in Figure 8. Setting W = 30x30 pixels was found to yield good accuracy (90%) while minimizing false negatives ~15%. The performance on the overall dataset, however, was slightly worse, as shown in Figure 9.

Figure 8. Effect of Initial Template Window Size on a Baseline (more Idealized) Dataset

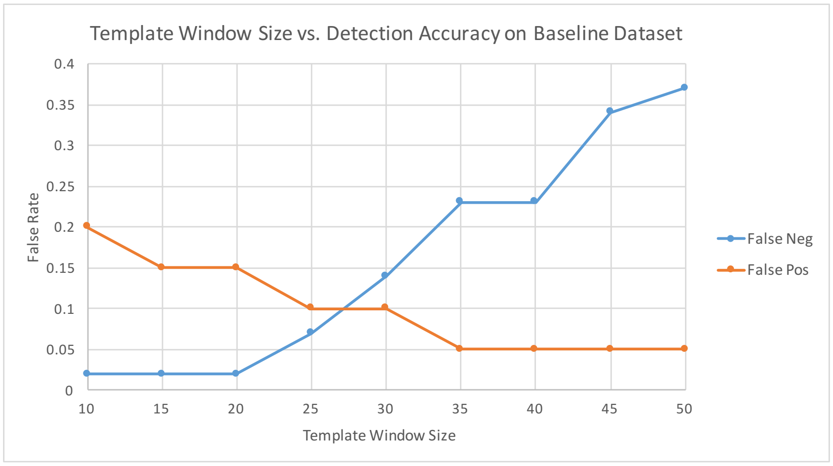
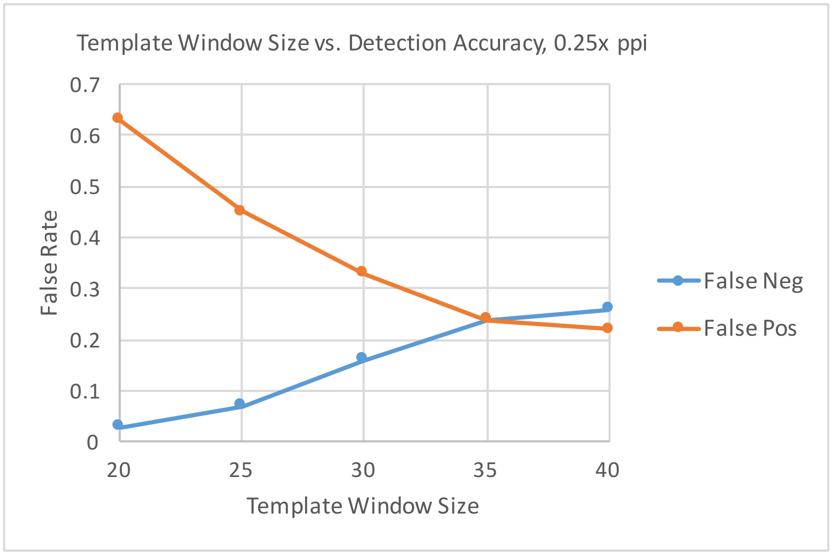


Figure 8. Effect of Initial Template Window Size on Entire Dataset

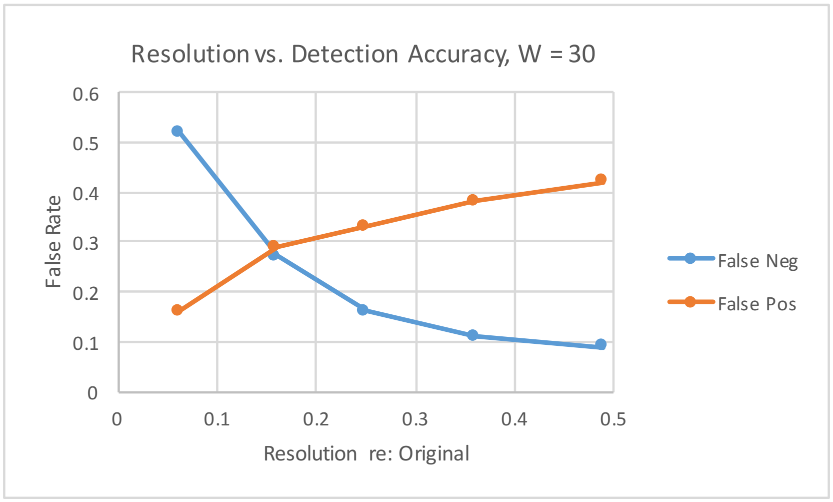


## Results and Discussion

We assumed there would be objects fitting the description of an obstacle found between each "previous" and "current" image pair we present to our algorithm in the dense forest dataset (dataset 1) and that there would be no obstacles in in the open field dataset (dataset 2). As such, we were able to obtain 84% accuracy in detecting obstacles in dataset 1 and a 33% false positive rate on dataset 2.

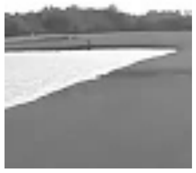
In an attempt to decrease computation time or allow the use of cheaper lower resolution cameras, we investigated the effect of down-sampled images on accuracy. The original videos were 720p (1280x720). We found that images resized to ¼ of the original resolution yielded the optimal tradeoff in accuracy and false positives. Further down-sampling to keep only 0.15 of the original yielded respectable results, but a sharp increase in false negatives results – features are getting missed when the resolution gets low. The effect of resolution on accuracy is shown in Figure 10.

Figure 10. Resolution vs. Detection Accuracy



One of the greatest challenges we faced was to define an "obstacle." We settled with a broad definition that included "any object that grows in size and is close (i.e. large) enough." This, however, also included objects on the ground that grew in size, but were clearly not an obstacle an aerial vehicle would collide with. Not only is this apparent from forest dataset, but also in the open field dataset. See Figure 11. The corner of a lake is detected as a feature that grew in size and thus falsely classified as an obstacle.

Figure 11. Sample False Positive Template: Corner of Lake Determined as Obstacle



Obstacle detection is of little relevance without the ability to avoid collisions. While our project demonstrated the ability to use SURF to detect obstacles, which is implementable in real-time processors, a more comprehensive set of odometry would be needed for robust navigation. At the very least, vehicle velocity would be required. Without it, we were left to estimate our extraction of frames so that, virtually, the aerial vehicle would have needed to gain enough forward movement to detect our definition of an obstacle. Furthermore, in keeping to a monocular camera, depth calculation is possible with visual SLAM research, most notably the development of the MonoSLAM [Davidson] algorithm.

We attempted using RANSAC to add robustness to our feature matching, but the computation was too slow even on our laptops. One possibility is to push the computation onto a server and wirelessly transmit the result to the UAV. If we venture into cloud robotics [Kehoe], we could also employ texture cues using offboard computation as long as we can assume a robust and wideband wireless link. Michel's investigation using textural cues could be useful in our case to distinguish bark, for example, from a body of water would increase accuracy and decrease false negatives. [Michel] Recent development such as the use of edge features in real-time to determine depth [Tarrio] appears to be a promising avenue of pursuit. We must also not forget that obstacles are not necessarily stagnant. Those that traverse horizontally across the robot's view (such as a pedestrian) must also be accounted for. Optical flow in this case seems appropriate, especially with demonstrated ability to operate in real-time on an embedded system [Briod], and thus should be incorporated in the future for a more robust obstacle detection and avoidance algorithm.

## Works Cited

Mori, Tomoyuki and Sebastian Scherer. "First Results in Detecting and Avoiding Frontal Obstacles from a Monocular Camera for Micro Unmanned Aerial Vehicles." ICRA, 2013.

Fuentes-Pacheco, Jorge et al. "Visual Simultaneous Localization and Mapping: A Survey." Artificial Intelligence Review, Volume 43, Issue 1, Jan 2015.

Bay, Herbert et al. "Speeded-Up Robust Features (SURF)." ECCV, 2006.

Agrawal, Pulkit et al. "Learning to See by Moving." ICCV, 2015.

Tarrio, Juan Jose. "Realtime Edge-based Visual Odometry for a Monocular Camera." ICCV, 2015.

Campbell, J. et al. "A Robust Visual Odometry and Precipice Detection System Using Consumer-grade Monocular Vision." ICRA, 2005.

Michels, J. et al. "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning." ICML, 2005.

Davidson, Andrew J. et al. "MonoSLAM: Real-Time Single Camera SLAM." PAMI, 2007.

Floreano, Dario and Robert J. Wood. "Science, Technology, and the Future of Small Autonomous Drones." Nature, Vol. 521, May 2015.

Briod, Adrien et al. "Optic-Flow Based Control of a 46g Quadrotor." IROS, 2013.

www.opencv.org

www.youtube.com

www.ffmpeg.org