Obstacle avoidance by combining background subtraction, optical flow and proximity estimation

Jaysinh Sagar and Arnoud Visser

ABSTRACT

For Micro Aerial Vehicles (MAVs), robust obstacle avoidance during flight is a challenging problem because only light weight sensors as monocular cameras can be mounted as payload. With monocular cameras depth perception cannot be estimated from a single view, which means that information has to be estimated by combining images from multiple viewpoints. Here we present a method which focuses only on regions classified as foreground and follow features in the foreground to estimate threat of potential objects being an obstacle in the flight path by depth segmentation and scale estimation. We evaluate results from our approach in an outdoor environment with a small quad-rotor drone and analyze the effect of the different stages in the image processing pipeline. We are able to determine evident obstacles in front of the drone with high confidence. Robust obstacle-avoidance algorithms as presented in this article will allow Micro Aerial Vehicles to navigate semi-autonomously in outdoor scenarios.

1 INTRODUCTION

In the domain of robot technology, we see a lot of progress using sophisticated hardware and software coming together to create a functional machine to accomplish varying tasks replicating human activity. Today, MAVs (Micro Aerial Vehicles) are used for tasks such as surveys, surveillance, search and rescue, and military applications which would otherwise be difficult or infeasible for humans considering harsher and challenging conditions. However, this is a challenging problem as activities that are simple and latent knowledge for living beings, are non-trivial when replicating on machines. Technology in this domain still has a way to go to achieve near human intelligence, but today, research tackles several sub-tasks which can someday be integrate into a system.

Specific kinds of drones have been developed such as flappy-wing drones, fixed-wing drones, and rotor based drones. Flappy wing drones have an advantage of being able to hover and fly close to objects being small, agile, and less dangerous due to their low weight. Fixed-wing drones are UAVs that use a gliding mechanism and use either external propulsion mechanism or built in linear propellers. Unlike flappy wind drones, fixed-wing drones are not very agile but have longer flight durations and are extensively used for surveying and mapping applications.

In this study, we look at the challenge of obstacle avoidance using a quad-rotor drone in realistic environments with stationary obstacles. For this, we use the A.R.-Drone from Parrot S.A. which has the advantage of stabilized flight [1].

Previous approaches attempt to imitate biological approaches for obstacle detection and avoidance using optical flow in peripheral vision or stereo disparity from both eyes. However, these approaches are not suitable for oncoming obstacles directly in front of the camera. Scale based texture template matching algorithms mentioned in [2] may be used to detect on-coming obstacles but are subject to camera resolution and object texture. Supervised and semi-supervised approaches may be used as well if obstacle properties are known but may be challenging to provide in unknown environments.

The proposed method consists of a modular algorithm in order to generate confidence results using background/foreground classification, optical flow and filtering on affine relationships and scale estimation. The result of the algorithm generates a confidence mask used to direct and control the drone avoiding immediate obstacles. While clearly visible obstacles are effectively detected, performance of our method is subject to lighting and stability conditions. We employ existing methods of background/foreground classification and optical flow in conjunction with clustering, filtering and flow estimation methods to further improve avoidance results and effectiveness.

In section 2, we discuss related work in this area of study looking at advantages and limitations of these approaches. Section 3 touches upon general methods used in our obstacle avoidance application. Section 4 discusses the proposed method of this study followed by experiments, results and discussion in section 5. We then conclude and discuss future work in section 6.

2 RELATED WORK

In recent years, obstacle avoidance in MAVs has been a keen area of interest. Several approaches exist using monocular vision for detecting obstacle threats and avoiding such threats based on optical flow as well as using feature descriptors for relative size changes.

Bills *et al* [3] use an approach designed for indoor environments where there is uniformity in structure properties. The paper makes use of edges and detects lines using Hough transform and uses this information to classify the scene

based on a trained classifier. While the paper uses MAVs with a single camera, the proposed method in the paper allows the MAV to navigate in indoor environments where such features can be used. Using these features, a confidence value is determined based on line intersections crossing a visible vanishing point. Based on this confidence value, the MAV adopts a control scheme to react accordingly to its environment. When faced with an unknown environment; when confidence value is small; the MAV enters an unknown state where it adopts defined exploration strategies.

Celik *et al* [4] use an approach that exploits both lines and corner features detected in a scene. Similar to the Hough transform method, the paper uses line information extracted from the scene to determine the scene structure however, instead of running the method on the entire scene, as the Hough transform does, their method focuses only on scene parts where data exists. The next proposition of the paper is using a human like range estimation method using the height of the MAV and vanishing point of the scene. Using this information, the paper states that it is possible to obtain the relative distance of an object in the scene following feature points for its SLAM formulation. The paper uses a variation of optical flow as well for the proposed Helix Bearing Algorithm to create a motion field to determine turning points in the environment. This method is however also subject to camera capabilities and detection of strong features in the scene to navigate successfully.

Lee *et al* [5] talk about the use of MOPS (Multi-Scale-Oriented-Patches) and SIFT feature descriptors to obtain three-dimensional information of the environment. MOPS, similar to the Harris corner detector, is a quick and accurate corner point matching method. Using MOPS and SIFT, the paper attempts to reconstruct objects in 3D space where MOPS is used to extract edge and corner information and SIFT used to detect the internal outline information of an object. By combining these, a distinction between the outline and the inline of objects can be determined using matching distance between frames. By listing cases between the relationships of SIFT and MOPS features in the scene, it is possible to know the nature of objects around when the UAV moves and construct a 3D map of the environment. While the paper highlights experiments where they illustrate results from the proposed approach, additional knowledge of flight test data would help understanding the performance better.

Monocular cues used in the papers above ([3, 4, 5]) utilize information such as lines, corners and scene regularities to detect proximity and location of obstacles. However, the methods do not perform well in natural environments where such cues are challenging to find. Another limitation of these methods is that monocular approaches may not find distance to collision directly however [6] shows us that there exists a relationship between optical flow to time to collision which may be exploited to determine immediate threat of obstacles. Previous work on ground robots [7, 8, 9] use motion cues to estimate depth from motion and background subtraction to determine obstacles in a frontal scene however face the issue of computational cost as well as robustness of detection. De Croon *et al* [10] use a combination of appearance variation cues and optical flow for their approach to obstacle detection. This method however is highly influenced by blob-based texture variations in the environment and has limited performance in outdoor scenarios.

The approach of More and Scherer [2] also combines features in conjunction with template matching to estimate scale, resembles the method in this study. The platform is also similar so a direct comparison of the results can be made. The approach of [2] allows the drone to estimate depth of objects detected as potential obstacles and observes relative changes in size of the sub-image template around that feature point to detect if the potential obstacle appears to come closer to the MAV by a fixed metric. While results of the study are positive for frontal obstacles using SURF features, a limitation of this approach is that performance is subject to presence of sufficient textures in the scene. In cases where there is not sufficient texture available to track (smooth and regular surfaces) the matching of SURF features do not perform as well as in the case of detailed textures. Also, the experiments carried out in this paper focus purely on narrow, tree-like obstacles in an indoor setting.

3 METHOD

The method studied in this paper uses background/foreground classification to find regions of interest in the MAV flight environment. Inside the region of interest potential obstacles are identified by a combination of optical flow, depth classification and scale template matching. The theory behind those techniques is worked out in the following subsections.

3.1 Background/Foreground classification

Background subtraction is a method in computer vision and image processing where an image is classified into the background and the foreground. This is done to achieve a variety of applications which require segmentation of concerned objects from a scene and filtering out everything else. Background subtraction creates a model of the background and then segments out everything that does not belong to the background as foreground based on a spatial and temporal setting. It can be seen as similar to detecting the objects are both in the new image and the old image. Once foreground objects are retrieved in the foreground mask, it is easy to extract as well as localize the objects in the scene.

There are several approaches to achieve background subtraction. Yet, due to the egomotion of our aerial vehicle a highly adaptive method is needed. Our method is inspired by the Improved Adaptive Background Mixture Model proposed by KaewTraKulPong and Bowden [11]. The method proposed is in two parts where we first consider background modelling and then maximizing the expectation over the

background Gaussian mixture model. The background is modelled using an Adaptive Gaussian Mixture Model where each pixel in the scene is modelled by K Gaussian distributions [12].

The probability of a pixel x_N at time N belonging to one of the background components is given as:

$$p(x_N) = \sum_{j=1}^{K} w_j \mathcal{N}(x_N; \mu_j, \Sigma_j) \tag{1}$$

where w_j is the weight parameter of the k^{th} Gaussian component $\mathcal{N}$ with mean μ_j and covariance Σ_j. Usually, if foreground objects are clearly distinguishable they will also be represented with a Gaussian component, but because there will be no evidence in the training set those components should have a small weight w_j. If the components are sorted on descending weights, the background can be modelled with minimal collection of components:

$$B = \underset{b}{\operatorname{argmin}}(\sum_{j=1}^{b} w_j > T_b) \tag{2}$$

where T_b is a threshold which corresponds with a prior estimate of the minimal portion of background objects one expect. The Gaussian Mixture Model is made adaptive by updating each component based on ownership o_j (the closest component with the largest weight w_j, as introduced by [13]) and a constant α which determines the learning rate, i.e. how fast old data is forgotten. The classical updates rules for each Gaussian component, in accordance with [12], for the update between timestep $t = N$ and $t = N + 1$:

$$w_j^{N+1} = (1 - \alpha) w_j^N + \alpha o_j^{N+1} \tag{3}$$

$$\mu_j^{N+1} = (1 - \alpha) \mu_j^N + \alpha \delta_j^{N+1} \tag{4}$$

$$\Sigma_j^{N+1} = (1 - \alpha) \Sigma_j^N + \alpha {\delta_J^{N+1}}^T \delta_J^{N+1} \tag{5}$$

where $\delta_j = x - \mu_j$ is the distance from the pixel x to center of the Gaussian component μ_j. Although the adaptability of this algorithm can be tuned by the learning rate α, convergence could be too slow for the dynamics expected from a flying platform. Bowden *et al* [11] improved the algorithm by a window sampling method for faster convergence. In the L-recent window version the weight w_j^{N+1} is estimated from the previous L time frames, which gives the update rule:

$$w_j^{N+1} = \frac{1}{L} \sum_{t=0}^{L-1} o^{N+1-t} \tag{6}$$

The mean μ_j^{N+1} and covariance Σ_j^{N+1} of each Gaussian component are updated equivalently for the last L time frames.

With this approach, we have selected a method with could identify potential obstacles for the drone. Once we have the potential obstacles classified as foreground objects, we can use the tracking method explained in the next section to keep following those regions of interest.

3.2 Tracking and Depth Classification

We track features based on the classical Lucas-Kanade method for Optical Flow [14], pyramidal implemented as suggested by [15]. This method is based on the assumption that the flow of a concerned pixel in one image to another image should be present in the local neighborhood of that pixel in the first image. To recognize a equivalent pixels, independent of orientation and scaling, Lucas-Kanade describes a pixel with certain features. An algorithm that can provide adequate features is the Shi-Tomasi algorithm [16]. They propose the tracking of corners to avoid the aperture problem. Shi and Tomasi define a good feature in the same manner as Harris [17] which relies on the matrix of the second order derivatives of the image intensities. By calculating autocorrelation of the second derivative over small windows around each point we get a good description of the window. Shi and Tomasi found that a good corner could be easily described as long as the smaller of the two eigenvalues was greater than a minimum threshold.

Through optical flow, it is also possible to determine motion vectors or feature points that may be used to segregate faster moving features to slower moving features. For this segregation, we use clustering to condense the number of features in one region of the frame. To limit computation load we also added a neighbor count filter, which rejects isolated features. Our approach is inspired by building reciprocal nearest-neighbor chains [18], nicely described for the efficient average-link algorithm in [19]. The essence of this algorithm is as follows; one starts with a random feature and search for the nearest-neighbor. As long as the neighbor is close enough (relative to threshold θ), the neighbor is added to the chain. When the neighbor feature is already member of another chain, the two clusters are merged if they fulfill a *reducibility property* (the clusters are more similar to each other than any other cluster) [20]. When no close neighbor feature can be found a new random feature point is chosen to create a new chain. Note that the original approach [19] traverses through all combinations by sorting the all points from a common origin, while in our case a distance threshold is set. By calculating the average length of the optical flow for each cluster, it is possible to estimate affine relationships between closer and further obstacles, which can be used to evade closer obstacles.

3.3 Proximity estimation

In order to determine proximity of obstacles, calculating scale expansion of obstacles in the scene while moving closer makes it possible to estimate threat of collision. For this, we use a template matching approach based on *normalized*

cross-correlation [21] where we generate the previous image in different scales and then compare to the current image and find the best matching scale. For this, we take the mean location of dense feature groups from the result of tracking step and create a sub-window around that point from the previous frame. We then draw a sample template from the current image with the same location and dimensions and run the template matching algorithm to determine the relative change in scale.

3.4 Integration

To be able to robustly avoid obstacles, several methods (described in the previous subsections) have to be combined to generate control commands for the flying platform based on the images of the monocular camera. An overview is given in Figure 1. The control framework consists of two parts; the hardware side and the computer side.

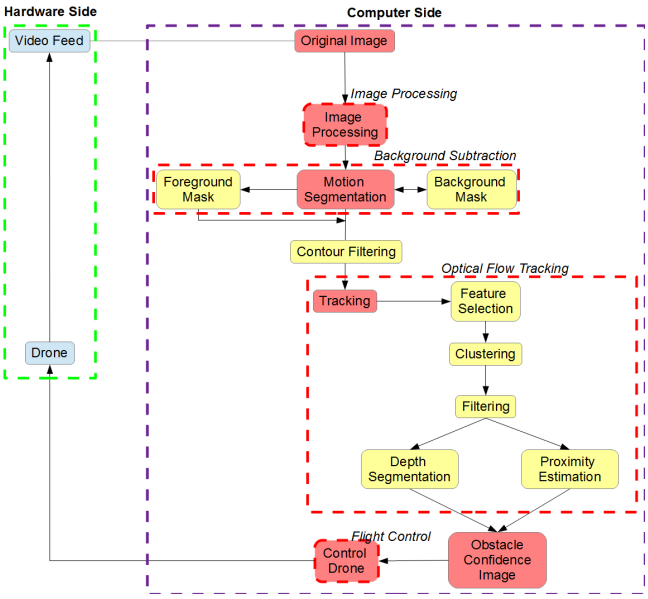

Figure 1: Overview of control pipeline

The final product of the image processing chain is an obstacle confidence image, which is used to steer the robot in a number of orthogonal directions (left, forward, right).

4 Results

We conduct flight experiments in two situations to test performance of our method to highlight success and failure conditions. Flight path illustrations are plotted based on linear directions given by the drone controller to the drone in a sequence during the flight run. Example: [forward, forward, forward, left, left, left, left, forward, left, left, forward] where the drone moves by linear velocity at $\sim 1m/s$ in the given direction.

4.1 Two narrow objects as obstacles

In this experiment, the drone has to avoid two trees at a slight offset from one another where one is behind the other, as illustrated in Figure 2. In this scenario, an early decision to evade the second tree could result in a collision with the first tree. The experiment will be first performed with our complete algorithm, followed by an experiment without foreground, depth and proximity estimations.

Figure 2: Two trees as an obstacle in the flight path

As can be seen from Figure 3 that out of five runs, two runs avoid the trees with a flight path through the gap between the trees, two runs dodge both trees from the left side and one run dodges the first tree from the right side and does not encounter the second tree at all in its path. In four runs the flight path is not straight due to feature clusters detected form shadows on the ground. We also note in two runs (red and blue) there was a risk of the drone colliding sideways into first ree when avoiding the second tree.

Figure 3: The path of the AR.Drone between two obstacles (left), based on the image including depth classification (right)

In Figure 4, we run the same experiment, yet this without filtering the optical flow in the background. Compared to the previous experiment, we see a large number of feature points detected on the trees in the background and not many on the obstacles. Flight results (Figure 4 (right)) do not have any successful results as features were detected everywhere in the scene except for the obstacles. As Tree 1 has more texture than Tree 2, two runs manage to successfully avoid Tree 1 however, the same level of accuracy is not achieved

with Tree 2 and feature points detected on the ground create a bias for the drone controller to fly left when it should be flying right resulting in collision with Tree 2. Three runs out of five do not detect enough features on Tree 1 resulting in collision. While a number of features detected may have detected features on the obstacles, we accept features only above a set score so that we do not have features with low scores. To summarize the impact of filtering of the optical flow; with BackGround/ForeGround classification 631 of 840 features can be found on the trees, while without this classification 159 of 1987 features are found on the trees.

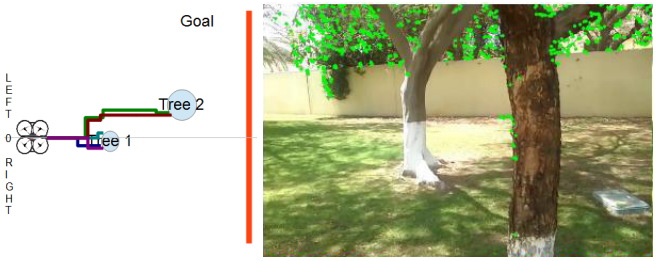

Figure 4: The path of the AR.Drone failing to avoide two obstacles (left) based on the image without ForeGround classification(right)

In Figure 5, we run an experiment using only proximity estimation. While the first tree is avoided well in all five runs, three runs fail due to collision with the second tree. This is due to the trunk of the second tree covers almost one half of the frame resulting in a poor template matching score. In two runs, the template matched with scale factor 1.1 of the previous frame texture and in one run, the frame was too obscured by the tree trunk due to insufficient score in the previous run. This may have been due to the fact that the second tree is wider than the first tree and there was not enough distance to measure the template accurately after avoiding the first tree.

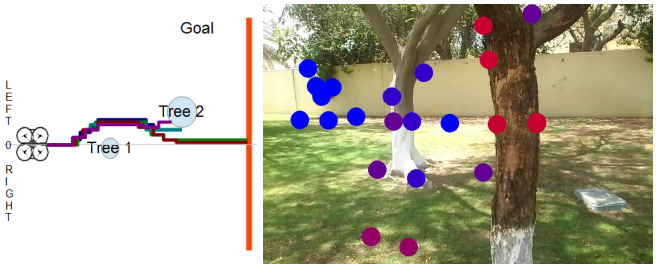

Figure 5: The path of the AR.Drone between two obstacles (left), without depth classification (right)

In Figure 6, we run an experiment using only optical flow without depth or proximity cues. All five flights in this setting are successful however the solution is a bit crude. As there is no depth perception, the process just detects both trees as an obstacle at the same degree and avoids them both immediately once detected. This method however is sensitive to feature patches found on the ground by tree shadows therefore we do not see a perfectly smooth flight once both obstacles are dodged.

Figure 6: The path of the AR.Drone around two obstacles (left), with no depth nor proximity estimation (right)

4.2 Two wide objects as obstacles

In this experiment, we park two cars as wide body obstacles as a series of obstacles for the drone to fly around, as illustrated in Figure 7. This is another challenge; cars have a complete different texture and are so wide that they could fill the complete field-of-view. On the positive side; shadows are less prominent and the obstacles are more equally illuminated.

Figure 7: Two cars as obstacles in the flight path

Again we start the experiment with the full algorithm. As can be seen from Figure 8 there are four successful runs and one run (turquoise) failed. The failed run collided into first car just after taking off. We observe this to happen due to the template matching failing as the drone take off point was too close to the first car. The matching score was low causing the method assumed that the obstacle is far away. For the next four runs, we move the drone starting point a few steps back. Even after this step, the template matching score was not high enough however, the optical flow depth segmentation detected faster shift in feature points on the first car compared to detected features elsewhere in the scene due to the motion parallax property of optical flow. With this experiment, we see that when objects are too close to the drone on take-off,

the template matching algorithm fails as the drone does not observe a high degree of change in scale to satisfy the scale threshold in our method. Two runs (green and purple) successfully avoid first car however there is some delay in going around the right way when the second car is detected. This is due to a greater number of feature points detected on the rear of the car (right half of frame) compared to the front of the car (left half of the car). Once the drone decides to move left, features detected on the left side shift by a larger degree than the right side of the frame resulting in the controller giving the drone direction to move right successfully avoiding the second car.

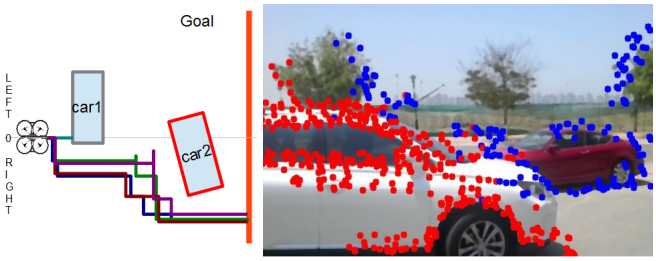

Figure 8: The path of the AR.Drone avoiding two cars (left), , based on the image including depth classification (right)

In Figure 9, we see results of the proposed method without the foreground/background classification along with flight results. In the frames illustrated, we see that there are a lot of features detected on background trees and fence in the background of the scene. While there are a few features detected on the obstacles, most features are detected outside the two cars. Due to this, we see again a strong bias from background objects leading to non-perfect obstacle avoidance runs. Overall, we have two successful runs and three unsuccessful runs. When faced with car 1, two runs (red and greed) detect a lot of features on the tree at the right side of the scene over features detected on the car 1. As not enough features are detected on car 1, the drone controller did not assume there to be an obstacle ahead resulting in collision. One run (purple) avoids car 1 and while crossing pass it, large number of incorrect features in the background and correct features detected on car 2 cause the drone controller to give instructions to move right colliding into car 1 along the drone's side. Two successful runs (blue and turquoise) avoid car 1 and get enough features on car 2 to successfully avoid it as well despite more features detected on other objects in the scene due to the depth and proximity estimations employed in the proposed method. If we analyze the number of correct features for this scenario, 874 of 1211 features are on the cars with background/foreground classification, while only 504 of the 1384 features are correct without this classification.

The experiment using the proximity estimation method without depth information confirms that this is no stable algorithm. As can be seen in Figure 10 all five runs were a failure.

Figure 9: The path of the AR.Drone avoiding two cars (left), based on the processed image (right)

The first car was positioned very close to the starting position of the drone, so the initial template was not very descriptive. As a result the template matching routine always gave poor score values, so the flight controller just gave forward commands as no threats with high confidence were detected.

Figure 10: The path of the AR.Drone between two obstacles, without depth classification

In the experiment in Figure 11, we test the crude version of our algorithm. The runs are not very successful even though the method does detect feature points on the first car. For the three successful runs, the first car is avoided but going in the wrong direction. Even though the first car was close to the drone take off point, and feature points of the second car were detected as well, the scene contained greater number of feature points on the right side of the scene leading to the drone controller give the left command as all points appeared to be at the same level. Once the first car was crossed, the flight path lead to the right direction however avoiding the second car completely as it was no longer in the way. We also observe two paths (green and turquoise) not moving in any set direction at once point as the drone kept shifting left and right when in the center of the first car. This was because there were occasions where feature points on the first car kept going in and out of the scene as it was too close, the drone was stuck shifting left and right.

5 Conclusion

For each experiment carried out in this study, we see a good performance of the proposed method in a variety of situations. During the experiments, we see some significant influence of lighting and texture conditions where performance

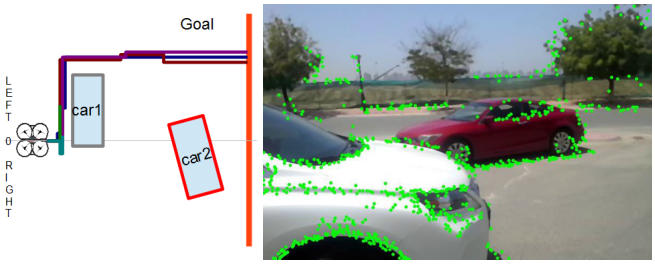

Figure 11: The path of the AR.Drone around two obstacles, with no depth nor proximity estimation

was noted to be better when there was good contrast between the obstacles and the surroundings. For outdoor experiments, we saw a lot of false positive features detected around trees in the background among others which caused a toll on the computation time required for optical flow between each frame pair. Due to which, we adjusted the contour threshold algorithm to have a higher contour threshold so as to eliminate small blobs caused by trees and leaves in the foreground mask from the background/foreground classification step. Feature detection is also significantly affected when there are not enough corner points found in large smooth parts of objects; e.g. car panels. Due to which the method has to rely entirely on corners and edges of the object. We also observe that accuracy of the template matching algorithm depends on regularity of illumination between frames. When there is a sharp change in illumination, the template matching algorithm returns low scores, and optical flow is also affected as feature are not matched. A failed run in the wide obstacle experiment shows us that, while the method performs adequately when the obstacle fits in the frame in its entirety, partially visible obstacles like the first car with low texture for a large part of the body cause an impact on the result. We also learn from comparison with other methods that while optical flow methods work, for obstacles and features moving from side to side, they do not function well when faced with walls or very narrow frontal obstacles. In the case with template matching, we see that it works fairly well when there is adequate distance between obstacles, but do not perform too well in tight navigation scenes. The proposed method having a combination of both methods performs better than them individually but is still subject to scene conditions of illumination and good features to track.

References

[1] Pierre-Jean Bristeau, François Callou, David Vissière, Nicolas Petit, et al. The navigation and control technology inside the ar.drone micro uav. In *18th IFAC World Congress*, pages 1477–1484, 2011.

[2] Tomoyuki Mori and Sebastian Scherer. First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *IEEE International Conference on Robotics and Automation (ICRA'13)*, pages 1750–1757, 2013.

[3] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *IEEE international conference on Robotics and automation (ICRA'11)*, pages 5776–5783, 2011.

[4] Koray Çelik, Soon-Jo Chung, Matthew Clausman, and Arun K Somani. Monocular vision slam for indoor aerial vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pages 1566–1573, 2009.

[5] Jeong-Oog Lee, Keun-Hwan Lee, Sang-Heon Park, Sung-Gyu Im, and Jungkeun Park. Obstacle avoidance for small uavs using monocular vision. *Aircraft Engineering and Aerospace Technology*, 83(6):397–406, 2011.

[6] David N Lee et al. A theory of visual control of braking based on information about time-to-collision. *Perception*, 5(4):437–459, 1976.

[7] Jeongdae Kim and Yongtae Do. Moving obstacle avoidance of a mobile robot using a single camera. *Procedia Engineering*, 41:911–916, 2012.

[8] Abhijit Kundu, CV Jawahar, and K Madhava Krishna. Realtime moving object detection from a freely moving monocular camera. In *IEEE International Conference on Robotics and Biomimetics (ROBIO'10)*, pages 1635–1640, 2010.

[9] Morimichi Nishigaki and Yiannis Aloimonos. Moving obstacle detection using cameras for driver assistance system. In *2010 IEEE Intelligent Vehicles Symposium (IV)*, pages 805–812, 2010.

[10] GCHE De Croon, E De Weerdt, C De Wagter, BDW Remes, and R Ruijsink. The appearance variation cue for obstacle avoidance. *IEEE Transactions on Robotics*, 28(2):529–534, 2012.

[11] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-Based Surveillance Systems*, pages 135–144. Springer, 2002.

[12] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[13] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.

[14] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[15] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker - description of the algorithm. Technical report, Intel Corporation, 2001.

[16] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600, 1994.

[17] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[18] J-P Benzecri. Construction d'une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques. *Cahiers de l'analyse des données*, 7(2):209–218, 1982. Rappel: *Cahiers de l'analyse des données*, 22(2):191-198, 1997.

[19] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77(1-3):259–289, 2008.

[20] Michel Bruynooghe. Méthodes nouvelles en classification automatique des données taxonomiques nombreuses. *Statistique et Analyse des données*, 3:24–42, 1977.

[21] JP Lewis. Fast template matching. In *Vision interface*, volume 10, pages 120–123, 1995. The report "Fast normalized cross-correlation" expands the original paper.