First Results in Detecting and Avoiding Frontal Obstacles from a Monocular Camera for Micro Unmanned Aerial Vehicles

Tomoyuki Mori and Sebastian Scherer

Abstract— **Obstacle avoidance is desirable for lightweight micro aerial vehicles and is a challenging problem since the payload constraints only permit monocular cameras and obstacles cannot be directly observed. Depth can however be inferred based on various cues in the image. Prior work has examined optical flow, and perspective cues, however these methods cannot handle frontal obstacles well. In this paper we examine the problem of detecting obstacles right in front of the vehicle. We developed a method to detect relative size changes of image patches that is able to detect size changes in the absence of optical flow. The method uses SURF feature matches in combination with template matching to compare relative obstacle sizes with different image spacing. We present results from our algorithm in autonomous flight tests on a small quadrotor. We are able to detect obstacles with a frame-to-frame enlargement of 120% with a high confidence and confirmed our algorithm in 20 successful flight experiments. In future work, we will improve the control algorithms to avoid more complicated obstacle configurations.**

I. Introduction

The ability to detect and avoid obstacles of birds flying in a forest is fascinating and has been subject of a lot of research. However, there are also many applications of small safe aerial vehicles for search and rescue, mapping, and information gathering.

Advances in battery technology, computing, and mechanical components have enabled small flying flapping wing vehicles[1]. These vehicles have the advantage of being able to fly close to obstacles and being nimble. Their low weight makes them more robust and less dangerous. However, the limited payload only permits lightweight sensors such as a single camera to be used for control and obstacle detection. Since camera based methods rely on external light sources such as the sun to illuminate the scene and rely on texture features to perceive the environment and there will be situations where obstacles cannot be detected. However, the small size of the vehicle permits a less reliable obstacle detection method since collisions can be considered in the vehicle design space.

Prior approaches have focused on bio-mimetic approaches for obstacle avoidance. However these approaches have mainly exploited optical flow or stereo disparity as depth cues to detect oncoming obstacles. Biological flying creatures on the other hand also use many other monocular cues to detect oncoming collisions as shown in Table I. A successful system will have to exploit all available cues to detect obstacles. Different cues, however, are useful at different times. For example perspective cues such as vanishing lines can be very useful in man-made environments while they do not work well in natural environments because of a lack of straight lines. Optical flow on the other hand fails in man-made environments, because large regions of homogeneous texture (e.g. painted wall) do not have sufficient information to enable flow calculations.

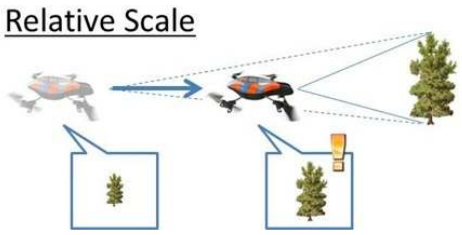


Fig. 1: An example frontal obstacle. The optical flow response of this obstacle is close to zero. However, our approach is able to detect and avoid this type of obstacle.

In particular, detecting and avoiding frontal collision in monocular imagery is challenging since there is no motion parallax and optical flow is close to zero for low resolution cameras. We develop and test a relative size detector that is particularly useful to detect and avoid frontal collisions (Fig. 1). This feature descriptor detects relative size changes of features and is executed in real-time to avoid collisions. In addition we also show a guidance algorithm that will permit flight through forests with small aerial vehicles. We convey results from 20 obstacle avoidance experiments.

First we put our work in context with prior research in Sec. 2, then describe our approach in Sec. 3, and analyze our results in Sec. 4.

II. Related Work

There has been a significant amount of work in making unmanned aerial vehicles aware of their environment and behave in an intelligent manner. A survey of current approaches can be found in [16]. There are many potential

Tomoyuki Mori is with Mitsubishi Heavy Industries, Ltd., Japan. Sebastian Scherer is with the Robotics Institute, Carnegie Mellon University, Pittsburgh. email: tomoyuki_mori@mhi.co.jp, basti@cmu.edu

	Method	Availability	Related Work
Motion Parallax	Optical Flow/SfM	Computation is large. Cannot detect obstacle straight ahead	[1], [2], [3], [4], [5], [6], [7], [8], [9]
Monocular Cues	Perspective	Only useful in structured environments	[10], [11]
	Relative Size	**Available for straight-on collisions**	[12], [13], [14], [15]
	Known Object Size	Features are useful for particular objects.	
	Texture Gradient		
	Occlusion/Haze/ Shadow		
	Depth from Focus	Not typically available for small aperture cameras	
Stereo vision	Stereoscopic parallax	Need a sufficient baseline and resolution	
	Convergence	Only in humans	

TABLE I: Depth cues in a biological vision system. There are many potential depth cues used by biological systems. We focus on the highlighted relative size cue.

sensor modalities and in particular lidar based approaches have been shown to be able to sense obstacles reliably in [17], [18]. However, for size, weight, and power (SWAP) constraint vehicles cameras are the lightest sensor and given computing advances and the type of computation can be potentially very light. In the following section, we will talk about depth detection using vision systems.

There are three principal ways that depth can be sensed in a biological vision system: Motion parallax, monocular cues, and stereo vision.

Motion parallax is utilized in optical flow methods. For example, Beyeler et al. [2] developed autonomous extremely lightweight indoor fliers that avoid obstacles and stay away from obstacles using a small 1-D camera. Oh et al. [3] and Green et al.[4] developed autonomous optical flow based obstacle avoidance similar in idea to the design by Zufferey et al. A larger fixed-wing aerial vehicle was used by Merrel et al. [5] to reactively avoid obstacles and follow a canyon with radar and optical flow. A fundamental limitation of optical flow methods is that from frame to frame the flow is proportional to the angle to the frontal direction. This makes these methods useful for wall following or corridor centering applications but difficult to use for frontal obstacle avoidance. Therefore, Hrabar et al. [19] also performed extensive experiments using optical flow and stereo cues. Since geometrically there will be no flow ahead of the robot stereo disparity is used in their work to detect frontal obstacles.

Monocular cues have been used to avoid collisions. For example, hallway centering was tested with perspective clues in [10], [11]. Other monocular approaches have tried to directly map appearance to an obstacle distance such as in [20], [21]. These methods were designed to work well in corridor environments, however the perspective cues exploited are many times not available in natural environments.

Even though many monocular approaches cannot measure distance to obstacles directly it is still possible to avoid obstacles since the optical flow has a relationship to the time to collision [22].

Our approach focuses on one of the monocular cues, a relative size cue to detect frontal collisions. When the size of the object in front of the UAV is expanding, it means that the object is approaching the UAV. The response is proportional to the time to collision by measuring the expansion of an obstacle. Also this approach can compensate for a lack of parallax in optical flow that makes it difficult to use for frontal obstacle avoidance.

In addition, Roberts et al. [23] developed an algorithm that detects trees with line segmentation and optical flow. However their approach is only for forests and they have not flown autonomously in the forest. We pursue the general approach to detect frontal objects, and fly autonomously.

III. Approach

In this paper we focus on a size expansion cue since applying other methods of depth detection such as perspective are not applicable in natural environments. Other methods would require a known object shape or size, and texture gradients are good at detecting the ground surface or an object which has a large surface. These methods are difficult to apply to detect natural objects like trees.

A size expansion cue is useful to detect obstacles in front of the vehicle to avoid a collision. From the biomimetic standpoint, a size expansion cue is known to alert of approaching objects. Gibson [24] was able to show that looming could simulate the sense of approach directly. When the objects visual size expands, a human will detect the approaching object, not the size expansion.

Recently, there have been some attempts to apply size expansion or similar cues for obstacle avoidance in UAVs. Croon et al. [12] used the appearance variation cue to detect approaching frontal obstacles and they successfully avoided a wall. However this approach depends on the texture in the environment and doesn't use the size expansion directly. Byrne et al. [13] executed the expansion segmentation with inertial aid. However their demonstration is limited to simulation.

We have developed a direct "relative size" detector without inertial aid and an obstacle avoidance algorithm that returns the apparent size of obstacles as a function of time to collision. For example, when the distance from an obstacle reduces to 2m from 3m, the apparent size in the image becomes 1.5 times larger (See Figure 2). Since the frame rate of the images is known, one can calculate the time to collision.

A. Frontal Obstacle Detection

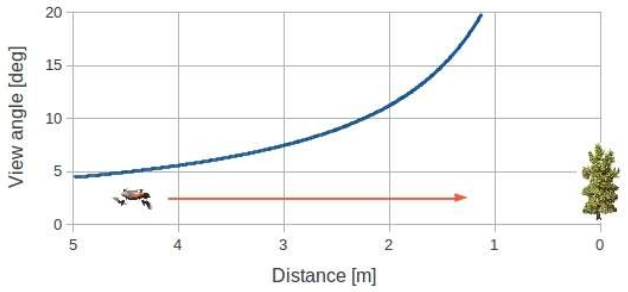


Fig. 2: Distance vs. scale. The obstacle size is 0.4m in this graph. As the distance to the obstacle decrease the viewing angle that is occupied in the field of view increases.

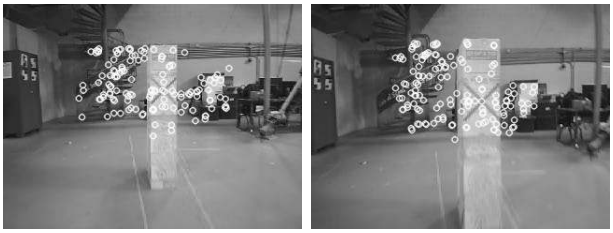
A frontal obstacle detector needs to detect the relative size change of obstacles in front, in consecutive images. One approach would be to segment the image and look at the size expansion in the segments. However, since we would like a method that is feasible in real-time we consider a different approach.

Recently, scale invariant features like SIFT/SURF features [25] have shown their usefulness in computer vision because they are relatively fast to compute and are invariant to large scale changes. The features can be matched across multiple images and also work for larger sections of skipped features. Chavez and Gustafson[14] show how to detect looming in consecutive images with SIFT features matching. However, the results where not real-time and not tested on an actual robot that is avoiding obstacles. Also, Alenya et al [9] present a comparison of three methods that measure time to contact that include a looming detection method with scale invariant features. We develop a similar approach that tries to be more accurate in the scale detection and test it on a small UAV that actually avoids obstacles. For real-time and reliable obstacle avoidance, we choose SURF features and template matching.

Our approach uses SURF features to detect the initial locations of potential locations that are good to detect relative size changes. SURF key points match even if their size has changed. Our algorithm matches key points in consecutive images and compares the size of the region around the key point, to recognize if the camera is approaching an obstacle. An advantage of this approach is that potentially any SURF matched feature could be used to avoid obstacles and SURF is quicker to calculate than SIFT.

After creating the SURF matches we discard any matches that did not get bigger. The next step is to match the template from one frame to the next. Template matching uses a region around the image defined by the SURF feature scale to calculate how well the features match. If an obstacle has gotten too close it is considered an obstacle and will be used for avoidance. Algorithm 1 shows our algorithm flow.

We used the OpenCV SURF algorithm to generate the key points and matches at line 1 and line 2 in Algorithm



(a) Previous image (b) Current image

Fig. 3: SURF key points in consecutive images. (o) are detected features in the images. The images are separate by a skip time of n_{skip}.

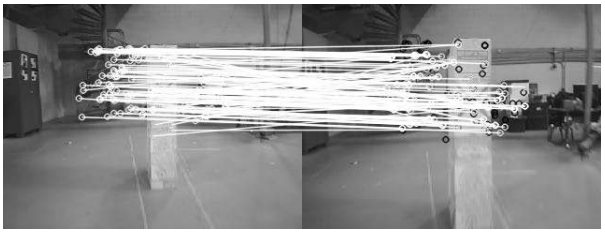


Fig. 4: The unfiltered SURF matches between the input images.

1(Also, Fig. 3, Fig. 4). Each match gives us the center of a point that has potentially grown in size. There is a large set of matched key points with many wrong matches. We calculate the match accuracy from the descriptors and discard inaccurate key points in line 3 (Fig. 5). Furthermore, we only need key points that become larger in the current image and therefore discard key points whose size is small or the same in line 4 (Fig. 6).

An accurate relative size is calculated using template matching on each key point in line 5 (Also, see Algorithm 2). We calculate a key point's scale ratio in the current image by changing the previous key points scale and applying a template matching algorithm to confirm the scale ratio. Figure 7 shows a sample template matching result. In this graph, the template matching result is best when the scale is 1.4. This means that the SURF feature became 1.4 times larger. We have found that the reported scale of the SURF features is not accurate enough to give a good time to collision estimate. If the correlation distance of the template matching does not change between the same scale (1.0) and

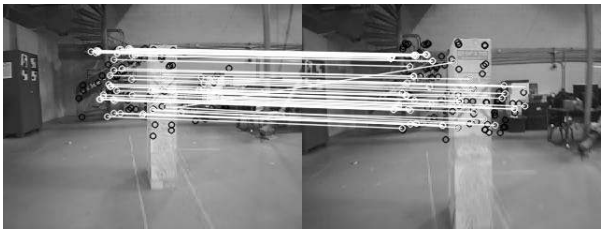


Fig. 5: The filtered matched features whose feature distance is small(good).

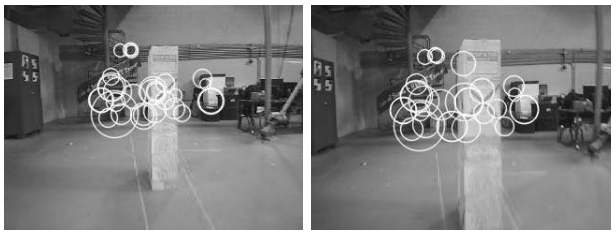
Algorithm 1 The scale expansion detector algorithm. This algorithm matches, filters and calculates the expansion of relevant SURF features in consecutive images.

Procedure

1. Generate SURF Key Points (Figure3)
   Output: $(KP_i^j (j = 0 \rightarrow n_{keypoint}))$
2. Match SURF Key Points in consecutive images (Figure 4)
   Output: $(M_j(KP_i^{index(j)}, KP_{i-n_{skip}}^{index(j)})\ (j = 0 \rightarrow n_{match})\ )$
3. Discard miss matching features (Figure 5)
   Equation: $(M_j.distance > 0.25\ (j = 0 \rightarrow n_{match})\ )$
4. Discard Key Points which become small or the same (Figure 6)
   Equation: $(M_j.KP_i^{index(j)}.size > M_j.KP_{i-n_{skip}}^{index(j)}.size\ (j = 0 \rightarrow n_{match}$,except discarded at line 3))
5. Confirm scale with template matching (Figure 7)
   See Algorithm 2
6. ϵ = average($KP_i^{index(j)}$) (Figure 8)
   $(j = 0 \rightarrow n_{match}$,only recognized obstacle at line 5)

Parameters:

KP_i^j	: Key points, i=frame no., j=number of key points
$n_{keypoint}$	: Total number of key points
$M_j(KP, KP)$	: Pair of matched key points
index(j)	: Number of key points at j_{th} matched
n_{skip}	: Frame number of skip when key points are matched
n_{match}	: Number of matched key points
$M.distance$	: Distance of matched key points calculated by SURF descriptors
$KP.size$	: Size of key points
ϵ	: Position of the obstacle



(a) Previous image (b) Current image

Fig. 6: All detected expanding features(o). These features show all the areas in the image where a significant expansion is detected.

the best matched scale ($Scale_{min}$), we discard this key point because the key point is likely to have homogeneous texture.

With this template matching algorithm, we can detect the ratio of SURF key point expansion. As shown in Fig. 2, the distance from the UAV to the expanding SURF key point is close and the time to collision is small. Finally, we can get the group of expanding SURF key points. Figure 8 shows a sample result. Red circles refer to expanding SURF points.

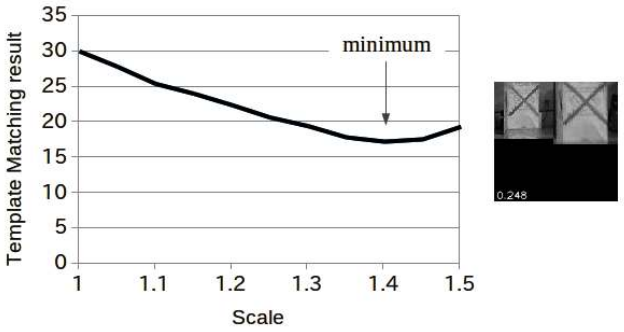


Fig. 7: A sample template matching result for the image shown on the right. The x-axis is the scale (as in algorithm 2). The result is at a minimum at scale 1.4 . This SURF feature became 1.4 times from previous image to current image.

B. Reactive Obstacle Avoidance for Forest Flight

To evaluate our scale expansion detector, we applied it to the forest flight challenge. With our scale expansion detector, the UAV can avoid frontal tree obstacles and fly goal directed in forests. Some prior work has considered flight in forests. In Karaman and Frazzoli [26] a position distribution is

Algorithm 2 The template matching algorithm. This algorithm looks for a time to collision that is relevant for our vehicle to detect.

For($j = 0 \rightarrow n_{match}$,except discarded)

5.1 Create template image from $image_{i-n_{skip}}$(previous image)

$Template1$:image around $M_j.KP_i^{idx(j)}$, $Template1.size = M_j.KP_{i-n_{skip}}^{idx(j)}.size * 1.2/9 * 20$

For($Scale = 1 \rightarrow 1.5$)

5.2 Expand $Template1_j$ as scale

5.3 Create template image from $image_i$(current image)

$Template2$:image around $M_j.KP_i^{idx(j)}$, $Template2.size = M_j.KP_{i-n_{skip}}^{idx(j)}.size * 1.2/9 * 20 * Scale$

5.4 $TM_{scale} = Matching(Template1, Template2)/Scale^2$

5.5 $If(TM_{scale} < TM_{min})$

$TM_{min} = TM_{scale}$, $Scale_{min} = Scale$

5.6 Next Scale

5.7 $If(Scale_{min} > 1.2) \bigwedge (TM_{min} < 0.8TM_{1.0}))$

$KP_i^{idx(j)} \rightarrow Obstacle$

5.8 Next j

Parameters

$image_i$: Image at i frame

$Template1$: Template for template matching 1

$Template.size$: Size of template(length of square side)

$Scale$: Expansion scale of Template1

TM_{scale} : Result of template matching at scale

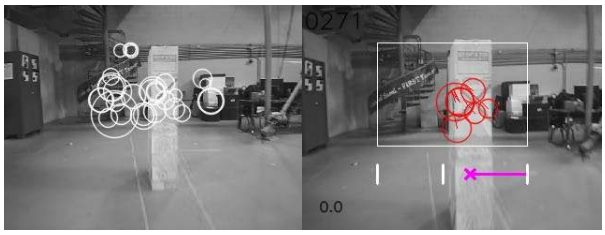
$Matching(image1, image2) : Function$

Calculate a difference between image1 and image2 for each pixel.

Output is a image whose size is same as image1 and image2

TM_{min} : Minimum result of template matching

$Scale_{min}$: Scale at template matching result is minimum.



(a) Previous image (b) Current image.

Fig. 8: Sample image of the expanding selected key points and command. The red circles represent expanding key points. The purple line is the command given to the vehicle. One can see that many other key points in the field of view were rejected.

used to model the tree locations in a forest and we use a similar approach to model the distribution of trees in our simulation. According to the authors for this kind of forest only the closest obstacle is really relevant. This motivated our approach for a simple reactive obstacle avoidance law.

Frew et al.[27] proposed a trajectory generator for small UAV to fly in forests with limited field of view and distance detection. The authors demonstrated their trajectory generator in forests. Since, we assume a small vehicle that has to react quickly, we don't use a deliberate path planning algorithm, and instead use a reactive guidance law to avoid obstacles and fly to the goal. This approach is a "sensor based navigation."

The strategy is that the UAV will avoid the frontal obstacle when it detects the approaching obstacle with the scale expansion detector as proposed in Sec 3.1. Gibson [24] determined that animals detect "visual collision" with looming and do not detect the distance. Likewise, we do not detect the distance from obstacles. The vehicle simply avoids the frontal obstacle when it detects that the time to collision of the approaching obstacles is too small.

The information the aerial vehicle has to know about the environment is only the bearing of the closest tree. Also if the aerial vehicle has to go to a goal, it must know it's own position and goal position.

For simulation, we apply the control method of the actual quadrotor we used in our experiment (Sec. 4.2). Bristeau et al. [28] explain the control technology of this commercial UAV. They estimate the vehicle velocity with a down-looking camera for control since the MEMS inertial measurement unit cannot be integrated for velocity estimates. Our control commands to the UAV are velocity commands in simulation and in the experiments. The approach is as follows:

- The vehicle will fly sideways when the obstacle is found in the field of view

Algorithm 3 The obstacle avoidance control algorithm. This algorithm will fly towards the goal if possible and command a bang-bang avoidance maneuver if necessary to avoid collision with an obstacle.

Yaw control:

$\psi_{com} = -goal_{bearing} \cdot K_\psi$

X control:

$v_{x,com} = 1.2\frac{m}{s}$ (constant)

$a_{x,com} = (v_{x,com} - v_x)K_v$

Y control:

$v_{y,com} = 2\frac{m}{s} \cdot sign(obstacle_{bearing})$ (if there is an obstacle, command for 0.4 sec)

$a_{y,com} = (v_{y,com} - v_y)K_v$

Parameters

v_x, v_y :UAV Velocity (Body coordinate)

a_x, a_y :UAV Acceleration (Body coordinate)

K_ψ, K_v :feedback gain

- Otherwise it always controls its yaw angle to achieve the goal bearing

With this simple method, our vehicle can fly through the forest. There is a large number of parameters that determine if that flight is feasible: response time, acceleration limits, flying speed, obstacle sensing distance, field of view, trunk size, minimum distance between trees. In the simulation we present in Fig. 9, the UAV response time, flying speed are almost same as of the AR.Drone that we used in our experiments.

IV. Experiments

A. Setup

We evaluated our algorithms on a small commercial quadrotor vehicle (Parrot AR.Drone) that is robust enough to tolerate sensor failure. The vehicle is equipped with two cameras. The field of view of the forward camera is 92 deg, 320X240 resolution at a frequency of 10Hz. We use gray scale images for our algorithm however the images produced are 8-bit RGB. All our algorithms process the images on the ground on a laptop which is a quad core Intel i5-2410M@2.3GHz running Linux. In addition a sonar height sensor, inertial measurement unit (gyros, and acceleration sensors), and downward looking camera are available and used to control the velocity and height of the AR.Drone. We did not add any additional sensors on the AR.Drone.

B. Obstacle Avoidance Experiments

We let the quadrotor fly toward an obstacle to verify if our algorithm can detect and avoid the obstacles. Using the velocity and attitude estimates from the AR.Drone we modified the AR.Drone to fly autonomously. The algorithm for flying autonomously is the same as Algorithm 3. It flies forward from the start point to the goal point. If it detects an obstacle during the flight, it will avoid the obstacle, and continue to fly toward the goal. Figure 11 shows the experiment course.

Fig. 10: Our test vehicle (Parrot AR.Drone) in flight while autonomously avoiding obstacles.

	TOTAL	SUCCESS	FAILURE	RATIO
Run	23	20	3	87%
Objects	107	104	3	97%

TABLE II: Result of a total of 23 runs with 107 avoided obstacles. Of these runs 3 failed because the quadrotor reacted too late to the detected obstacles.

Table II shows our experimental results. We have flown the vehicle 23 times through this experiment course. There are 3 failure cases where the AR.Drone was not able to avoid the obstacles because of a slow response time. In terms of the number of obstacles avoided, the success ratio is 97%. Figure 12 shows the result of many obstacle avoidance flights. These passes are the result of the combination of the scale expansion detector and the AR.Drone's reactive behavior. Also Figure 13 shows the v_x and v_y data.

Figure 14 are images from the AR.Drone when it avoided an obstacle. The white rectangle is the gate where obstacles are detected. Our algorithm discriminates expanding key points on the obstacle from non-expanding key points in a complicated background with a monocular camera. We do not need any prior learning to detect the approaching obstacles. It also works with a low resolution camera and a regular laptop PC in real-time. However, the approach needs texture on the obstacles to detect SURF key points.

V. Conclusions & Future Work

We developed the scale expansion detector to detect the approaching object with monocular vision. We use SURF features to extract the expanding key point and template matching to verify the ratio of expansion. Our scale expansion detector can detect frontal objects that are difficult to detect with optical flow. It works in real time with a low resolution camera on a commercial laptop. In our experiments the algorithm can discriminate close from far

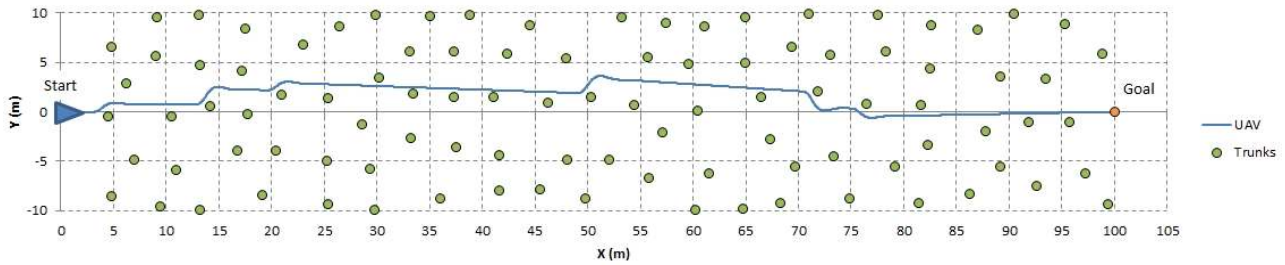


Fig. 9: A simulated-flight result in forests. Ideally, the quadrotor can detect the expanding obstacle from about 1.6m. In algorithm 2 line 5.7, the vehicle can extract the SURF key point whose expansion rate is over 1.2, the speed is ~ 1m/sec, image frequency is 6Hz. We skip $n_{skip} = 1$ image to compare scale.

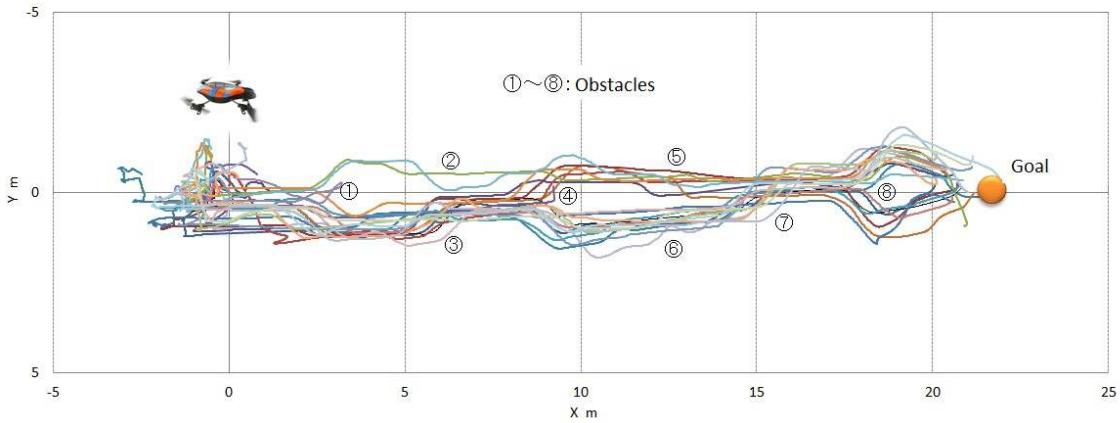


Fig. 12: A set of 23 avoidance runs. To estimate the state to generate this paths we integrate the reported visual velocity and gyro heading. We had to adjust the data to coincide with the obstacle and goal location.

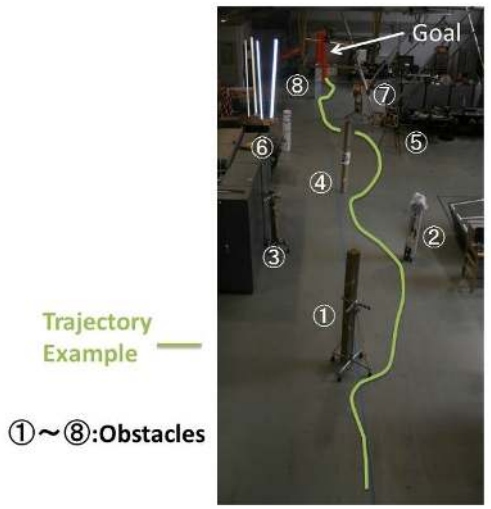


Fig. 11: The experimental setup for the obstacle avoidance experiments and a depiction of the typical behavior of the vehicle. There are eight obstacles in the straight line path of the vehicle.

features in a cluttered background. It is not necessary to learn any prior information about obstacles.

With our scale expansion detector and sensor based navigation, the quadrotor can avoid slim obstacles like trees. We flew the quadrotor 23 times on a course with 8 obstacles and the vehicle reached the goal 20 times. Overall the vehicle avoided a total of 104 obstacles with 3 failed avoidance maneuvers. The reason for these failures was the quadrotor's slow response time that could be reduced with a better vehicle platform.

Our scale expansion detector works for frontal obstacles. However, obstacles have to have sufficient texture to make SURF key points. Trees have a slight texture on their trunks so it may be possible to extract SURF key points from tree trunks if we use a higher resolution camera. In future work, we intend to improve performance with a good camera system. Since a higher resolution image means more CPU consumption, on-board processing may be more challenging. The approach however, is trivially parallelizable and could be implemented efficiently on a signal processor and FPGA combination.

Furthermore, we will combine multiple algorithms such as optical flow and perspective cues that have already shown their effectiveness in textured natural environments and homogeneous urban environments(corridors). These cues can

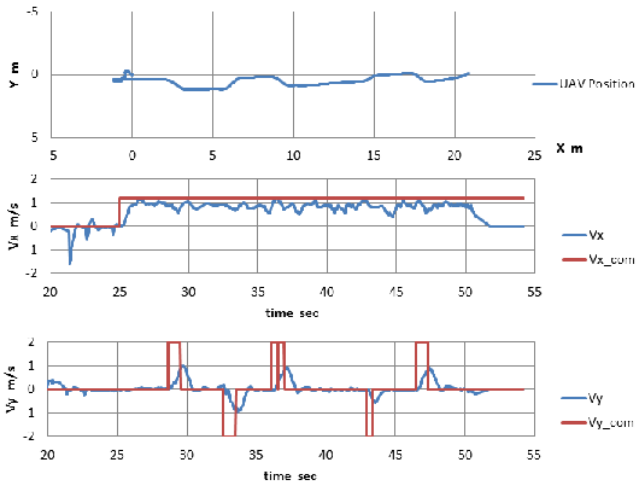


Fig. 13: An example obstacle avoidance maneuver. An obstacle avoidance command ($v_{y,com}$) was generated 5 times.

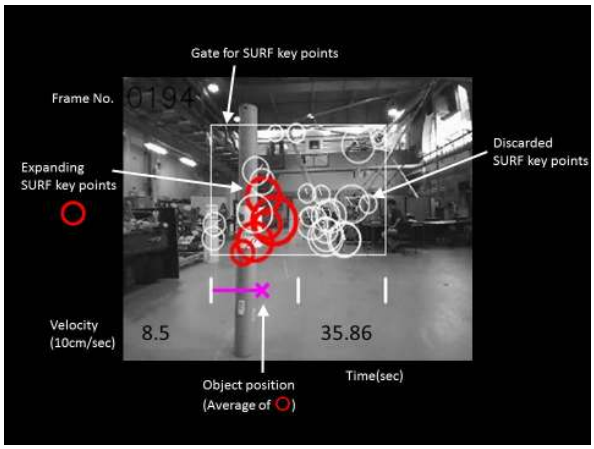


Fig. 14: A detected obstacle during flight. All the red circles are detected expanding feature points in the region of interest (white box). The purple line shows the side the obstacle is detected on. White circles depict rejected feature points.

compensate for the shortcomings of each other and combined will mimic a biological vision system that enables collision-free operation of flying micro aerial vehicles.

REFERENCES

[1] G. C. H. E. de Croon, K. M. E. de Clercq, R. Ruijsink, B. Remes, and C. de Wagter, "Design, aerodynamics, and vision-based control of the DelFly," *International Journal of Micro Air Vehicles*, 2009.

[2] A. Beyeler, J. Zufferey, and D. Floreano, "3D Vision-based Navigation for Indoor Microflyers," *Proceedings IEEE International Conference on Robotics and Automation*, 2007.

[3] P. Oh, W. Green, and G. Barrows, "Neural nets and optic flow for autonomous micro-air-vehicle navigation," *Proc. Int. Mech. Eng. Congress and Exposition*, 2004.

[4] W. Green and P. Oh, "Optic-Flow-Based Collision Avoidance," *Robotics & Automation Magazine, IEEE*, 2008.

[5] P. C. Merrell, D.-J. Lee, and R. W. Beard, "Obstacle Avoidance for Unmanned Air Vehicles Using Optical Flow Probability Distributions," *Mobile Robots XVII*, 2004.

[6] J.-C. Zufferey and D. Floreano, "Fly-inspired visual steering of an ultralight indoor aircraft," *Robotics, IEEE Transactions on*, 2006.

[7] L. Muratet, S. Doncieux, and J. Meyer, "A biomimetic reactive navigation system using the optical flow for a rotary-wing UAV in urban environment," in *Proceedings of ISR2004*, 2004.

[8] S. Hrabar, G. Sukhatme, P. Corke, K. Usher, and J. Roberts, "Combined optic-flow and stereo-based navigation of urban canyons for a UAV," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[9] G. Alenya, A. Negre, and J. Crowley, "A comparison of three methods for measure of Time to Contact," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009.

[10] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011.

[11] K. Celik, S.-J. Chung, M. Clausman, and A. Somani, "Monocular vision SLAM for indoor aerial vehicles," *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009.

[12] G. C. H. E. de Croon, E. de Weerdt, C. de Wagter, B. D. W. Remes, and R. Ruijsink, "The Appearance Variation Cue for Obstacle Avoidance," *IEEE TRANSACTIONS ON ROBOTICS, VOL. 28, NO. 2, APRIL 2012*, 2012.

[13] J. Byrne and C. J. Taylor, "Expansion Segmentation for Visual Collision Detection and Estimation," *2009 IEEE International Conference on Robotics and Automation*, 2009.

[14] A. Chavez and D. Gustafson, "Vision-based obstacle avoidance using SIFT features," *Advances in Visual Computing*, pp. 550–557, 2009.

[15] A. Negre, C. Braillon, J. Crowley, and C. Laugier, "Real-time time-to-collision from variation of intrinsic scale," *Experimental Robotics*, 2008.

[16] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, 2012.

[17] M. Whalley, M. Takahashi, G. Schulein, and C. Goerzen, "Field-Testing of a Helicopter UAV Obstacle Field Navigation and Landing System," in *Proceedings of the 65th Annual forum of the American Helicopter Society (AHS)*, 2009.

[18] S. Scherer, S. Singh, L. Chamberlain, and M. Elgersma, "Flying Fast and Low Among Obstacles: Methodology and Experiments," *The International Journal of Robotics Research*, 2008.

[19] S. Hrabar and S. Gaurav, "Vision-Based Navigation through Urban Canyons," *Journal of Field Robotics*, 2009.

[20] J.-O. Lee, K.-H. Lee, S.-H. Park, S.-G. Im, and J. Park, "Obstacle avoidance for small UAVs using monocular vision," *Aircraft Engineering and Aerospace Technology*, 2011.

[21] J. Michels, A. Saxena, and A. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," *ICML '05: Proceedings of the 22nd international conference on Machine learning*, 2005.

[22] D. N. Lee, "A theory of visual control of braking based on information about time-to-collision," *Perception*, 1976.

[23] R. Roberts, D.-N. Ta, J. Straub, K. Ok, and F. Dellaert, "Saliency Detection and Model-based Tracking: a Two Part Vision System for Small Robot Navigation in Forested Environments," *Unmanned Systems Technology XIV - SPIE Defense; Security; and Sensing*, 2012.

[24] J. J. Gibson, "The Ecological Approach To Visual Perception," 1979.

[25] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, 2008.

[26] Karaman and Frazzoli, "High-speed flight through an ergodic forest," in *IEEE Conference on Robotics and Automation (submitted)*, 2012.

[27] E. W. Frew, J. Langelaan, and S. Joo, "Adaptive Receding Horizon Control for Vision-Based Navigation of Small Unmanned Aircraft," *American Control Conference, 2006*, 2006.

[28] P.-J. Bristeau, F. Callou, D. VissiÃšre, and N. Petit, "The Navigation and Control technology inside the AR.Drone micro UAV," *18th International Federation of Automatic Control(IFAC) World Congress*, 2011.