

Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues

Cooper Bills, Joyce Chen, and Ashutosh Saxena

Abstract—We consider the problem of autonomously flying Miniature Aerial Vehicles (MAVs) in indoor environments such as home and office buildings. The primary long range sensor in these MAVs is a miniature camera. While previous approaches first try to build a 3D model in order to do planning and control, our method neither attempts to build nor requires a 3D model. Instead, our method first classifies the type of indoor environment the MAV is in, and then uses vision algorithms based on perspective cues to estimate the desired direction to fly. We test our method on two MAV platforms: a co-axial miniature helicopter and a toy quadrotor. Our experiments show that our vision algorithms are quite reliable, and they enable our MAVs to fly in a variety of corridors and staircases.

I. INTRODUCTION

We consider flying a Miniature Aerial Vehicle (MAV) such as a quadrotor equipped with a camera (Figure 1) in indoor environments such as corridors and stairs (Figure 2).

Miniature aerial vehicles (MAVs) that can fly autonomously in unstructured indoor GPS-denied environments such as homes and offices would be useful for exploration, rescue, surveillance and entertainment. These aerial vehicles need to be small and lightweight, so a passive, low-power long-range sensor is more suitable than power-hungry sensors such as LIDAR [1]. Our solution is to use a small camera as the primary long-range sensor.

However, the 2D visual information from a camera is difficult to use for navigation in a 3D world. Some algorithms (such as visual SLAM [2]) attempt to infer 3D structure from multiple images. This is challenging because reliable odometry is not available for aerial robots and building a 3D model of the environment takes considerable computation power [3]. Furthermore, the constructed 3D model often has holes for textureless surfaces such as walls. In our work, we instead rely on perspective cues from a single image, which require very little computational power and can be used by a controller directly without building a 3D model.

In the past, vision-based algorithms have been shown to fly in simple environments such as corridors [4]. The ability to travel between different floors is a big advantage for aerial robots over ground-based robots. To the best of our knowledge, our work is the first one that considers flying in previously unseen staircase environments using vision.

We can categorize almost all indoor environments as belonging to one of the following categories: long-narrow spaces (e.g. corridors), areas with non-uniform height (e.g. stairs), open spaces (e.g. atriums), small enclosed spaces (e.g.



Fig. 1. Our indoor quadrotor Parrot AR Drone.

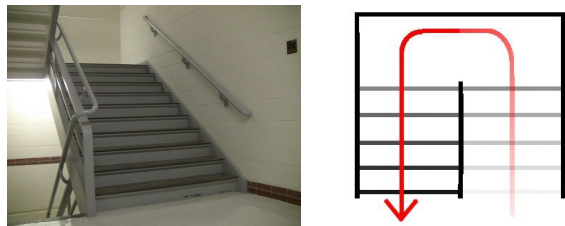


Fig. 2. An example of an environment where we want to fly the MAV in. (Left) Picture of a staircase. (Right) Overhead map of the staircase. Note narrow spaces, unknown terrain, and turns at the end of the staircase.

offices). For each of these environments, there are certain perspective cues that give us information about the 3D structure, and hence depths in different directions. Previously, such single image cues [5] have been successfully used in obstacle avoidance for ground robots [6]. In our work, we first classify the type of environment the MAV is in, and for each case we compute the perspective cues. These cues are then used to navigate the MAV within the environment.

We test our approach in several buildings, including areas never seen before by our algorithms. The testing environments included several corridors and staircases, which often involved turns; for example, a turn may connect two corridors or two flights of stairs.

II. RELATED WORK

Previous work on aerial robots can be arranged as follows: non-vision vs. vision sensors, with a focus on stabilization or navigation, in both outdoor and indoor environments. In indoor environments, the perception and navigation problem is more important because they are GPS denied and more constrained; while in outdoor environments, the focus has been on designing control algorithms [7], [8], [9], [10].

Non-vision sensors: There is extensive work on autonomous quad-rotor navigation using active sensors such as laser range

scanners, sonar, and infra-red. Roberts et al. [11] used infra-red and ultrasonic sensors to fly a quad-rotor indoor in a large ($7 \times 6\text{m}$) room, but it cannot do long-range sensing. Achteik et al. [1] used a laser rangefinder and a stereo camera for quadrotor navigation, with the eventual goal of combining the two complementary sensors. Custom-built quadrotors are used in order to support heavy sensors or additional battery weight necessary to support active, power-hungry sensors; however, most miniature MAVs can carry only a light-weight and low-power sensor such as a camera.

Vision sensors: Recently, there is an interest in building MAVs that use vision to fly in common indoor environments, particularly because vision offers long-range sensing with low power and less weight, allowing smaller aerial robots to be built. Nicoud et al. [12] discussed the tradeoffs of designing indoor helicopters while Schaferoth et al. [13] designed various test benches for micro helicopters and designed a dual-rotor single-axis helicopter with an omnidirectional camera. Mejias et al. [14] used vision to land a helicopter while avoiding power lines. Zingg et al. [4] presented optical flow based algorithms for navigating an MAV in corridors.

Stabilization and Control: Research that used purely vision-based algorithms for stabilization and pose estimation of quad-rotors include using specialized cameras that can be programmed to refocus at certain depths or heavier stereo cameras. E.g., Moore et al. [15] used stereo cameras for autonomous flight. Johnson [16] used vision to make a quad-rotor hover stably indoors. Among the most recent work on UAV flight, work involving vision-based stabilization and pose estimation of the aerial vehicle include [17], [18], [19], but these works consider only stabilization, not navigation.

Navigation: Even when simple lightweight cameras are used, work on navigation often depends on known patterns or environments. For example, Tournier et al. [20] used Moire patterns pasted in the environment to estimate the attitude and position of quad-rotor vehicles. Soundararaj, Prasanth and Saxena [21] and Courbon et al. [22]) used vision to fly in *known* environments; however their method does not apply to scenarios where full visual databases are not available. Mori et al. [23] used markers to stably hover a co-axial helicopter and go from one marker to another.

Other Related Works: In other related work on indoor navigation, Michels, Saxena and Ng [6] used an on-board camera for autonomous obstacle avoidance in a small RC car driving at high speeds. They compute image features that capture single image distance cues (Make3D, [5], [24], [25]) to predict distances to obstacles. Our work is motivated by this work; however their obstacle avoidance algorithms for a ground robots do not directly apply to obstacle detection for MAVs.

One can also build a map of the environment using Visual SLAM (e.g., [26], [27], [28]), in which the map is built using images captured from a camera. Ribnick et al. [29] estimate positions and velocities from monocular views. Such methods for estimating position/location from visual landmarks have been used in many robots which navigate on 2D ground. Celik et al. [2] use vision to reconstruct 3D



Fig. 3. Our co-axial indoor platform (inset shows the camera used).

properties of the environment for UAV path-planning; this technique only applies to situations where there are strong feature points that could be tracked from frame to frame, and would not apply in many indoor environments that are devoid of trackable features (e.g., walls). Our approach does not explicitly perform 3D reconstruction and hence is less computationally intensive.

Our vision-based algorithm does not require high-resolution cameras; in fact, our algorithm is robust enough that a camera resolution of 128×128 pixels suffices, making our algorithm attractive for even smaller aerial robots. For example, extremely small robots such as 25mm long micro-mechanical flying insects [30] can use miniature cameras. Most flying platforms have a camera installed (e.g., [31]), and therefore our algorithms could also be used on them.

III. HARDWARE PLATFORM

Our primary platform is the Parrot AR.Drone quadrotor (Figure 1). This inexpensive toy, currently available to the general public and contains two cameras (one facing forward, the other horizontally downwards), a sonar height sensor, and an onboard computer running proprietary software for communication and command handling. Commands and images are exchanged via a WiFi ad-hoc connection between our host machine and the AR.Drone. Our algorithms run on the host machine—a 2010 Apple Macbook Pro (2.66GHz Intel Core i7, 4GB RAM), running Ubuntu 10.04.

We get an image of resolution of 320×240 pixels from the AR.Drone's forward-facing camera, and get 88×72 pixels from the downward-facing camera. Due to the limitations of the proprietary software on the drone, the only way to receive both images is to overlay one on top of the other. Therefore, we cannot use the full image data on the front camera, as we extract the bottom camera's image from it.

As a secondary platform, a co-axial hobby helicopter was used (The Blade CX2 Coaxial Micro Helicopter, shown in Figure 3) with a Spectrum DX6i 2.4GHz transmitter to communicate. The camera on this platform is a miniature on-board KX141 camera, with a 795×596 resolution. However, we choose to use the AR.Drone as our primary platform for its stability and previously developed API.

On both platforms, three sonars (LV MaxSonar-EZ0) are used for proximity detection on the sides and front of the MAV, and one additional sonar sensor for height detection. We transmit the short-range sensor data back to the host computer using an XBee transmitter module. These inexpensive MAV platforms are less stable (i.e., they drift quite a lot) and are not capable of performing acrobatic maneuvers.

IV. APPROACH

One possible approach to navigation is to create a 3D model of the environment from camera images (using techniques such as structure-from-motion [2]) and then perform path-planning. Such an approach requires high camera resolution and often constructs incomplete models of the environment [3], [32]. Furthermore, even if complete models of the environment could be created, creating these models is computationally expensive. This causes unacceptable delays between the perception and the action of the MAV, thus severely limiting the MAV's autonomous abilities.

We note that a tight connection between perception and action is crucial in attaining the quick real-time responses required to navigate in constrained indoor environments. Our approach is also motivated by the studies on insect flight that demonstrate houseflies do not explicitly comprehend the 3D structure of their surroundings, but rather react to basic stimuli such as optical flow [33]. Therefore in our approach, the vision algorithms are closely coupled to the control, minimizing delay.

We surveyed the types of indoor environments found in common home and office buildings. We found that the majority of the indoor environments belong to the following categories: long spaces (e.g., corridors), staircases, and other (such as rooms and corners). Each type of environment has a certain structure that lends itself to simple navigation rules. For example, in order to fly in a corridor, we only need to figure out the distance from the ground/walls and the direction of the end of the corridor (Figure 4). When facing a staircase, we only need to know the center of the staircase and which direction to fly in (Figure 5).

Given images of the current environment, our classifier detects the environment type. Depending on the classifier result, the MAV follows different strategies: for corridors and stairs, it relies on perspective cues to navigate; and for corners, it instead relies on short-range sensors to navigate. In the future, this setup can be augmented with additional algorithms for new environments and thus expanded beyond the scope of this work.

V. ALGORITHMS

Indoor environments are comprised of long straight parallel lines, and from the MAV's perspective each environment type has unique visual cues. Our algorithms are designed to take advantage of these perspective cues to navigate the MAV through the environment.

A. Corridor

When in a corridor, the goal of the MAV is usually to fly to one end of the corridor. The ends of the corridor can be observed as vanishing points in images from the MAV's camera. Vanishing points have been used successfully for estimating 3D structure from a single image [34]. In indoor environments, vanishing points can be found consistently and therefore we use them to locate the end of the corridor.

In detail, we first compute the edges of the image using the Canny edge detector, and then use the probabilistic Hough transform to find long lines. In a corridor environment, the long lines along the corridor converge towards a vanishing point at the end of the corridor. Because of errors, all the lines do not intersect at exactly one point, so we look for the region in the image which has the highest density of pairwise line intersections (Figure 4). To do this, we divide the image plane into a 11×11 grid G and count how many line intersections fall into each grid element. Let (a^*, b^*) be the grid element that has the highest number of intersections.

More formally, let

- ℓ_l : a line in the image, discovered via Hough transform. Let L be the number of lines found in the image, and $l \in [0, L)$. The line ℓ_l can be represented by parameters m_l, b_l as $y = m_l x + b_l$.
- $(x_k, y_k) \in \mathbb{R}_+^2$: the coordinates of the intersection of two lines. There are K total intersections. In detail, if the lines ℓ_i and ℓ_j do not intersect, let $(x_k, y_k) = (\infty, \infty)$. If they do, then we get them by solving $y_k = m_i x_k + b_i = m_j x_k + b_j$.
- G : The $n \times n$ grid that tiles the image plane ($n = 11$ in our experiments). $G_{a,b}$ represents the number of line intersections falling in the grid element (a, b) . ($a, b \in [0, n)$ are integers.). I.e.,

$$G_{a,b} = \sum_{k=1}^K 1\{a \leq \frac{nx_k}{w} < a+1, b \leq \frac{ny_k}{h} < b+1\}$$

where w is the width of the image, and h is the height of the image.

The grid with the maximum number of intersections is:

$$(a^*, b^*) = \arg \max_{(a,b)} G_{a,b}$$

Therefore the initial estimate of the vanishing point is:

$$(x^*, y^*) = \left(\frac{w}{n}(a^* + 0.5), \frac{h}{n}(b^* + 0.5) \right)$$

However this estimate is noisy, and therefore we average the actual location of the intersections lying near (x^*, y^*) in order to accurately estimate the vanishing point. \mathcal{N} represents the set of points lying close to (x^*, y^*) :

$$\mathcal{N} = \{s \in [0, K) : \|(x_s, y_s) - (x^*, y^*)\|_2 \leq \delta\}$$

where δ is a distance threshold. We then compute the new estimate of the vanishing point as:

$$(\bar{x}, \bar{y}) = \frac{1}{|\mathcal{N}|} \sum_{s \in \mathcal{N}} (x_s, y_s)$$

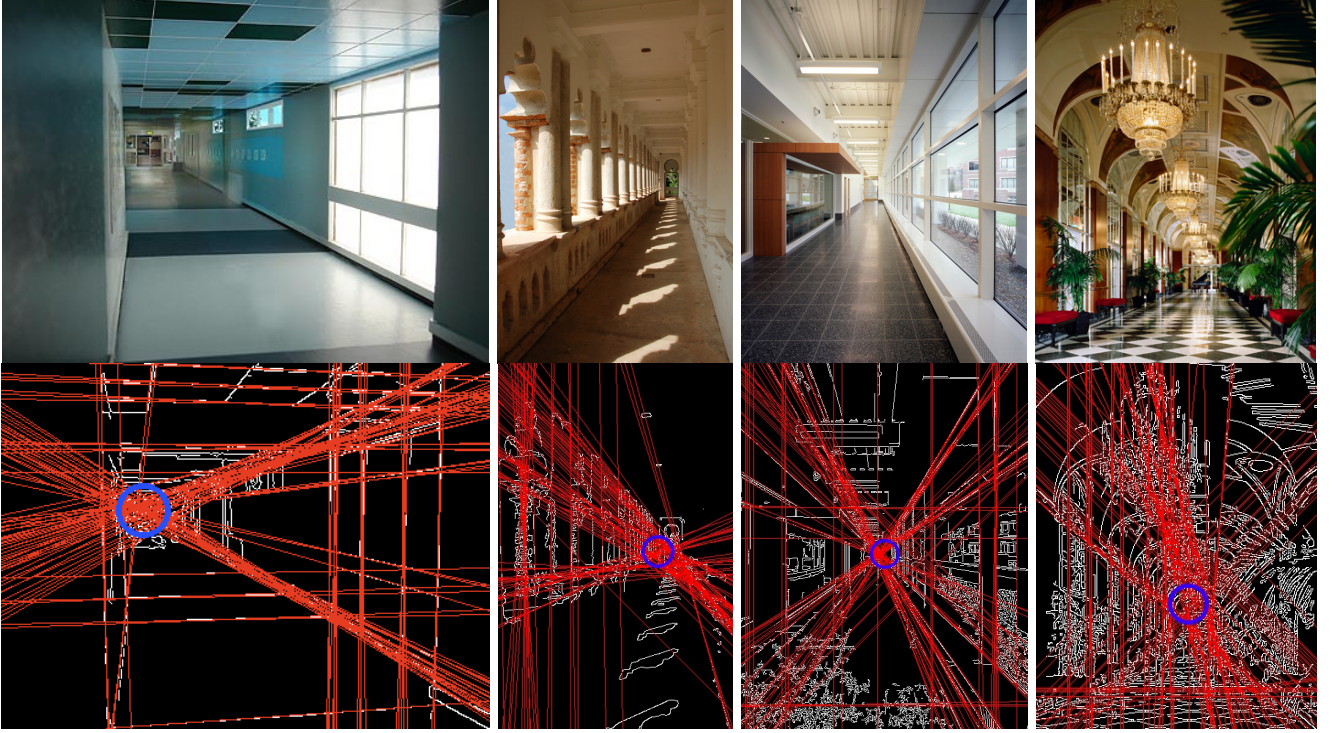


Fig. 4. (Top) Original corridor images. (Bottom) Corridor processed with Canny edge detector and Hough transform to find vanishing point using our grid-based approach. The vanishing point is marked with a blue circle. Note that in the rightmost image, the confidence of the vanishing point estimate would be low because the variance σ^2 would be high (see text).

If the pair-wise intersections are closely concentrated, then it indicates a high confidence in the estimate of the vanishing point; therefore we compute the variance:

$$\sigma^2 = \frac{1}{|\mathcal{N}|} \sum_{s \in \mathcal{N}} \|(x_s, y_s) - (\bar{x}, \bar{y})\|_2^2 \quad (1)$$

It is important that the algorithm has low false positive rate (i.e. it does not report a vanishing point when none is actually present) because the MAV requires precision control. We use the number of pair-wise intersections found (i.e., G_{a^*, b^*}), the variance (σ^2) and the presence/absence of vanishing point in the previous frame in order to compute the probability that there is a vanishing point in the current frame. If the previous frames in the video sequence had strong vanishing points, the current frame is likely to have a vanishing point. Furthermore, the current frame’s vanishing point is likely to be close to those of the previous frames. We construct a probability model with Gaussians to model the noise in the location of the vanishing point. The probability of the location of the vanishing point X_t is given by the product of two conditional probabilities: $P(X_t|X_{t+1})$ and $P(X_t|I)$, i.e., we model it as a Markov Model, and solve it using standard Viterbi algorithm [35].

B. Staircase

In order to fly up a staircase, the MAV needs to advance and ascend carefully without hitting the stairs, and remain in the center of the staircase.

Our goal is to find the center of the staircase, so the MAV can follow the staircase using its low-level controller.

We start by detecting the horizontal lines that form the staircase with the Canny edge detector and probabilistic Hough transform to acquire line segments. Typically, an image has many horizontal lines in an image, and therefore we need to specifically look for a sequence of horizontal lines that represent the staircase. We do so by classifying the line segments as horizontal or vertical and looking for the largest horizontal-line cluster in the Hough transform space. Our estimate of finding the stairs is based on the number of lines found in the cluster. Once the cluster of horizontal lines is detected, we take the mean of the lines’ endpoints; this gives the desired direction the quadrotor should go to—we show this by the red vertical line in Figure 5.

When an MAV, flying upwards, reaches the top of a flight of stairs, the forward-facing camera of the MAV can no longer see the stairs; the stairs are underneath the MAV, out of view. In order to detect that the MAV is above a flight of stairs and to increase its navigational accuracy, we use the downward-facing camera on the MAV. Again, using a Hough transform, we extract the long lines in the image, then filter them. The filter picks out lines within 45 degrees of horizontal (perpendicular to the MAV’s direction of travel). Two examples are shown in Figure 6.

Because most detectable lines in indoor environments are at right angles, the slopes of the resulting lines are used to fine-tune the MAV’s heading, with the goal to make the lines detected perpendicular to the MAV’s direction. The location of the detected lines is also used to help center the MAV on a desired path. This fine-tuning helps the MAV return to a desired path if it drifts off course.

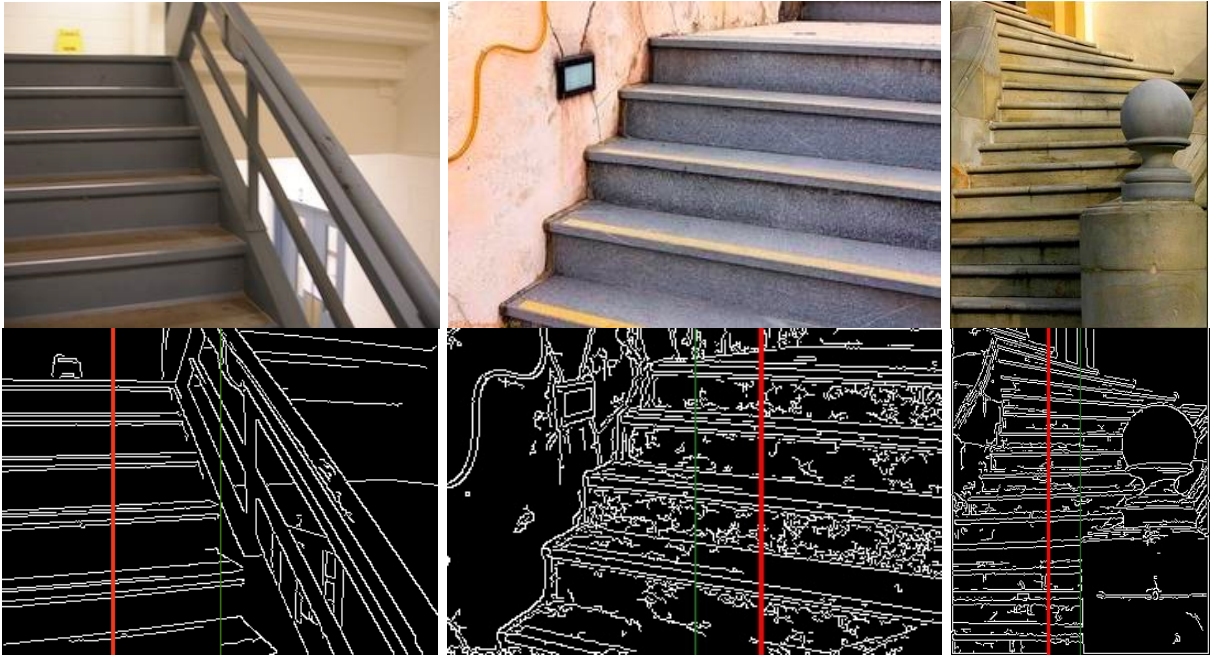


Fig. 5. (Top) Original images of staircases. (Bottom) Image with bold red line marking location of staircase. Green line marks center of image for reference. For example, if the red line is to left of green line, the MAV should turn left.

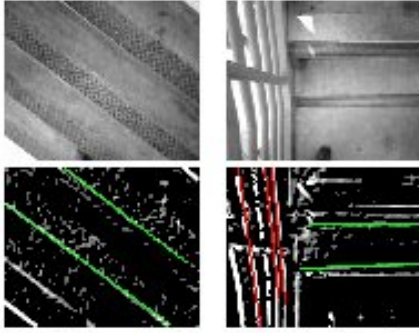


Fig. 6. (Top) Original images from bottom camera. (Bottom) Detected lines with filter applied. Accepted lines are green, rejected are red. The left set shows the MAV in a situation needing yaw correction, the right needing drift correction.

There are two major components to control the MAV's heading: Yaw and Drift. Yaw is the adjustment to the rotational direction of the MAV. We compute the average slope of the filtered lines from the downward facing camera (and call it *Slope*). Let *Pos* be the distance of the desired direction (i.e., the red vertical line in Figure 5) from the center of the image in the front-facing stair algorithm. We adjust the Yaw as:

$$\Delta \text{Yaw} = \alpha_1 * \text{Slope} + \alpha_2 * \text{Pos}$$

where α_1, α_2 are the gain parameters.

Drift is the side-to-side movement of the MAV. Finding the drift adjustment using the bottom image is a little more difficult, as we only want the MAV to drift if it is properly oriented. We thus reintroduce the average slope from above, combined with the average *offset* (from center) of the line midpoints in the downward-facing camera. Using these, and

a similar scaling parameter β , we obtain the following:

$$\Delta \text{Drift} = \beta * \frac{\text{offset}}{1 - |\text{slope}|}$$

Because we are dividing offset by $1 - |\text{slope}|$, ΔDrift will approach 0 as *slope* approaches 1 or -1 (45 degrees from horizontal), and will peak when *slope* is 0. This reduces the drift correction in situations when the MAV is not properly oriented. The drift correction is especially useful for returning to the desired path when the close range sensors cannot detect anything (i.e. stairs with an open railing).

C. Environment Classifier

We used two algorithms for classifying which environment the MAV is currently in. The first one uses GIST features with a learning algorithm, and the second uses “confidence” values generated by the algorithms described above.

GIST Classifier: We first extract the GIST descriptor [36] of each image by shrinking the input image down to 128×128 . We then use SVM [37] for classifying indoor images into these three categories: (1) corridor, (2) stairway, (3) room (neither). GIST descriptors are well-suited for this task because they directly measure the global distribution of oriented line segments in an image, which takes advantage of the long lines found in indoor images. This classifier is used with the helicopter platform, but the quadrotor uses our second algorithm below.

“Confidence” Classifier: In this classifier, “confidence” estimates are gathered from the stair and corridor vision algorithms. These confidences are based on the number of lines found in the respective cluster. The confidence values are then compared, and the highest (above a threshold) is deemed the current environment the MAV is in. For example,

in the corridor algorithm, if 80% of the lines cross the vanishing point, the algorithm will return a high confidence. Where as if only 20% of the lines cross the vanishing point, it will have a low confidence. If all of the vision algorithms return a confidence below a threshold, then the current environment is classified as unknown.

D. Corners and Unknown Environments

Connecting stairs and corridors are corners and rooms where our vision algorithms would not be able to reliably find perspective cues. This is often because only a blank empty wall is visible by the front-facing camera. In such areas, the MAV's goal is to explore the area to find adjacent environments to fly to where the vision algorithms may work.

In these areas, the system falls back to an exploration algorithm as follows. When the MAV first enters a corner, it keeps on moving forward until the front-facing short-range sensor reports proximity to a wall. Then, it uses the side short-range sensors to turn to the most open area. While turning, it checks the front short-range sensors, and will continue moving if there is enough space. If it does not find an open space, it turns back in the other direction while still looking for an open space. This avoids the MAV turning completely around, unless it has actually reached a closed corner. If at any time another environment is detected, the corner algorithm will give control to the respective vision algorithm(s). We found this approach to work surprisingly well in most situations.

VI. CONTROL

Our control algorithm is a set of low-level proportional-derivative controllers that adjust the MAV's throttle, yaw, pitch, and roll. The low-level controllers work together to stabilize the helicopter. (The quadrotor has some stabilization in its proprietary low-level controllers.)

Navigation to reach Goal: The corridor and staircase vision algorithms give a value in the range $[-1, 1]$; a value closer to -1 indicates that the goal is to the left of the MAV, and a value closer to 1 indicates the goal is to the right. (A value of 0 means the goal is exactly in the center.) This was used to control the yaw, pitch and drift of the MAVs. (Similar to Section V-B.)

Height Control: In most situations, there is a ϵ -margin within which it is fine to fly the MAV. (I.e., we don't care if the MAV flies closer or farther from the ground, just as long as it does not get too close or far from it.) For our helicopter, we use a PD controller with a non-linear response (flat within the ϵ -margin). For our quadrotor, we provided the output of the sonar height estimator to low-level controllers using a P -controller with similar response.

Short-range proximity sensors: The short-range sonar range sensors work reliably for upto approximately 1.5m. If any of the front or side sensors detects a proximity to an obstacle for more than a few milliseconds, then the low-level controllers move the MAV in the opposite direction of the detected obstacle, until a threshold distance is reached. The proximity sensors take priority over the vision algorithms.

TABLE I. Performance of the GIST classifier for different environments.

Environment	Number of test images	False positive	False negative
Corridor	314	1.3%	8.9%
Staircase	269	9.7%	0.7%
Room	414	8.9%	1.0%

VII. EXPERIMENTS

A. GIST Classification Experiments

We use our GIST classifier to classify which environment the MAV is currently in. We trained our GIST classifier on training images of 464 corridors, 339 stairs, and 484 rooms in a building (Upson Hall at Cornell University). We used a one-against-all method of training/testing. We then tested the GIST classifier on images taken in a *different* building (Phillips Hall at Cornell University). For testing, we had 314 corridor images, 269 staircase images and 414 room images. Table I shows the performance of the classifier.

Note that even though the GIST classifier makes some errors, in practice, other parts of the algorithm correct for them because of the two following reasons. First, our corridor and stair classifiers also report the confidence in their goal-predictions (see Section V-A). Second, we get a series of images over time and one mistake by the classifier will not be detrimental for the MAV's navigation overall.

B. Testing Environments and Conditions

Our tests were conducted in a variety of indoor environments, shown in Figure 7. For a corridor test to be considered successful, the MAV must progress at least 100ft (many tests progressed further). For a stairs test to be considered successful, the MAV must start behind the first step of the staircase and go up the stairs beyond the final step.¹

C. Helicopter Experiments

Our first platform was a co-axial helicopter. In our experiments, we tested two different corridors and two different stairs (in different buildings). Table II shows the quantitative results and Figure 8 shows some screenshots. While our experiments demonstrate that our vision algorithms are quite robust and work on multiple platforms, this platform suffered from severe drift problems. Therefore we performed our full experiments on the quadrotor platform.

D. Quadrotor Experiments

Our second platform was a AR Drone quadrotor. Table III shows our experimental results of flying the quadrotor in three different corridors and three different stairs, in a total of 41 experiments. These environments were quite varied in appearance.

Corridors: We achieve an overall success rate of 92% on the quadrotor platform. Furthermore, we found our corridor algorithm to consistently detect the vanishing point 100% of

¹The Parrot AR.Drone has an automated take-off procedure, however we found it to be unreliable in constrained indoor environments. Due to this, about half of our experiments began with manual take-off by the experimenter, and after the quadrotor reached a height of more than 2ft, the control was given to the vision algorithm.

TABLE II. Co-axial Helicopter: Experimental Performance.

Environment	No. of experiments	Vision success rate	Success rate (full flight)
Corridors (2 different)	4	100%	75%
Staircases (2 different)	10	70%	50%

*Note that this platform suffers significantly from stability problems and does not have downward facing camera.

TABLE III. Quadrotor: Experimental Performance.

Environment	No. of experiments	Vision success rate	Success rate (no contact)	Success rate (full flight)
Corridor 1	9	100%	78%	100%
Corridor 2	7	100%	42%	86%
Corridor 3	10	90%	20%*	90%
Corr. Total	26	96%	46%	92%
Staircase 1	5	80%	80%	80%
Staircase 2	6	83%	67%	83%
Staircase 3	4	100%	60%	100%
Stair Total	15	87%	67%	87%

*This corridor was very narrow - barely 3x the quadrotor's width. Furthermore, there was strong air currents because of air conditioning.

the time in corridor 1 and corridor 2, but only 90% of the time in the corridor 3, as it was dimly lit with little contrast. Flying in a corridor is challenging because the wind from the quadrotor's blades creates turbulence. Because of this, gentle contact with the wall was made in a few tests (especially in corridor 3 which was very narrow). This situation could be fixed using a more robust short-range sensor system. Results for all quadrotor experiments are given for both contact and non-contact outcomes.

Staircases: Flying in a staircase is *more* challenging than flying in a corridor. First, the turbulence is significantly greater, thereby affecting stability. Second, staircases are much narrower giving less margin for error in the controller. Lastly, the quadrotor changes its pitch in order to go up the stairs which changes the angle of the front-facing camera. In our experiments, the quadrotor can traverse a full flight of stairs successfully 87% of the time. Some of the stairs included two flights of stairs, where the quadrotor had to turn—once reaching on the middle level, it turned using “unknown environment” mode, until it saw the next set of stairs to go up.

The video showing some of these test flights is available at:

<http://www.cs.cornell.edu/~asaxena/MAV>

VIII. CONCLUSIONS

We presented a set of vision algorithms to autonomously fly a Miniature Aerial Vehicle (MAV) in common indoor environments based on single image perspective cues. We use classification algorithms to detect the type of environment, and then for each environment we extract perspective cues for estimating the desired direction for the MAV to fly. These vision algorithms are shown to be robust and can be applied to many different types of MAVs, enabling them to traverse corridors, stairs, and corners they have never seen before. Our method requires only a small, light-weight camera and

therefore it is a good solution for MAVs with payload and power restrictions. Furthermore, these algorithms require significantly less computational power, enabling the MAV to quickly react and navigate new indoor environments.

ACKNOWLEDGMENTS

We thank Tung S. Leung, Arjun Prakash and Henry Chan for their help.

REFERENCES

- [1] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, “Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments,” in *SPIE Unmanned Systems Technology XI*, 2009.
- [2] K. Celik, S.-J. Chung, M. Clausman, and A. K. Somani, “Monocular vision slam for indoor aerial vehicles,” in *IROS*, 2009.
- [3] M. Goesele, N. Snavely, B. Curless, S. M. Seitz, and H. Hoppe, “Multi-view stereo for community photo collections,” in *ICCV*, 2007.
- [4] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, “Mav navigation through indoor corridors using optical flow,” in *ICRA*, 2010.
- [5] A. Saxena, S. Chung, and A. Ng, “Learning depth from single monocular images,” in *NIPS*, 2005.
- [6] J. Michels, A. Saxena, and A. Y. Ng, “High speed obstacle avoidance using monocular vision and reinforcement learning,” in *ICML*, 2005.
- [7] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, “An application of reinforcement learning to aerobatic helicopter flight,” in *NIPS*, 2006.
- [8] E. Feron and S. Bayraktar, “Aggressive landing maneuvers for unmanned aerial vehicles,” in *AIAA GN&C*, 2006.
- [9] V. Gavrilits, I. Martinos, B. Mettler, and E. Feron, “Control logic for automated aerobatic flight of miniature helicopter,” in *AIAA GN&C*, 2002.
- [10] A. Coates, P. Abbeel, and A. Y. Ng, “Learning for control from multiple demonstrations,” in *ICML*, 2008.
- [11] J. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, “Quadrotor using minimal sensing for autonomous indoor flight,” in *EMAV*, 2007.
- [12] J.-D. Nicoud and J.-C. Zufferey, “Toward indoor flying robots,” in *IROS*, 2002.
- [13] D. Schafroth, S. Bouabdallah, C. Bernes, and R. Siegwart, “From the test benches to the first prototype of the muffy micro helicopter,” *JIRS*, vol. 54, pp. 245–260, 2009.
- [14] L. Mejias, J. Roberts, K. Usher, P. Corke, and P. Campoy, “Two seconds to touchdown vision-based controlled forced landing,” in *IROS*, 2006.
- [15] R. J. D. Moore, S. Thurrowgood, D. P. Bland, D. Soccol, and M. Srinivasan, “A stereo vision system for uav guidance,” in *IROS*, 2009.
- [16] N. Johnson, “Vision-assisted control of a hovering air vehicle in an indoor setting,” Ph.D. dissertation, Brigham Young University, 2008.
- [17] F. Kendoul and K. Nonami, “A visual navigation system for autonomous flight of micro air vehicles,” in *IROS*, 2009.
- [18] A. Cherian, J. Andersh, V. Morellas, N. Papanikolopoulos, and B. Mettler, “Autonomous altitude estimation of a uav using a single onboard camera,” in *IROS*, 2009.
- [19] C. Fan, S. Baoquan, X. Cai, and Y. Liu, “Dynamic visual servoing of a small scale autonomous helicopter in uncalibrated environments,” in *IROS*, 2009.
- [20] G. Tournier, M. Valenti, and J. P. How, “Estimation and control of a quadrotor vehicle using monocular vision and moirre patterns,” in *AIAA GN&C*, 2006.
- [21] S. Soundararaj, A. Sujeeth, and A. Saxena, “Autonomous indoor helicopter flight using a single onboard camera,” in *IROS*, 2009.
- [22] J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet, “Visual navigation of a quadrotor aerial vehicle,” in *IROS*, 2009.
- [23] R. Mori, K. Hirata, and T. Kinoshita, “Vision-based guidance control of a small-scale unmanned helicopter,” in *IROS*, 2007.
- [24] A. Saxena, S. Chung, and A. Ng, “3-d depth reconstruction from a single still image,” in *IJCV*, vol. 76, no. 1, 2008, pp. 53–69.
- [25] A. Saxena, M. Sun, and A. Ng, “Make3d: Learning 3D Scene Structure from a Single Still Image,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 824–840, 2008.
- [26] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardos, “An image-to-map loop closing method for monocular slam,” in *IROS*, 2008.

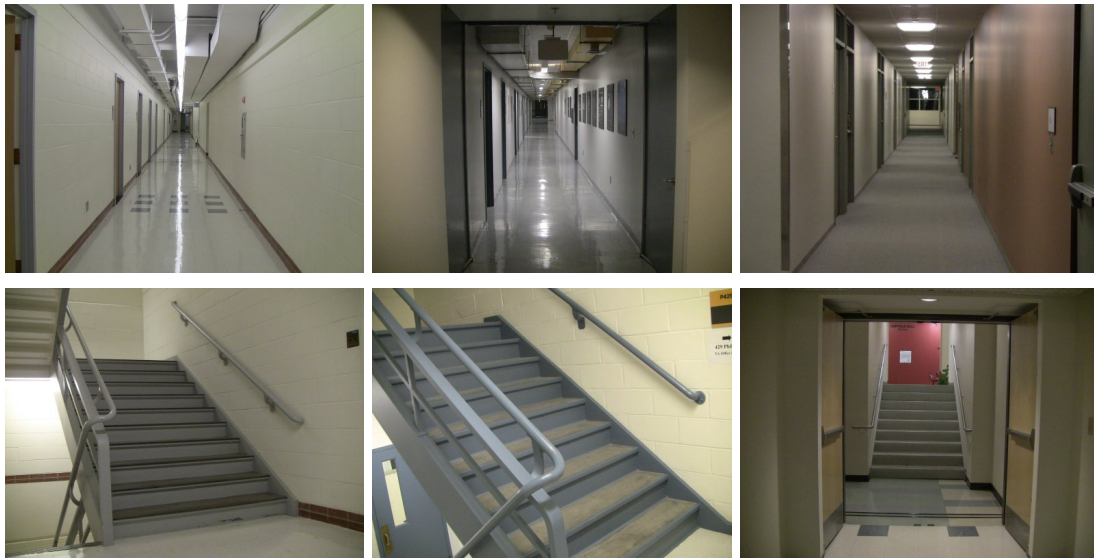


Fig. 7. Corridor and stair environments tested. (Top row) Corridors, (Bottom row) staircases. Note the variations in their appearance.

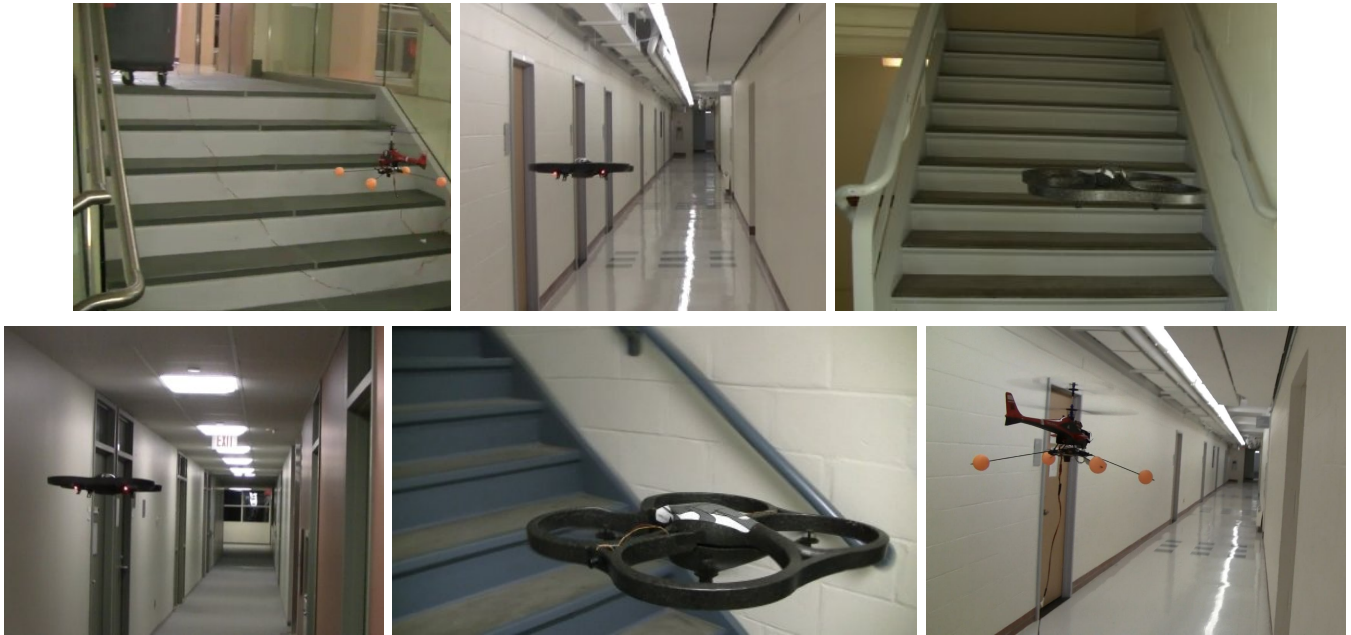


Fig. 8. Screenshots of our MAVs flying in different environments. First and last image show co-axial helicopter and the others show our quadrotor platform.

- [27] M. Cummins and P. Newman, "Fab-map: Probabilistic localisation and mapping in the space of appearance," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.
- [28] B. Steder, G. Grisetti, C. Stachniss, and W. Burgard, "Visual slam for flying vehicles," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1088–1093, 2008.
- [29] E. Ribnick, S. Atev, and N. Papanikolopoulos, "Estimating 3d positions and velocities of projectiles from monocular views," in *TPAMI*, vol. 31, no. 5, 2008, pp. 938–944.
- [30] X. Deng, L. Schenato, W.-C. Wu, and S. Sastry, "Flapping flight for biomimetic robotic insects: Part ii- flight control design," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 789–803, 2006.
- [31] Parrot, "Ar.drone," <http://ardrone.parrot.com/>, 2010.
- [32] A. Saxena, J. Schulte, and A. Y. Ng, "Depth estimation using monocular and stereo cues," in *IJCAI*, 2007.
- [33] D. Marr, S. Ullman, and T. Poggio, *Vision: a computational investigation into the human representation and processing of visual information*. W.H. Freeman, 1982.
- [34] A. Criminisi and A. Zisserman, "Single view metrology," in *IJCV*, vol. 40, no. 2, 2000, pp. 123–148.
- [35] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [36] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *IJCV*, vol. 42, pp. 145–175, 2001.
- [37] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector learning for interdependent and structured output spaces," in *ICML*, 2004.