Efficient Algorithms for Indoor MAV Flight Using Vision and Sonar Sensors

Kyungnam Kim(✉), David J. Huber, Jiejun Xu, and Deepak Khosla

HRL Laboratories, LLC, 3011 Malibu Canyon Rd, Malibu, CA 90265, USA
kkim@hrl.com

Abstract. This work describes an efficient perception-control coupled system and its underlying algorithms that enable autonomous exploration of indoor environments by a Micro Aerial Vehicle (MAV) equipped with a monocular camera and sonar sensors. The perception subsystem uses inputs from the camera to detect the vanishing point and doors in corridors. It detects the vanishing point by grid-based line-intersection voting (GLV) and Mixture-of-Gaussians (MoG)-based classification, while doors are detected by using simple but effective geometric scene properties (GSP) with template matching and temporal filtering. It also detects distance to obstacles, for example walls, using inputs from one forward-looking and two side-looking sonar sensors. These algorithms are accurate, computationally efficient, and suitable for real-time operation on offboard and onboard power-constrained computing platforms. The control subsystem employs a priority-based planner that combines outputs from the perception subsystem to compute high-level direction and velocity commands for the MAV. We evaluate our perception-control system on a commercially available AR.Drone 2.0 MAV with offboard processing and successfully demonstrate collision-free autonomous exploration and flight in building corridors and rooms at approximately 2 m/s speed.

1 Introduction

Micro Aerial Vehicles (MAVs) require real-time accurate perception and control capabilities to autonomously navigate in many applications, such as search-and-rescue, mapping, and unmanned surveillance. Perception and control technologies for MAVs are an active research area especially due to their widespread availability, flexibility, and low cost. However, the current state of the art approaches are generally computationally complex and require *a priori* mapping and knowledge of the environment to navigate. In indoor environments, detection of vanishing point and doors in corridors and rooms is a necessary capability to enable the MAV to autonomously explore (e.g., enter a room through a door) and plan its flight in real-time. This requires efficient perception and control algorithms that can be mapped to power-constrained computing platforms.

This work describes a simple yet efficient coupled perception and control approach primarily designed to work for MAV in indoor environments. The perception subsystem uses inputs from the camera to detect the vanishing point and doors in corridors and rooms, while the control subsystem uses this information to plan the flight path and automatically navigate the MAV.

G. Bebis et al. (Eds.): ISVC 2015, Part I, LNCS 9474, pp. 419–431, 2015.
DOI: 10.1007/978-3-319-27857-5_38

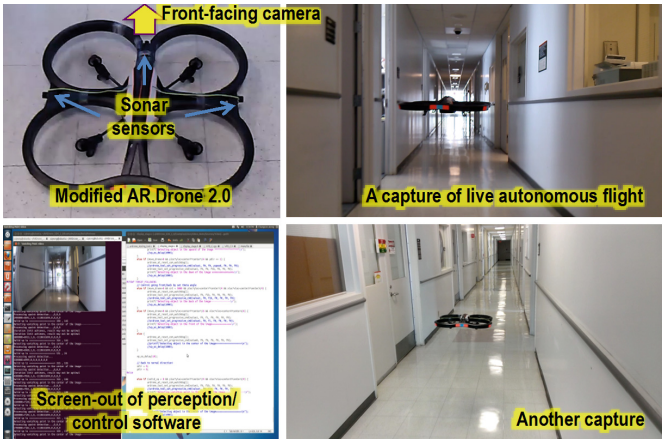

Fig. 1. Our modified AR.Drone 2.0 used in the experiments. From the laptop display, we can monitor the perception outputs and control commands being executed. Two sample screen snapshots of live MAV flight are shown here.

Our approach to detecting a vanishing point begins with fast edge detection and linking followed by grid-based line-intersection voting and Mixture-of-Gaussians (MoG)-based classification. This is accurate and robust in a variety of illumination and conditions, and yet more efficient than previous approaches that use computationally intensive algorithms, such as the Hough transform and Markov modeling [1]. Based on the detected vanishing point, our door detection algorithm chooses door candidates using the geometric scene properties, such as door width and relative door orientation, and intensity profile information, such as the mean and variance of intensities in the door and its adjacent areas. If no vanishing point is detected, a template matching-based approach is used to detect door lines that look similar to the last detected door boundaries. The final door candidates are selected by temporal filtering to eliminate inconsistent detections.

The topic of door detection using 2D and other sensors has been previously investigated [2–5]. The work in [2] detects door corners first and then groups the four corners to form door candidates. The grouped door-corner candidates are then verified by matching and combining edges and corners. Other work [3, 4] rely on both a camera and a 2D laser range finder. The method in [3] relies on detection of doorknobs and frames as additional information. However, these features are not always observable and limit the applicability of this method. The approach in [5] depends on not only shape information, but also appearance information, and uses a Bayesian model of shape and appearance likelihood to verify door candidates formed from detected corners. The methods, such as in [2, 5], usually need to deal with validation of several door candidates since there are hundreds of corner groups. Therefore, they are computationally intensive and not suited for real-time applications.

Our door detection approach is different from prior work in several aspects. First, it is based on linked edge information obtained from a single monocular camera and does not depend on the appearance of a door or detection of specific door parts, such as

doorknobs and doorframes. Second, it leverages the scene geometry information about the orientation of door lines and intensity profile of doors compared to its neighboring areas. This makes our approach work under different illumination conditions and/or viewing angles in the corridors. Third, it is efficient and can run onboard resource-limited computing platforms. Finally, it does not require any training.

The perception algorithms described above have been implemented and tested using a laptop and a quadrotor MAV (AR.Drone 2.0 [6]) for autonomous flight in a corridor (see Fig. 1). In order to accommodate indoor navigation challenges, such as unexpected air drifts, self-turbulence, and obstacles during flight, our control strategy combines a visual sensor for primary navigation with front and side-mounted sonar sensors to detect obstacles and avoid collision. A sonar sensor measures distances to obstacles. The control subsystem assigns different priorities to the sensor modalities and employs a planner that combines outputs from the perception subsystem to compute high-level direction and velocity commands for the MAV in a closed-loop manner. Our approach does not require optical flow calculation, complicated learning strategies, or additional external sensors.

The rest of the paper is organized as follows. Section 2 describes our proposed perception subsystem and its underlying algorithms. Section 3 described the details of our control subsystem. Section 4 presents experimental results of the proposed algorithm on MAV flight in real-worlds. Finally, Sect. 5 discusses the results and conclusion.

2 Perception Subsystem

Our perception subsystem consists of several processes to detect vanishing point (VP) and doors. Figure 2 illustrates the perception subsystem block diagram. Its input is video from a single front-facing monocular camera mounted on the MAV. An edge detector is first applied to each frame of the incoming video. The detected edge points are then connected to longer edges through edge linking. Horizontally-oriented edges are used to calculate the vanishing point, while vertically-oriented edges provide the initial candidates for door lines. If the vanishing point is available, the door candidates are evaluated using their geometric properties (e.g., length, location, distance) and intensity profiles. The detected vanishing point and doors are used by the MAV to autonomously explore the indoor

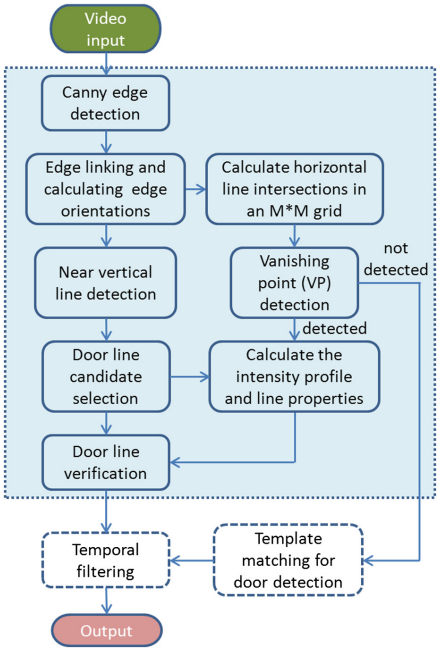

Fig. 2. Perception subsystem block diagram for vanishing point and doors detection using a single monocular camera.

environment. Details of the vanishing point and door detection algorithms are described below.

Vanishing Point Detection. The vanishing point (VP) is a useful destination for a MAV in a corridor, but can also be used for guiding the perception subsystem in finding doors. Due to its importance in our approach, the VP detection must perform accurately in real-time with high probability of detection and low false positives. To satisfy these requirements, we have developed an efficient VP detection algorithm that starts with the edge detection and linking described above and follows with grid-based line-intersection voting (GLV) and MoG-based classification.

A. Edge Detection

Given each frame in the incoming video, the Canny edge detector [11] is used to detect possible edge pixels whose intensity gradients fall within a preselected range. In addition, non-maximum suppression is used within the Canny detector to retain only thin lines (i.e., edge-thinning). The gradient directions at all edge pixels are calculated and discretized into N bins. Edge pixels with gradient orientations belonging to the same bin are linked using connected component analysis. In this way, shorter edges are connected to form longer lines. Each line's orientation is calculated and stored as its geometric feature along with its two end points and length. These line features are later used for door detection.

B. Grid-based line-intersection voting (GLV) and MoG classification

Based on the long edges detected in the image frame, we now estimate the position of the vanishing point (VP). This is the point where corridor horizontal lines intersect in the image frame. To detect the vanishing point in each frame, the previously detected edge lines are first filtered to remove lines that are too close to the horizontal direction (left to right). Specifically, if the angle between an edge line and the image horizontal line is below a threshold (e.g., 5°), this line is ignored for the VP calculation. The intersections of remaining lines are then calculated. To find the VP, the entire image area is split into an $M \times M$ grid and the number of line intersections within each grid cell is calculated. The cell with the maximum number of line intersections is assumed to contain the true vanishing point. This is based on the observation that there are many lines parallel to the floor of the corridor floor and they mostly intersect at the end of the corridor.

We define the centroid of intersections within the chosen cell as the vanishing point. To further reduce false alarms, a mixture of Gaussian (MoG) model is trained to classify a centroid of the intersections within a grid cell as the VP. The MoG model is represented as

$$p(v|\lambda) = \sum_{i=1}^{k} w_i g\left(v \middle| \mu_i, \sum_i\right)$$

where v is a 2-by-1 vector consisting of the number of intersections in the selected VP grid and the variance of their 2D locations, K is the number of Gaussian components (e.g., 3), w_i is the Gaussian component's weight and $g\left(v|\mu_i, \sum_i\right)$ is the Gaussian

component with a mean μ_i and covariance matrix $\sum_i$. Given a set of training data with labeled VP grids, we learn the parameters w_i, μ_i and $\sum_i$. Once a new grid cell is detected as a candidate VP grid in the image frame, we use the above MoG model to calculate its probability of containing a true VP.

Our VP detection algorithm is unique in (1) using the grid-based voting to find the cluster of near-horizontal edge intersections, and (2) using MoG classification for estimating the confidence of detecting a VP. In addition, our algorithm is computationally efficient. The first step of Canny edge detection is extremely fast. For a typical 720 × 480 image, it only requires about 2 ms to detect edges. The edge linking step depends on the complexity of the scene. Its computation is on the order of the number of edges in the image and typically takes about 20 ms. The other steps in VP detection require only a few additional milliseconds. Typically our VP detection for a 720 × 480 image takes about 30 ms per frame and therefore achieves real-time 30 fps speed. The closest prior work to our VP detection approach is described in [12]. It uses the Hough Transform (HT) to detect lines (HT actually has Canny operators inside) and additionally requires estimation of parameters. We use the MoG model for estimating the probability of a VP. In contrast, [12] uses a Markov Model to estimate the probability of the location of the vanishing point and reduce false alarms, and its inference was done using the Viterbi algorithm, which increases its computational complexity/time. We implemented and evaluated the Markov Model according to [12] as a baseline experiment. Our MoG results are presented in Sect. 5.

Door Detection. We propose a computationally-efficient door detection algorithm that is based only on edge and geometric shape information. Our algorithm assumes doors are mostly imaged upright in the video which is a reasonable assumption for a stable MAV flight. It uses the near-vertical lines obtained from the previous VP detection as the initial door candidates, and then filters them based on their geometric scene properties (GSP) to detect doors.

A. Edge Based Door Detection

When a MAV is flying down a corridor, doors are usually aligned along the vertical direction in each frame. Using this assumption, we first select the edges that are close to the vertical direction, allowing some small angular deviation of α degree (for example ±10°). In addition, the door line candidates are filtered according to their lengths and locations of their lower and upper end points. In typical indoor situations, the image location that divides the left and right walls of a corridor is at the vanishing point and is usually close to the image center. For simplicity, we detect doors separately on the left side and right side of this location. For detecting doors on the left side, only edges in the leftmost portion of the image are considered. In our approach, we used the leftmost fraction λ% of the image width (e.g., 35 % of the image width). A similar approach is used for detecting doors on the right side.

The initially filtered vertical edges are further merged by connecting short edges to form longer lines. Only edges with similar distances to the middle point of the left-most image border and similar edge orientations are merged, which reduces the number of candidate door lines and saves computation. This process generates longer vertical lines that may better represent the candidates of door lines. These candidate lines are further

validated by a series of filtering (validation) processes described below to detect the true door that is closest to the MAV on each side of the corridor:

(1) If the candidate line intersects the top or bottom of the image, but the intersection is outside of the image, it is not a valid candidate and removed.
(2) The distance from the vanishing point to the candidate door line is calculated. If the door line is too close to the vanishing point (e.g., 10 % of image width), we assume that it does not belong to a door and is removed.
(3) Each candidate line is paired with another candidate line to form a candidate door. If the distance between a pair of lines (i.e., the candidate door's width) falls outside of a typical door width (e.g., 15 % to 35 % of image width), this candidate door is invalid.
(4) We use the intensity profile of the candidate door and its adjacent sides to further filter door hypotheses even further. Specifically, the image area around the pair of door lines is split into three parts as seen in Fig. 3: the portion to the left of the door (A), the door (B), and the portion to the right of the door (C). An intensity profile is calculated for these parts. As illustrated by Fig. 3, a horizontal line is crossed over the door lines and the pixels along the horizontal line are used to calculate the intensity profile. One can see that the horizontal line is split into three segments by the pair of door lines. The mean and variance of intensities within each segment are calculated. The candidate pair of door lines is filtered according to their intensity profile and the prior knowledge of the corridor. For example, if the door is open, its mean intensity is often lower than the other two parts for a white wall. In addition, the intensity variance within an open door area is often larger than the other two parts because of the cluttered objects in an office. This filtering process needs to be adjusted for different scenarios because the prior knowledge may change.

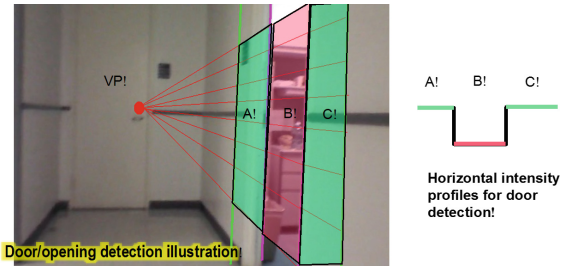

Fig. 3. Calculation of the intensity profile at the adjacent areas of a candidate door.

(5) The above step 4 is processed repeatedly at multiple vertical locations for more reliable decision, as shown with the red lines in Fig. 3, along the candidate door lines. The number of iterations that pass the intensity profile filtering is accumulated. If the total number is above a predefined threshold (e.g., 50 %), the pair of door lines is declared as valid and a door is reported to be detected.

In the current implementation, we only find a single door at each side of the corridor, which usually corresponds to the door closest to the MAV. However, the above process can be applied iteratively to find more doors.

B. Template Matching for Door Detection

The edge-based door detection method described above requires the vanishing point to validate door hypotheses. Therefore, it can only be applied when a vanishing point is successfully detected. However, there might be cases where neither a vanishing point is detected nor are door hypotheses generated due to low contrast caused by illumination changes and motion. In such cases, we need other methods to detect doors. One such method is to apply template matching based on previously detected doors. When a vanishing point and door are successfully detected in a frame, we extract the narrow image patch around the door's boundary as a template and store such templates in a buffer. This buffer is updated online and it automatically clears old templates after a few frames (e.g., 5 frames). When a new door is detected, its template is added to the buffer.

When no vanishing point is detected or no door hypotheses are generated in a new frame, the latest template in the buffer is used to compare against the new image frame to find a matching image patch. This is achieved using a standard template matching approach based on normalized correlation [13]. If the matching score is above a predefined threshold (e.g., 0.95), a door line that looks similar to the previously detected door line is detected. The previously detected door line is then updated with the new location.

The above template matching method can generally find the new location of a recently detected door line. However, the template buffer size must be carefully controlled because template matching may fail when the actual door is already too far from the previously detected location. In such cases, the door hypothesis generated by template matching should be rejected. When no door is detected for a long period, the template buffer will become empty and no template matching can be applied.

C. Temporal Filtering of Detected Doors

The door detection generated by all processes discussed above may still be noisy, considering the large variations and clutter in typical indoor settings. To further reduce errors, we use a temporal filtering to ensure that the detections are temporally consistent. The filtering process checks if a door is continuously detected in each frame within a predetermined temporal window. If the door is detected in most of the frames within this window (e.g., 3 out of 5 frames), it is considered a valid detection. Otherwise it is rejected as a valid door. This temporal filtering is an optional process for refining door detection.

Our GSP-based door detection approach depends only on the geometric scene properties of door edges (e.g., intensity profile, door width, relative location of the top/bottom points of door lines) instead of complex structures like door corners [2, 5] or door knobs [3], which are sometimes not observable in the image. Since our approach only requires the commonly and reliably occurring features of a door (i.e., the two long edges) to generate door hypotheses, it is readily applicable to real-world settings. The validation of door hypotheses is based on these general geometric properties. For corridor scenarios, our algorithm also relies on the generally reliable

vanishing point position to filter door candidates. In addition, we have added a template matching process to deal with situations where the vanishing point is not available or no door hypotheses are found. Finally, our algorithm is efficient because it re-uses the edges detected by the VP detection step. The only additional computation is the selection of near-vertical line pairs and validation of the selected pairs. Since we limit the width of the door with respect to its height, the complexity of the door candidate selection and validation is on the order of the number of long vertical lines. In typical corridor scenarios, there are a few dozen such long near-vertical lines and the validation process usually requires less than 10 ms. The VP detection and door detection together can run at about 20 fps on a standard quad-core laptop (Sect. 4), which is fast enough for a normal speed MAV to navigate through the corridor.

3 Control Subsystem

Our control strategy combines a visual sensor for primary navigation with side-mounted sonar sensors to avoid obstacles that fall outside of the field of view. Unlike other approaches for MAV control in the prior art that incorporate optic flow to dictate MAV direction [7, 10], our approach combines feature matching from the front-facing camera with additional override inputs from the lateral sonar sensors. While computationally inexpensive, optical flow approaches to control often exhibit drift in both position and scale. This can present problems in the indoor environment, where even small errors can lead to collision and loss of control. Instead, we employ a feature-based approach wherein we receive a target and track its features across frames. The system (MAV) centers the target in the field of view and then proceeds in its direction, turning as necessary to maintain the correct heading. This straightforward approach minimizes the overall calculation required to visually navigate the MAV toward the target and avoids the need for complicated learning strategies [8] or additional external sensors [9].

A detailed block diagram of our control methodology is illustrated in Fig. 4. The system can also receive input from the front-facing camera and user input via the graphical user interface by clicking the mouse on a location in the field of view. Additionally, a sonar system is used to allow the MAV to determine its distance from obstacles to its right and left, which are out of the view of the camera. Based on the location of the target in camera coordinates relative to the center of the frame, the system will send a small control command to the MAV. The system immediately captures another frame of video and issues subsequent commands in an iterative, closed-loop manner. This continues until the MAV has arrived at the target or the system has lost the target. The sonar sensors run in a parallel control loop that has priority over the main control loop and can interrupt commands from the visual input and issue their own in order to avoid collision.

The standard operation of the system is to detect a vanishing point in a corridor and move the MAV towards it. The vanishing point is computed from the video frames of the front-facing camera using the algorithm discussed in Sect. 3. Alternatively, if the MAV is not in a corridor, the user may use the graphical user interface to choose a specific location in the scene or the location can be computed automatically based on

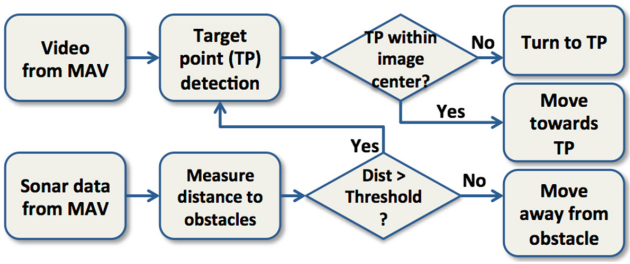

Fig. 4. Block diagram of our control strategy of indoor MAV flight.

heuristics (e.g., a landmark or a corner table) to which the MAV should fly. In either case, the input to the system is a location in camera-coordinates for a specific frame. Once a set of target coordinates has been selected, image information is extracted around its location for a given radius. Feature keypoints are extracted using a local feature algorithm (such as SIFT [14] or SURF [15]) for this image region. When a significant number of keypoints from the target region correspond with keypoints from the next frame, the target location can be imprinted on the next frame and further calculations can be made.

For each iteration of the control algorithm, the system computes the difference between the camera coordinates of the target with the center of the frame; the resulting vector indicates the direction that the MAV must move in order to acquire its target. Steering commands are issued if the horizontal difference between the target and the center of the image exceeds a given threshold. When the target is centered, the MAV will proceed forward until the originally slight discrepancy between the target and center of the image frame becomes large and another steering correction is necessary.

The control system contains sonar sensors to avoid collisions with obstacles to the sides of the MAV. Prior to each iteration of the control algorithm, the system checks the status of the sonar sensors. If the sonar detects that the MAV is too close to an obstacle, the system automatically ignores inputs from the vision system and takes immediate action to avoid the obstacle; this is a simple command to move the MAV in the direction opposite of the obstacle.

4 Experiments

The detection performance of our perception subsystem was evaluated using a number of corridor images under different lighting and viewing conditions. The perception and control subsystems described above were implemented on an offboard laptop computer and tested with a low-cost MAV quadrotor (AR.Drone2.0). We used the AR.Drone's existing front camera as our visual sensor and then added three sonar sensors for obstacle avoidance. The perception software for vanishing point and door detection was integrated with other capabilities that we have additionally developed for sensor data capturing and transmission, basic flight commands (move forward/backward,

left/right, turn left/right, and others) and perception visualization, along with our control algorithm to perform autonomous corridor flying tests (see Fig. 1).

A. Evaluation of Vanishing Point and Door Detection

We tested our VP detection on 2470 test images of 720 × 480 resolution captured in different corridors using the AR.Drone camera (e.g., Figs. 5 and 6). A different set of 120 images was manually annotated with the VP to first train our MoG model. Our VP detection algorithm achieved 97.1 % true positive detection rate (recall) with 3.0 % false positive rate (alarms). In addition, we also implemented a baseline approach based on a Hidden Markov Model (HMM) by following [12] to generate the probability of VP detection through the standard Viterbi inference. This approach performed poorly compared to our approach with 88.4 % true positive detection rate and 5.1 % false positive rate. As described previously in Sect. 2, our computational complexity is lower than this baseline approach (Table 1).

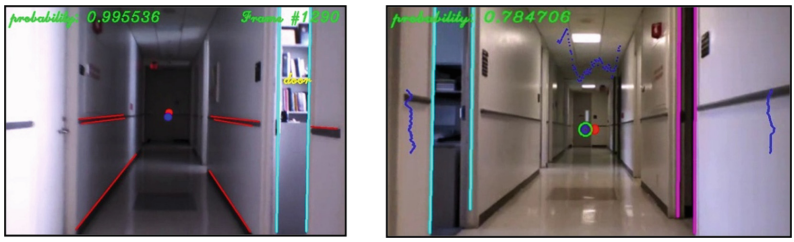

Fig. 5. (Left) An example of vanishing point and door detection during a MAV flight test in a corridor. The cyan lines are the boundaries of one detected door. (Right) Another example of vanishing point and door detection. The cyan (left) and purple (right) lines are the boundaries of the detected doors. The sonar feedbacks (front, left, right) are visualized in the blue plots (Color figure online).

Table 1. Performance evaluation of vanishing point detection

	Training images	Testing images	True positive rate (recall) %)	False positive rate (%)
Use MoG	120	2470	97.1	3.0
Use HMM	120	2470	88.4	5.1

We additionally evaluated our door detection algorithm on multiple MAV flights in a building corridor with multiple doors. The corridors have a total of 23 doors, about evenly distributed on the left and right sides. The average flying speed of our MAV was about 2 m/s. Table 2 summarizes the door detection performance of our algorithm with 10 runs of MAV flight in each direction for a total of 20 runs. We achieved 91.3 % true positive detection rate, with false positive and false negative rates of 4.3 % and 4.4 %, respectively. The main reasons for false positives were large windows, which look similar to doors, and strong vertical shadows on the wall.

Table 2. Performance evaluation of door detection algorithm in a corridor with 23 doors for 20 full runs of MAV flight.

Total # of doors (23 doors over 20 flights)	True positive rate (%)	False positive rate (%)	False negative rate (missed detections) (%)
460	91.3	4.3	4.4

It should also be mentioned that doors were usually hard to detect during fast flight, such as instances where the speed exceeded 5 m/s. The motion blur in such cases significantly reduced the contrast of door lines and made them harder to detect. In addition, the rolling motion of the MAV invalidated the assumption that the door lines were near-vertically oriented, causing increased missed detections and false alarms.

B. Indoor Flight Navigation

In our flight testing, the offboard laptop receives and processes video and sonar sensor data from the MAV to determine whether it should shift/turn or fly forward to the target/destination point (i.e., vanishing point, door location, or user-clicked location). If the destination point is not in the central region of the image, the MAV must to turn/shift to the left or right so that it is centered. If the destination point is not detected, the hovering command is sent to the MAV. These perception and control processes run on a laptop (with Intel® Core™ i7-2860QM CPU at 2.5 GHz running the Ubuntu Linux system). Three USB-powered sonar sensors (LV-MaxSonar-EZ1) were installed on the drone hull in the front and two sides as the proximity sensors. The transmission of sonar readings from the drone to the laptop was done by a Zigbee (XBee DigiMesh 2.4) connection. We developed custom software to interface to the AR.Drone SDK for control and camera data capture. With all sub-systems integrated and running together, we could achieve about 15–20 frames per second of visual processing on the laptop without specific optimization in a single-threaded implementation. The sonar-based update rate was even higher (e.g., >30 fps). Each time the perception module processes the sensor data, the system sends the best control command to the MAV (hover, turn left/right, move left/right). We were able to conduct multiple test flights in corridors at speeds of approximately 2 m/s speed without collision, while detecting the vanishing point and doors reliably.

Figure 5 shows two examples of our vanishing point and door detection results under different lighting conditions during MAV flight. The detected door lines are highlighted with colored lines. The probability in the upper left corner is the computed confidence of the detected vanishing point. We observed that the doors close to the camera were detected reliably. As can be seen in these examples, the

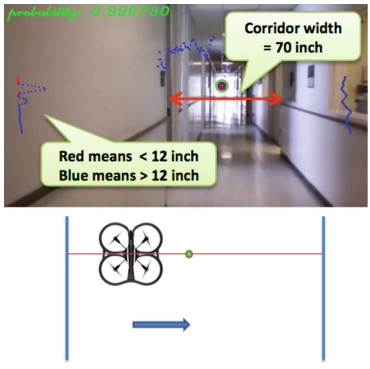

Fig. 6. Sonar-based collision avoidance. When the MAV is close to the wall, the control strategy is to move it away from the wall towards the corridor center.

illumination conditions, background, object scales and viewpoints are quite different, and yet our perception algorithms perform accurately. This is due to the fact our algorithms do not rely on any special assumptions about lightning conditions and viewpoints.

The corridors in our test environment were typically 70 inches wide, while the MAV hull is approximately 20 inches wide. This provides a 25-inch margin from each sidewall when the drone is flying in the center of the corridor. As seen in Fig. 6, if any of the side sonar distance measurements falls below a given threshold (e.g., 12 inches), then our control algorithm commands the drone to move away from the wall. This control process for moving away from the wall has a higher priority than moving forward to the destination point.

5 Conclusion and Discussion

This work describes efficient visual perception algorithms to accurately detect vanishing point and doors in indoor environments. It also describes a control and planning strategy for collision-free navigation. The key advantages of our approach are: (1) fast and efficient perception algorithms that can run onboard in power-constrained applications and platforms, e.g. MAVs; (2) robust perception algorithms that perform accurately in spite of challenging environmental conditions (e.g., changing illumination, viewpoints, appearance) due to the use of invariant geometric features; and (3) a priority-based planner and control strategy that combines outputs from the multimodal perception subsystem to navigate and avoid obstacles.

A next step for us is to port the perception-control coupled system to an onboard processing kit and evaluate its onboard performance for MAV flight. The door detection capability will also be used in a global exploration strategy to move through open doors and progressively explore the environment. We will also investigate the use of additional modalities such as 3D range sensor to improve door detection and detect moving obstacles for reactive control.

Acknowledgement. This material is based upon work supported by Defense Advanced Research Projects Agency under contract numbers W31P4Q-08-C-0264 and HR0011-09-C-0001. The views, opinions, and/or findings contained in this material are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for Public Release, Distribution Unlimited.

References

1. Forney, G.D.: The Viterbi algorithm. Proc. IEEE **61**(3), 268–278 (1973)
2. Yang, X., Tian, Y.: Robust door detection in unfamiliar environments by combining edge and corner features. In: Proceedings of the 3rd Workshop on Computer Vision Applications for the Visually Impaired (CVAVI) (2010)

3. Hensler, J., Blaich, M., Bittel, O.: Real-time door detection based on AdaBoost learning algorithm. In: Gottscheber, A., Obdržálek, D., Schmidt, C. (eds.) EUROBOT 2009. CCIS, vol. 82, pp. 61–73. Springer, Heidelberg (2010)
4. Lee, J.-S., Doh, N.L. Chung, W.K., You, B.-J., Youm, Y.: Door detection algorithm of mobile robot in hallway using PC-camera. In: International Symposium on Automation and Robotics in Construction (2004)
5. Murillo, A.C., Košecká, J., Guerrero, J.J., Sagüés, C.: Visual door detection integrating appearance and shape cues. Robot. Auton. Syst. **56**(6), 512–521 (2008)
6. Parrot AR.Drone 2.0. http://ardrone2.parrot.com/
7. Ranft, B., Dugelay, J.-L., Apvrille, L.: 3D perception for autonomous navigation of a low-cost MAV using minimal landmarks. In: Proceedings of IMAV (2013)
8. Soundararaj, S.P., Sujeeth, A.K., Saxena, A.: Autonomous indoor helicopter flight using a single onboard camera. In: Proceedings of IROS (2009)
9. Eckert, J., German, R., Dressler, F.: On autonomous indoor flights: high-quality real-time localization using low-cost sensors. In: Proceedings of IEEE International Conference on Communications (ICC) (2012)
10. Briod, A., Zufferey, J.-C., Floreano, D.: Optic-flow based control of a 46 g quadrotor. In: Proceedings of IROS (2013)
11. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986)
12. Bills, C., Chen, J., Saxena, A.: Autonomous MAV flight in indoor environments using single image perspective cues. In: Proceedings of ICRA (2011)
13. Brunelli, R.: Template Matching Techniques in Computer Vision: Theory and Practice. Wiley, Hoboken (2009). ISBN 978-0-470-51706-2
14. Lowe, D.: Object recognition from local scale-invariant features. International Conference on Computer Vision, Corfu, Greece, pp. 1150–1157 (1999)
15. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)