2015 International Conference on Unmanned Aircraft Systems (ICUAS)
Denver Marriott Tech Center
Denver, Colorado, USA, June 9-12, 2015

High-speed Vision-based Autonomous Indoor Navigation of a Quadcopter

Adriano Garcia, Edward Mattison and Kanad Ghose
Department of Computer Science, State University of New York, Binghamton, NY 13902

***Abstract*— A monocular vision-based approach to indoor autonomous navigation for an off-the-shelf, low-cost Micro Air Vehicle (MAV) quadcopter is presented. Our approach is fully automated and relies on the extraction and analysis of the visual contours of the surrounding physical environment to successfully steer the MAV in hallways and to turn at intersections. All image analysis and processing necessary for deriving and controlling the flight trajectory take place off-board on a system external to the MAV. This permits the use of sophisticated multithreaded real-time algorithms that do not limit the speed of the drone. Furthermore, due to the elimination of on-board processing and possibly the use of additional sensors for realizing such autonomy, stock drones can be used and flight times realized on a single charge remain unaffected. We describe a prototype implementation on a Parrot AR.Drone quadcopter and initial results for autonomous navigation in a structured indoor environment that establishes the viability of the approach.**

I. INTRODUCTION AND OVERVIEW

Airborne indoor autonomous navigation is an expanding research field that poses several unique challenges not found in outdoor or ground based navigation. The confined nature of GPS-denied indoor environments and relatively higher inertia of the aircraft (compared to ground vehicles) severely limit the margin of error of its movements. In addition, given the Micro Air Vehicle's (MAV) small size and current limitations in battery technology, lift capacity is very restricted and any extra weight diminishes the duration of the flight. As such, we developed a system that overcomes these challenges and limitations by adhering to a strict monocular vision-based autonomous navigation approach where all resource intensive processing takes place off-board and no additional sensors are used.

We address the general problem of designing the control software for a Micro Air Vehicle (MAV) quadcopter to allow for autonomous navigation in unknown structured hallway environments where the edges formed by the intersection of the walls and the floor can be detected by image processing algorithms in order to allow the drone to determine its location with respect to its surroundings. The images are extracted from video captured by a single onboard, forward-facing camera and are processed off-board on a laptop that is static or onboard an autonomous ground vehicle that follows the drone.

Two concrete objectives guided the development of the system. First, the drone needed to consistently and reliably traverse a corridor from end to end without hitting any walls or reducing its top speed. Secondly, the system needed to be able to detect intersections well ahead of time in order to execute stopping and turning maneuvers to take a different path. Accomplishing these two objectives required the development of very precise control algorithms that take into account the aerodynamics of the drone, and used fast, multithreaded image processing to determine the location of the drone with respect to the environment. Our approach does not compromise the speed of the drone, resulting in relatively high average velocities of 1.6 meters per second.

Figure 1. AR.Drone 2.0 and some pertinent Euler angles

A. Quadcopter Platform

The quadcopter used for our prototype implementation is the AR.Drone 2.0 manufactured by Parrot [1] and depicted in Figure 1. Its low cost and wide array of sensors, along with a programmable API makes it a great platform for rapid software development for a variety of applications. The AR.Drone has an Inertial Measurement Unit (IMU) that provides pitch, roll and yaw measurements, a down-facing camera for vision based speed estimates, and an ultrasound telemeter and pressure sensor for altitude measurements at different heights. Additionally, the AR.Drone has a 1 GHz processor and 1 GB of memory that allow it to run firmware for automatic stability control. Combined with the onboard hardware, the firmware enables a high degree of out-of-the-box autonomy where the drone is able to hover in place unaided and successfully execute remotely issued smooth and fairly precise movement commands sent over a wireless link.

The AR.Drone uses an onboard Wi-Fi module to set up an Access Point where a control device such as a smartphone or a laptop can connect to obtain video information from a front facing camera and vehicle state information from the onboard sensors. It also uses this wireless link to receive

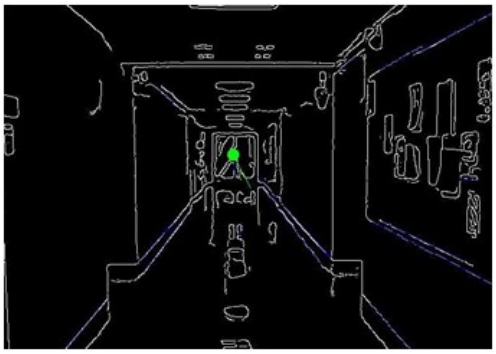

Figure 2. Lines extracted from a hallway image and the vanishing point (green dot)

navigational commands from the off-board control device. For the effort reported here, the control device is a laptop that is static. The laptop can also be located on an autonomous ground vehicle (in development) that follows the drone.

The front facing camera captures video at 30 frames per second and streams it to the laptop for processing. Each frame is 640 pixels wide and 360 pixels high.

B. Off-Board Control of Flight Trajectory

We use an off-board, monocular vision based control approach to control the flight trajectory. Specifically, our solution has two control loops. The first of these loops is internal to the AR.Drone ("internal loop") and uses its sensors to maintain stable flight characteristics. The second control loop ("external loop") is relatively slower and uses the off-board control system to generate the control signals that will affect the drone's flight trajectory. The frames captured by the drone's forward-facing camera are used by the off-board controller to infer the drone's trajectory within the physical environment and generate control signals that maintain the desired flight path.

The delays in the external control loop are as follows:

1) Delay in capturing an image, compressing it and transmitting the frame as part of the video stream.
2) The delay in receiving and extracting an image frame from the video stream sent by the drone on the video channel.
3) The processing latency of the frame and the derivation of the control signals for changing the flight trajectory.
4) The delay in sending the control packet to the drone (on the control channel).
5) The delay of the drone's on-board system in processing the control command and generating the actuation signals.

The sum of these delays determines how quickly the drone can react to external factors and defines the effectiveness of our real-time control system. Estimating these delays accurately has been the subject of technical investigations such as those reported in [2, 3] and is part of our current investigations. We report somewhat coarser estimates later in Section I.E.

Off-board processing for trajectory detection and flight path planning, as well as the derivation of flight control signals offers several advantages. First, sophisticated, multithreaded algorithms can be used to reduce the processing time. Second, computational resources and battery life on the drone are not stressed. Third, the navigation solutions developed can be scaled down and adapted easily to smaller quadcopters.

C. Visual Approach

Our vision-based approach relies on the fact that most indoor environments are structured and their two-dimensional image have well defined geometric patterns that can reveal a rich set of information to assist indoor navigation. The image of a hallway, looking head-on, after line extraction using image processing algorithms [4, 5] is shown in Figure 2. The line-extracted image provides useful information about unobstructed flight paths and are used in our technique as the basis for localization of the MAV with respect to its environment.

The line images are extracted from the video frames provided by the MAV's front camera using the OpenCV implementation of the Canny edge detection algorithm [5]. We follow with a Hough Line Transform [4] that connects the points of local maxima from the Canny processed image in order to detect the lines in the frame.

For the line-extracted hallway image of Figure 2, the lines at the intersection of the walls with the ceilings and the floor converge at an imaginary point, defined in the literature as the vanishing point [6] – [9]. The vanishing point provides a useful basis for estimating structure in 3D environments from 2D images. Our implementation uses the vanishing point as the reference point for straight navigation along hallways, and it also uses the lines that define the vanishing point as special markers for determining proximity to walls.

D. Control Software Architecture

The control software architecture is comprised of three ROS (Robot Operating System) nodes. A master control node runs the image processing algorithms and issues the commands necessary to permit autonomous flight. A controller node decouples the AR.Drone API software from the master control node, and allows the master control node to work independently of the lower level API software. Lastly, an API node handles direct communication with the drone and receives and sends information.

The heart of our implementation is the master control node where all navigation control algorithms are implemented. Depicted in Figure 3, the control node follows a sequence of steps that continuously query its current turning status in order to take appropriate action. The drone can be in three turning states and each state has the following effects:

1) *No intersection detected and not currently turning*: Execute "Fly Straight" control algorithm in order to continue to fly forward along the center line.
2) *Intersection detected:* Execute "Turn right" or "Turn left" maneuvers where the drone starts slowing down in such a way that it stops in the middle of the intersection.
3) *Currently turning at intersection:* Finish turning and ignore other input.

The dataflow model implemented by the ROS nodes is depicted in Figure 4, where the inner data flow of the master

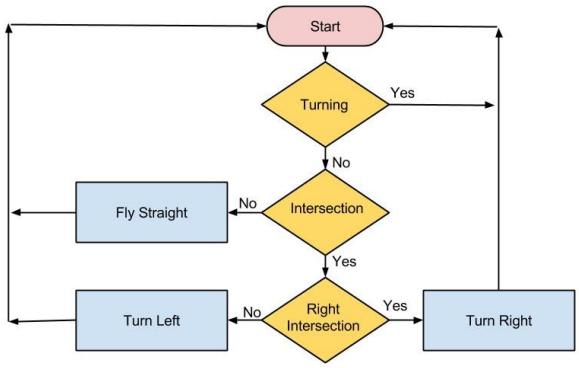

Figure 3. Control loop algorithm

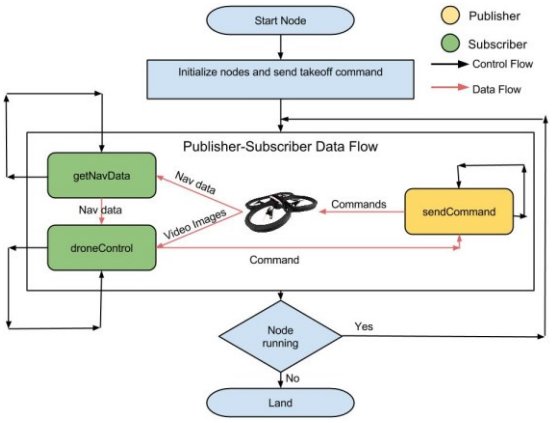

Figure 4. Data and control flow of ROS control node

control node is described. The green colored modules run asynchronously and constantly obtain new data from the drone and the yellow publisher module sends the control commands based on the information obtained from the green modules.

E. External Control Loop Delays

Through very coarse measurements based on the delay in the number of frames from when an action occurred to when it was registered by the off-board processing unit, we determined the delays of Steps 1 and 2 (all Steps refer to those in Sec. I.B) to range between 133 milliseconds (ms) and 166 ms (4 to 5 frame delay). We estimated from our control software that the delay associated with Step 3 is about 10 ms, and measured the delay in transmitting the control packet in Step 4 to the drone as 13 ms. The latency incurred processing the control packet onboard the drone in Step 5 is an unknown value that we assume to be smaller than the 10 ms from Step 3, as Step 3 is comprised of much more resource intensive image processing algorithms. As such, the total external control delay is estimated to range between 156 ms and 189 ms. This delay is anticipated and handled by our control algorithms in order to abate any negative effects on the drone's flight.

The reduced delay in the off-board processing for trajectory detection, path planning and flight control results in an external control loop delay that is dominated by the communication latency. In spite of this communication delay, the fast forward navigation algorithm using a vanishing point and the fast corner detection, along with the additional control for stable flight allows a high flight speed to be maintained.

II. RELATED WORK

The AR.Drone has become a popular platform for autonomous MAV development due to its low cost, flexible open API, and high stability out-of-the-box. In [10] Thomas et al describe the AR.Drone platform in detail and provide an in-depth survey of its many benefits and potential for drone research. In addition, the developers of the AR.Drone explain the technology behind its stability control software in [11].

Autonomous indoor navigation research based on the AR.Drone varies widely. In [12] markers and their 3-D coordinates are used in a 2-D framework to steer an AR.Drone in an enclosed space based on the localization of the markers from the camera image on the drone, using an external, ground-based controller. This work is similar to our work in its use of an AR.Drone and off-board control for navigation. However, our effort does not rely on markers and uses image processing techniques to recognize attributes of the space to steer the drone.

In another AR.Drone-based solution [13], pose estimation using the forward camera-based monocular SLAM, inertial sensing with extended Kalman filtering for data fusion and compensation of communication and processing delays and a PID controller are used to steer the drone along a specified trajectory in a variety of indoor environments. The approach used here, in contrast, uses simpler algorithms to locate open passageways and intersections to navigate the AR.Drone, discovering a relevant trajectory during the flight. Although not described in this paper, we use similar algorithms for dynamic obstacle avoidance.

A vision system based approach to make an AR.Drone follow another drone is described in [14]. The AR.Drone's visual odometry system is used to steer the AR.Drone along a specific trajectory using a fully on-board approach in [15].

The AR.Drone's flight and control characteristic models, and a video stream from an external webcam are used in [16] to steer the drone along a specific trajectory.

A learning system based approach using the drone's monocular vision system to steer an AR.Drone in a wooded environment, avoiding the trees, is presented in [17].

A funnel theory based path estimation and indoor navigation technique for an AR.Drone that uses the vision system, reference images and the recognition of corner features is described in [18]. Our effort uses simpler image processing technique to path discovery and autonomous navigation, relying on the use of the vanishing point for forward steering without requiring any reference images. The vision system of the AR.Drone is used to locate land-mine like objects in the work of [19].

Our approach most closely resembles the approach taken in [6], where the AR.Drone was used to validate vision based

navigation techniques that enable the drone to classify its environment, fly straight, turn at intersections, and fly up stairs. In contrast to our all vision approach, the system of [6] mounted three additional proximity sensors facing sideways and forward to detect walls. Straight navigation is implemented by having the drone follow a vanishing point and using the proximity sensors to avoid hitting walls. The success rate of the system of [6] without hitting walls ranges from 78%, 42%, and 20% on different corridors. Their turning algorithm implementation does not use visual cues but relies solely on proximity sensors to detect an intersection by using the front facing proximity sensor to identify a wall blocking the way. When a wall is detected, it uses the side proximity sensors to detect open space and turn in the direction of the open space until the front proximity sensor clears the wall and reports empty space again. This approach does not address the likely scenario of encountering intersections where there is no wall blocking the drone's path and also limits its speed.

Autonomous indoor navigation techniques on MAVs other than AR.Drones have also been implemented - we now describe some of these efforts in the remainder of this section.

A quadcopter equipped with a laser range finder and an augmented 2-D representation of the space is used for SLAM and autonomous navigation in [20].

A micro-air vehicle incorporating inertial sensors, a camera and on-board laser scanner and algorithms based on pose detection, localization and planning for fully-autonomous multi-floor indoor navigation is described in [21].

A vision system based approach for navigating an ensemble of MAVs and ground vehicles is described in [22]. A linear quadratic tracking based control strategy for a quadcopter is described in [23].

Two cameras and inertial sensing is used for fast, autonomous navigation of a MAV in [24]. A collaborative approach to SLAM and autonomous navigation of a small swarm of MAVs is described in [25].

Research involving high speed indoor drones is somewhat scarcer. The closest such work we were able to identify is presented in [26], where speeds of 2 m/s are achieved indoors by making use of a robust visual-inertial state estimator that allows the drone to know its location with respect to the environment based on visual and IMU data. Two forward-facing fisheye cameras are used as navigational guides. One camera runs at a high frame rate and provides pose estimation and mapping, and the other camera operates at a lower frame rate and compensates for the limitations of a monocular base approach. According to the results provided, high rates of speed of up to 2 m/s were achieved when following an 8 shaped figure, 4 m/s in a straight unobstructed flight, and 1.5 m/s when avoiding two obstacles.

III. QUADCOPTER AERODYNAMIC MODEL

Our aerodynamic model is based on Newtonian physics where the properties of the drone were studied in order to determine the movement commands necessary to keep the drone true to the center line and also allow for controlled stops at intersections. In addition, blowback from the indoor surfaces and inertial drift due to the lack of a strong frictional force in the air mass pose significant challenges for indoor MAV navigation [27] and are successfully mitigated in our model.

Dataset #	Average Velocity (degrees/ ms)	Average Acceleration (degrees / ms^2)
1	0.033789886	0.000074907664
2	0.032798178	0.000131496063
3	0.027307298	0.000063724886
4	0.030062801	0.000087327172
5	0.034627899	0.000088421053
Total Average	0.030989541	0.000074312806

Table I. Example of data used for derivation of drone's constant velocity and acceleration values

The aerodynamic model adheres to a map-less approach where no a priori knowledge of the environment exists and the drone stays as close to the center line as possible in order to avoid collisions with the walls [28]. Image processing algorithms provide the necessary markers to stay on course and detect intersections ahead of time while small but measured movements keep it on track.

A. Yaw Drift Correction

Drift along the yaw angle (as depicted in Figure 1) of the drone plays a significant detrimental role in the navigation of fast moving MAVs. As a result, a vanishing point is used as the external frame of reference necessary to enable the drone to acquire a sense of its relative position with the environment in order to avoid walls and successfully traverse hallways. Our approach relies on applying a corrective rotational force to the drone in such a way that the vanishing point and the yaw angle of the drone align.

Following a vanishing point in a constricted indoor environment requires very precise maneuvers in order to mitigate rotational inertia without any overcorrection. We solve this problem by implementing a Newtonian physics based prediction model that enables the drone to always apply a corrective rotational force long enough to return the yaw angle to alignment, but not long enough to overcompensate.

Finding an acceptable amount of time to apply a corrective rotational force is a two step process. Firstly, as in [10], we observed and deduced that the thrust generated by the four rotors is constant. This permitted us to measure the constant values of the angular velocity w and angular acceleration a as demonstrated in Table I. With these constants, we were able to use the equation $w = w_0 + \alpha t_{stop}$ to derive the constant time $t_{stop} = \frac{(w_0 - w)}{-\alpha}$ that it takes the drone to stop rotating.

The second step is to calculate the overall time that the drone should exert a corrective rotational force, taking into account the time that it takes to come to a complete stop. This is accomplished by the following algorithm:

1) *Determine number of pixels by which the vanishing point is off center:* The video frames captured by the front camera are 640 pixels wide and 360 pixels high. The pixel at the center of this image ("center pixel")

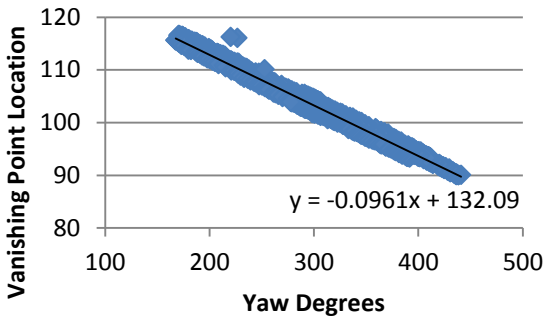

Figure 5. Vanishing point location to yaw degree correlation

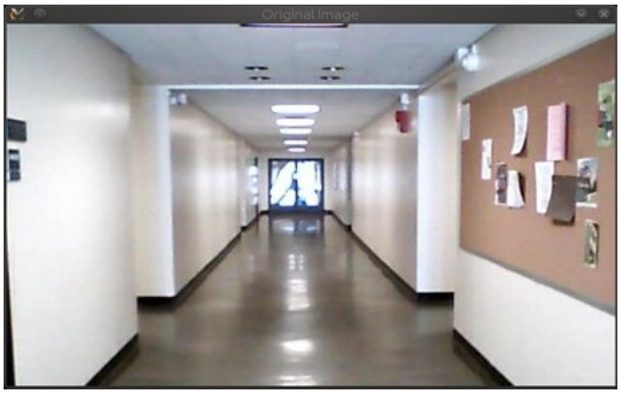

Figure 6. A typical hallway intersection from the drone's POV

represents the horizontal center of the drone and the objective is to keep this as closely aligned with the position of the vanishing point as possible. Calculating the absolute difference between the *x* coordinate of the vanishing point and the center pixel, results in the number of pixels away from the center line ("pixel displacement"). The sign indicates whether a left or right correction is necessary to the drone's yaw angle.

2) *Correlate pixel displacement to angular displacement:* The angular deviation from the gyro-controlled heading and the horizontal location of the vanishing point is related to the pixel displacement obtained in Step 1. Figure 5 demonstrates such correlation where a linear relationship can clearly be seen and used to correlate pixel length to degree displacement.

3) *Calculate time to realign the vanishing point with the center assuming constant acceleration:* The formula for angular displacement with constant angular acceleration: $\theta = w_0 t_{total} + \frac{1}{2}\alpha t_{total}^2$ can be restated as: $t_{total} = \frac{\sqrt{2\times\theta}}{\alpha} - t_{stop}$, giving as a result the amount of time that a corrective rotational force should be applied in order to return the drone's yaw angle to alignment with the center line.

B. Roll Correction

The aerodynamic model for roll correction entails adjusting along the longitudinal *x* axis of the drone (Figure 1) in such a way that a rapid change in horizontal position occurs in order to avoid hitting walls. This compliments yaw angle correction and plays an important role in allowing for high speeds in indoor environments.

Given the low texture features of the floor in our test environment, the vision based velocity estimates described by the makers of the AR.Drone in [11] are not reliable enough to be used in our model. As a result, we devised a roll correction control algorithm that uses the *y* axis pitch angle and forward speed as a guide to how drastic a change in angle is necessary to move away from a wall. Similar to yaw correction, the angular deviation is directly proportional to the length of time that the drone is commanded to roll.

The correction algorithm to determine when the drone should roll to the left in order to avoid a wall on its right side is as follows:

1) While flying too close to any wall:

- If roll angle > R, where R is a threshold value that was empirically set at two degrees, roll toward wall: A roll angle greater than R is too acute and results in overcorrection. Rolling toward the wall for a brief amount of time has the benefit of arresting the sideway momentum and allowing for greater control
- If roll angle <= R: Determine the amount of time to roll away from the wall by the following formula that correlates the speed of the drone and its roll angle to the amount of time it should roll: $Roll\ duration = forward\ velocity - (roll\ angle^2 * K)$, where the constant K was determined empirically to have a value of 5.

2) When drone has returned to center line, continue forward flight.

IV. IMAGE-BASED AUTONOMOUS NAVIGATION

A typical indoor environment (Figure 6) does not immediately convey the quantifiable information necessary for autonomous flight. However, when Canny edge and Hough Line transform image processing algorithms are applied, edges and lines are extracted and leveraged to generate the vanishing point shown in Figure 7.

A. Vanishing Point Identification and Following

Vanishing point identification and following is a critical process in our navigation system, as it provides the necessary reference point for center line navigation. We adopt the MLESAC [29] based vanishing point identification method described in [7] where the number of iterations required to find the vanishing point is greatly reduced by predicting its location based on the idea that its location will not change very drastically from frame to frame.

The identified vanishing point shown in Figure 7 is represented as a vector of *x* and *y* coordinates where *x* is the horizontal distance from the origin (top left corner) and *y* is

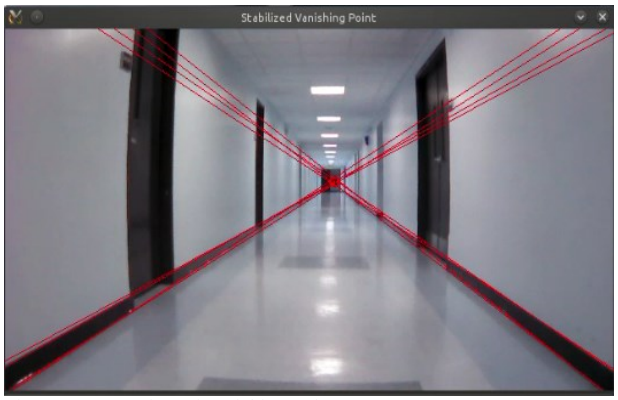

Figure 7. Extraction of vanishing point

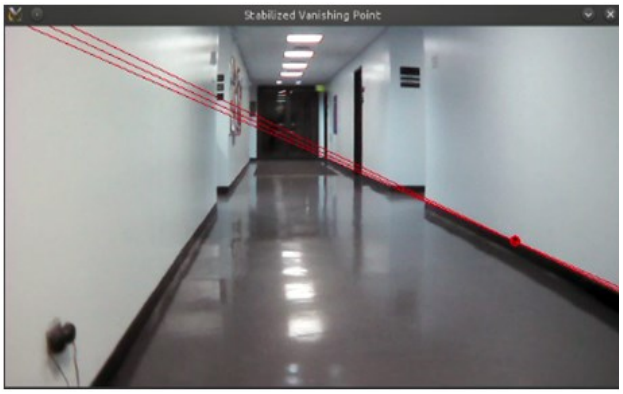

Figure 8. Left diagonal lines disappear as drone approaches left wall and an invalid vanishing point appears on lower right side of frame

the vertical distance. The *y* value is ignored, as the height of the drone remains constant throughout flight.

The vanishing point's *x* coordinate is measured in pixels and it represents the position of the true center of the hallway. Since the width of each frame is 640, the center pixel (320-th pixel on the horizontal line of pixels at the center of the frame) is the relative center of the drone. The absolute value of the difference of this relative center and the vanishing point's *x* coordinate value indicates how far off the center line the drone currently is.

Straight navigation relies on determining the number of pixels off center and using the aerodynamic control algorithm described in section III.A to correlate pixel amount to degrees off center in order to take proper heading corrective action.

B. *Vanishing Point Stabilization*

Although the probabilistic vanishing point identification method described in [7] correctly identifies the location of the vanishing point in most situations, the quick change of scenery, visual clutter, and closeness to a wall sometimes results in erratic vanishing point outliers as depicted in Figure 8, where an invalid vanishing point is detected in the lower right side of the frame. These outliers pose a serious problem for navigation, as they force an incorrect change in the yaw angle of the drone.

The problem of outlier vanishing points is ameliorated by a stabilizing algorithm that filters out the outliers by comparing all new vanishing points against the average of the last ten vanishing points. The stabilizing algorithm works as follows:

1) *Obtain vanishing point:* Use the current video frame to extract the vanishing point.

2) *Compare vanishing point against previous values:* Use an accumulator array to keep track of the locations of the last ten vanishing points in order to be able to determine if the newest vanishing point is an outlier.

3) *Discard outlier:* If the current vanishing point's *x* coordinate value is more than 25% off the center of the camera (in our case more than 80 pixels off the 320 middle), it is an outlier an must be discarded. We rely on the idea that vanishing points don't generally change very drastically from frame to frame as mentioned in [7].

The correctness of our stabilized vanishing point algorithm is verified experimentally where in twenty consecutive trials not a single outlier was accepted.

C. *Visual Roll Correction*

As far as we have been able to ascertain, our method for roll correction has not been implemented in any previous research work and it plays a key role in enabling high rates of speed that average 1.6 meters per second. It relies on using the lines that create the vanishing point to establish whether the drone is getting too close to any of the walls, and therefore take quick corrective actions using the aerodynamic method described in section III.B.

As each line used to determine the vanishing point is extracted, we calculate its slope to determine whether it is negative or positive. A negative slope indicates that the line belongs to a line that is decreasing in slope, meaning that it belongs to the right side of the wall. A positive increasing slope indicates the line belongs to the group of lines on the left side of the wall. Both positive and negative intersecting slope lines are needed to form the vanishing point of the hallway. When only positive or only negative slope lines are present, it can be inferred that the drone is getting too close to a wall. Figure 8 demonstrates this idea where all the positive slope lines have disappeared and only negative slope lines are present, clearly showing that the drone is too close to the left wall. Additionally, it can be seen why the left vanishing point lines have disappeared; their angle is almost 90 degrees and they have been intentionally discounted, as vanishing point lines cannot be vertical.

By counting the number of consecutive negative or positive sloped lines, a threshold can be determined to indicate that a roll corrective action is necessary. We determined experimentally that 30 consecutive negative or positive slopes provided a very reliable threshold for roll correction.

In addition, our implementation takes advantage of the abundant off-board processing power to execute the yaw drift correction algorithm in parallel with the roll correction

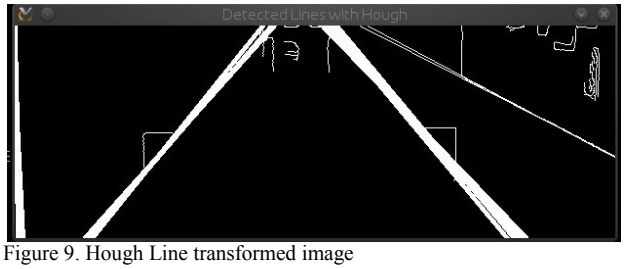

Figure 9. Hough Line transformed image

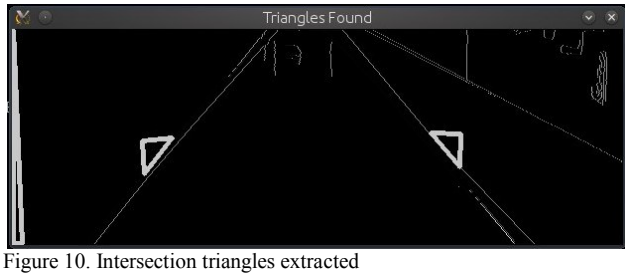

Figure 10. Intersection triangles extracted

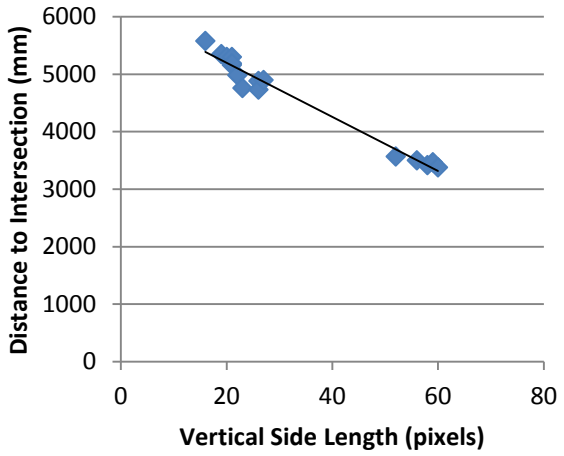

Figure 11. Distance to intersection vs. vertical triangle side size

algorithm, allowing for simultaneous correction of the yaw and roll angles of the drone. This results in a smoother and much faster flight as it eliminates the need to constantly reduce the speed of the drone to adjust either angle.

D. Visual-based Intersection Detection

The first step to successfully turn at an intersection is to anticipate the intersection in order for the drone to have enough time to arrest its momentum and stop as close to the middle of the intersection as possible. Our intersection detection approach works by detecting the right triangles that are formed by the intersecting lines of the floor and the walls.

When a Canny edge detection algorithm and a Hough Line transform algorithm is applied to an approaching intersection frame, two clear geometric patterns emerge (Figure 9). Two right triangles ("intersection triangles") emerge at the left and right intersection. Moreover, these two right triangles are shaped in a very specific manner where their right angle is formed by the intersection of the two walls and the hypotenuse is formed by the vanishing point line at the intersection of the wall and the floor.

The intersection triangles are extracted by executing the following sequence of steps:

1) Remove top half of original image: The top half is not needed and may contain many false positive triangles

2) Apply Canny edge and Hough Line transform algorithms: This obtains the lines of interest from the image produced by Step 1.

3) Extract image contours: Extract all the contours in the image and store them as a vector of points.

4) Find triangles: Go through the vector of contours and eliminate any polynomial shape that does not have exactly three points

5) Eliminate False Positives: False positives are very common since many common objects such as bulletin boards, doors, etc, contain triangular shapes. Eliminate triangles that are too large, too small, or shaped differently than the intersecting triangles depicted in Figure 10.

We found that every time we followed the described intersection triangle extraction steps, we always obtained the triangles as depicted in Figure 10 and we could reliably use this method to detect upcoming intersections.

E. Distance to Intersection

A useful property of the intersection triangles described in the previous section is the fact that the size of their vertical side is not affected by the horizontal position or displacement of the drone. In addition, their vertical size was found to be inversely proportional to the distance to the intersection, permitting the mathematical modeling of the relationship between vertical size and distance to intersection.

We set out to find a function that modeled this relationship by placing the drone on a stand 86 centimeters off the ground that emulated its average flight altitude. The stand was then moved from the back of the hallway to the middle of the intersection while measuring the decreasing distance to the intersection and the increasing size of the vertical triangle side. The resulting measurements were plotted as depicted in Figure 11, resulting in a linear regression line represented by the function $y = -47.148x + 6144.8$ that could be used to find the distance to the intersection when the triangle's vertical size is known.

In implementing obstacle avoidance (not described in this paper), the altitude of the drone changes and the formulas above use an additional parameter to account for scale changes with altitude.

V. STOPPING AND TURNING AT INTERSECTIONS

When an intersection is found using the method described in the previous section, the navigation mode switches from a

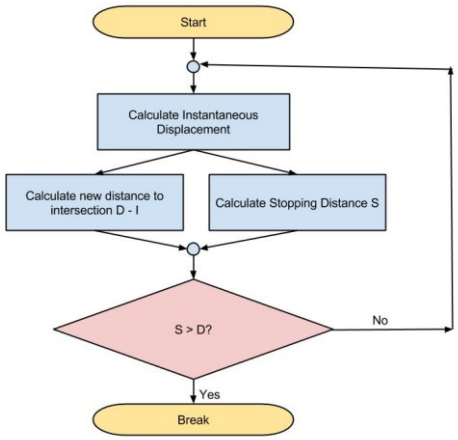

Figure 12. Intersection Negotiation Algorithm

vanishing point following algorithm to an intersection negotiation algorithm. The main steps of the intersection negotiation algorithm are depicted in Figure 12. The basic idea is to continuously calculate the drone's current distance to the intersection and check if the stopping distance is greater. The moment the stopping distance is greater than the distance to the intersection, the drone must start braking in order to avoid overshooting the intersection. Note that some additional information may be needed to decide whether to turn in a specific direction at an intersection or to continue in the straight direction. This information can be programmed separately and can come, for instance, from a rough map available at the control station. For our experiments, the turn was explicitly coded to take place at the first intersection and in a specific direction.

As the intersection negotiating algorithm (Figure 12) shows, three dynamic values for the instantaneous displacement, the instantaneous distance to the intersection, and the stopping distance need to be recalculated approximately every 67 milliseconds as new navigational data is received from the drone:

1) Instantaneous Displacement:

The instantaneous displacement is the value that represents how far the drone has traveled since the last iteration of the loop. We calculate this value as follows:

- At the start of the loop in Figure 12, perform an update to obtain the latest measurements of instantaneous velocity V_i and timestamp T_i.
- At the end of the loop, record the instantaneous velocity V_f and timestamp T_f.
- Calculate instantaneous displacement by using the formula: $d = \frac{(V_f + V_i)}{2} * (T_f - T_i)$.

2) Instantaneous Distance to Intersection

Recalculate the instantaneous distance to the intersection by subtracting the instantaneous displacement from previously calculated instantaneous distance to the intersection.

3) Stopping Distance

The stopping distance of the drone is the distance that the drone will travel due to inertia after stopping procedure is initiated. To stop the drone, it is commanded to pitch backward, redirecting the thrust of the drone backward, opposing the forward momentum of the drone and forcing it to stop.

The formula to calculate the stopping distance was derived as $d = \frac{-(V_i^2)}{(2a)}$ where V_i is the instantaneous velocity and a is the deceleration rate of the drone, which we experimentally measured to be -1531.05 mm/s^2.

VI. Results

We fully met our objectives of achieving consistent straight autonomous navigation at high speeds and recognition of an intersection and turning into the orthogonal hallway. In our main test hallway we obtained a 100% success rate with no collisions with the walls. In our intersection detection and turning tests, we obtained eight consecutive successful intersection identifications and turns from different starting points.

A. Straight Hallway Navigation Results

Testing the drone's straight autonomous navigation ability consisted of having the drone fly straight from one end of the hallway to the other noting the number of collisions with walls and its success rate. We define success as traversing the hallway from its beginning to end.

Three different hallways of different dimensions were used. In our main test hallway (Hallway 1 in Table II) we achieved a 100% success rate with no wall collisions as shown in Table II. Furthermore, we also tested extreme scenarios where the drone was started 30 cm from the wall and 30 cm from the middle of the hallway to test the drone's ability to hover from side to side in order to return to the middle of the intersection and find and follow the vanishing point.

In Hallway 3 we obtained very positive results as well, even though the drone hit the wall in four tests. Hitting the wall was the result of a long stretch of empty wall space on the left side where no lines could be detected easily due to the lack of sufficient contrast in the image for estimating the vanishing point. (Contrast adjustments on the image is possible but were not attempted). However, testament to the consistency and resiliency of our solution, the drone was always able correct itself and resume normal flight.

Autonomous straight flight in Hallway 2 was much more challenging due to the fact that the margin for error is drastically reduced in the narrower hallway and blowback from the wall is stronger. The success rate and wall hit rate in Hallway 2 were 50%. Every time the drone hit the hall, it was unable to recuperate and continue normal flight due to the difficulty in finding the vanishing point again. We are currently working on improvements to our control algorithms to allow for smoother and more measured control in narrower hallways.

Hallway	Length (m)	Width (m)	Wall collisions	Success rate
1	27.95	2.36	0 / 20	20 / 20
2	17.37	1.75	10 / 20	10 / 20
3	53.04	2.36	4 / 20	20 / 20

Table II. Straight Navigation Results

Test #	Dist. to Intersection (meters)	Intersection detection	Turn	Turn Success Rate (%)
1	10.97	2 / 2	2 / 2	100
2	9.75	2 / 2	2 / 2	100
3	7.62	2 / 2	2 / 2	100
4	6.09	2 / 2	2 / 2	100
5	5.49	0 / 2	0 / 2	0

Table III. Intersection Detection and Turning Results

B. *Intersection Detection and Turning*

Intersection turning tests were performed in Hallway 1 with dimensions described in Table II. To test the drone's ability to stop and turn at intersections, we placed the drone at different distances from the intersection to evaluate its ability to adjust to different scenarios where the initial velocity, vertical triangle side size, stopping distance, and stopping time would all be different. We ran each test twice and regarded a successful turn as one where the drone was able to detect the intersection, stop at the intersection, turn 90 degrees in the direction of the new hallway, and follow the new vanishing point.

Based on the results shown in Table III, it can be seen that the results strongly validate our approach. The only test scenario that failed occurred when the drone was placed too close to the intersection and could not detect it.

C. *Flight Videos of Prototype System*

The video footage was recorded showing straight navigation and intersection detection and turning as described in this paper is available at:

https://vimeo.com/127293061

The video stream from the drone and the corresponding line segment images that show the derived stable vanishing point, as well as the image artifacts used for detecting an intersection are shown side-by-side in some of the clips.

D. *Summary of Experimental Results*

The experimental results demonstrate that the monocular vision based autonomous navigation system presented in this paper met its objectives fairly successfully. In particular, the results show that the proposed approach is viable and amenable to off-the-shelf, low-cost quadcopters, requiring no extra sensors and permitting high speed autonomous flights in indoor hallways. Improvements are certainly possible to have a perfect or almost perfect success rate in all cases.

VII. Conclusion and Improvements

A completely autonomous flight control system, including aerodynamic and monocular vision based control algorithms, was presented. This system allows for the high-speed navigation of a quadcopter in a constrained indoor hallway environment.

Two image-based trajectory control methods were introduced to enable quick roll correction to avoid collisions with walls, and detect and calculate distance to intersections for autonomous intersection turning. Strong experimental results validate the correctness and viability of our methods and provide the foundation for further improvements and advancement in the field of autonomous indoor navigation of MAVs.

Our ongoing research aims to continue our vision based approach to expand the drone's capabilities to include obstacle avoidance, going up and down staircases, and improvements for stable, autonomous flights in a wider variety of indoor environments through the use of statistical machine learning methods. An autonomous ground vehicle that carries the control laptop and follows the drone is under development, with the ultimate goal of exploring and mapping unknown indoor environments.

ACKNOWLEDGEMENTS

We gratefully acknowledge and thank the open source community for making the presented work possible. This includes the teams behind the Linux ecosystem, ROS, OpenCV, and the AR.Drone developer community. This work was supported in part by the NSF through award Nos. 0958501 and 1040666.

REFERENCES

[1] Parrot Inc., AR.Drone SDK 2.0 Developer Guide. Parrot Inc. Web. https://projects.ardrone.org/projects/show/ardrone-api

[2] Mellado-Bataller, Ignacio, et al. "Rapid prototyping framework for visual control of autonomous micro aerial vehicles." *Intelligent Autonomous Systems 12*. Springer Berlin Heidelberg, 2013. 487-499.

[3] Engel, Jakob, Jürgen Sturm, and Daniel Cremers. "Scale-aware navigation of a low-cost quadrocopter with a monocular camera." *Robotics and Autonomous Systems* 62.11 (2014): 1646-1656.

[4] Duda, Richard O., and Peter E. Hart. "Use of the Hough transformation to detect lines and curves in pictures." Communications of the ACM 15.1 (1972): 11-15.

[5] Canny, John. "A computational approach to edge detection." Pattern Analysis and Machine Intelligence, IEEE Transactions on 6 (1986): 679-698.

[6] Bills, C.; Chen, J.; Saxena, A., "Autonomous MAV flight in indoor environments using single image perspective cues," Robotics and Automation (ICRA), 2011 IEEE International Conference on , vol., no., pp.5776,5783, 9-13 May 2011

[7] Marcos Nieto and Luis Salgado. "Real-time robust estimation of vanishing points through nonlinear optimization", Proc. SPIE 7724, Real-Time Image and Video Processing 2010, 772402 (May 04, 2010).

[8] Barnard, Stephen T. "Interpreting perspective images." Artificial intelligence 21.4 (1983): 435-462.

[9] McLean, G. F., and D. Kotturi. "Vanishing point detection by line clustering." Pattern Analysis and Machine Intelligence, IEEE Transactions on 17.11 (1995): 1090-1095.
[10] T Krajník, V Vonásek, D Fišer, J Faigl. "AR-drone as a platform for robotic research and education." Research and Education in Robotics-EUROBOT 2011, 172-186.
[11] Bristeau, Pierre-Jean, et al. "The navigation and control technology inside the ar. drone micro uav." *18th IFAC World Congress*. Vol. 18. No. 1. 2011.
[12] Santana, Lucas Vago, et al. "A trajectory tracking and 3D positioning controller for the AR. Drone quadrotor." Unmanned Aircraft Systems (ICUAS), 2014 International Conference on. IEEE, 2014.
[13] Engel, Jakob, Jürgen Sturm, and Daniel Cremers. "Accurate figure flying with a quadrocopter using onboard visual and inertial sensing." IMU 320 (2012): 240.
[14] Jimenez Lugo, J.; Masselli, A.; Zell, A., "Following a quadrotor with another quadrotor using onboard vision," Mobile Robots (ECMR), 2013 European Conference on , vol., no., pp.26,31, 25-27 Sept. 2013
[15] Jimenez Lugo, J.; Zell, A., "Framework for autonomous onboard navigation with the AR.Drone," Unmanned Aircraft Systems (ICUAS), 2013 International Conference on , vol., no., pp.575,583, 28-31 May 2013
[16] Hernandez, Andres, et al. "Identification and path following control of an AR. Drone quadrotor." System Theory, Control and Computing (ICSTCC), 2013 17th International Conference. IEEE, 2013.
[17] Ross, Stéphane, et al. "Learning monocular reactive uav control in cluttered natural environments." Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013.
[18] Nguyen, T.; Mann, G.K.I.; Gosine, R.G., "Vision-based qualitative path-following control of quadrotor aerial vehicle," Unmanned Aircraft Systems (ICUAS), 2014 International Conference on , vol., no., pp.412,417, 27-30 May 2014.
[19] Rodriguez, J.; Castiblanco, C.; Mondragon, I.; Colorado, J., "Low-cost quadrotor applied for visual detection of landmine-like objects," Unmanned Aircraft Systems (ICUAS), 2014 International Conference on , vol., no., pp.83,88, 27-30 May 2014.
[20] S. Grzonka, G. Grisetti and W. Burgard. "A Fully-Autonomous Indoor Quadcopter." IEEE Trams. On Robotics, Vol. 28(1), Feb. 2012.
[21] Shen, Shaojie, Nathan Michael, and Vijay Kumar. "Autonomous multi-floor indoor navigation with a computationally constrained MAV." Robotics and automation (ICRA), 2011 IEEE international conference on. IEEE, 2011.
[22] Saska, M.; Krajnik, T.; Vonasek, V.; Vanek, P.; Preucil, L., "Navigation, localization and stabilization of formations of unmanned aerial and ground vehicles," Unmanned Aircraft Systems (ICUAS), 2013 International Conference on , vol., no., pp.831,840, 28-31 May 2013.
[23] Suicmez, E.C.; Kutay, A.T., "Optimal path tracking control of a quadrotor UAV," Unmanned Aircraft Systems (ICUAS), 2014 International Conference on , vol., no., pp.115,125, 27-30 May 2014.
[24] Shen, S.; Mulgaonkar, Y.; Michael, N.; Kumar, V., "Vision-based state estimation for autonomous rotorcraft MAVs in complex environments," Robotics and Automation (ICRA), 2013 IEEE International Conference on , vol., no., pp.1758,1764, 6-10 May 2013.
[25] Scaramuzza, D.; Achtelik, M.C.; Doitsidis, L.; Friedrich, F.; Kosmatopoulos, E.; Martinelli, A.; Achtelik, M.W.; Chli, M.; Chatzichristofis, S.; Kneip, L.; Gurdan, D.; Heng, L.; Gim Hee Lee; Lynen, S.; Pollefeys, M.; Renzaglia, A.; Siegwart, R.; Stumpf, J.C.; Tanskanen, P.; Troiani, C.; Weiss, S.; Meier, L., "Vision-Controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-Denied Environments," Robotics & Automation Magazine, IEEE , vol.21, no.3, pp.26,40, Sept. 2014.
[26] Shen, Shaojie, et al. "Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor." Robotics: Science and Systems. 2013.
[27] Mahony, R.; Kumar, V.; Corke, P., "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," Robotics & Automation Magazine, IEEE , vol.19, no.3, pp.20,32, Sept. 2012.
[28] Saitoh, Takeshi, Naoya Tada, and Ryosuke Konishi. *Indoor mobile robot navigation by center following based on monocular vision*. INTECH Open Access Publisher, 2008.
[29] Torr, Philip HS, and Andrew Zisserman. "MLESAC: A new robust estimator with application to estimating image geometry." *Computer Vision and Image Understanding* 78.1 (2000): 138-156.