Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.
Vision-Based Guidance and Control of a Hovering Vehicle in Unknown, GPS-denied Environments

Spencer Ahrens, Daniel Levine, Gregory Andrews, and Jonathan P. How

Abstract—This paper describes the system architecture and core algorithms for a quadrotor helicopter that uses vision data to navigate an unknown, indoor, GPS-denied environment. Without external sensing, an estimation system that relies only on integrating inertial data will have rapidly drifting position estimates. Micro aerial vehicles (MAVs) are stringently weight-constrained, leaving little margin for additional sensors beyond the mission payload. The approach taken in this paper is to introduce an architecture that exploits a common mission payload, namely a video camera, as a dual-use sensor to aid in navigation. Several core algorithms, including a fast environment mapper and a novel heuristic for obstacle avoidance, are also presented. Finally, drift-free hover and obstacle avoidance flight tests in a controlled environment are presented and analyzed.

I. INTRODUCTION

Recent warfighting efforts have focused increasingly on operations in dense, urban environments where GPS interference, whether unintentional or malicious, is to be expected. Simultaneously, unmanned aerial systems (UASs) have seen increasing use in military conflicts as a means to both reducing casualties and providing critical data to the warfighter. However, operations in indoor environments require the vehicle to autonomously maneuver through openings, down hallways, and around obstacles, all necessitating highly accurate position estimation and environment mapping.

While many ground vehicle solutions exist that perform autonomous indoor operations, these vehicles typically utilize multiple localization sensors at the expense of weight. UASs and especially micro aerial vehicles (MAVs) are stringently weight-constrained, with mass adversely affecting agility, endurance, and range. Yet by exploiting for the purpose of navigating through a GPS-denied environment the video feed already onboard a UAS, the camera becomes a dual-use sensor.

There are MAV and hover-capable MAV (HMAV) systems currently under development that address the autonomous flight problem for indoor, GPS-denied, or unknown environments. Forward flight MAVs that use optical flow for obstacle avoidance have demonstrated robust flight stability and collision avoidance in small flight spaces, however, they neither map the 3D environment nor accurately estimate position [1], [2], [3]

Refs. [4] and [5] present quadrotor MAV solutions that utilize downward-looking cameras to enable drift-free autonomous hover. Although very capable in the hands of a human pilot, these solutions: are incapable of estimating absolute position while traversing unknown environments; cannot map obstacles; and, thus, cannot navigate unknown environments autonomously.

Ref. [6] demonstrates the utility of LIDAR onboard HMAVs for position estimation in indoor environments. These systems show promise for HMAV development with the continuing miniaturization of LIDAR modules, yet they currently require prior maps of the environment to enable registration of measurements, and position estimates are too noisy to accurately estimate velocity for effective control feedback. Ref. [7] demonstrates accurate simultaneous localization and mapping of corner features, although the methods used are specific to unobstructed hallways and are not formulated for absolute position estimation. Given this prior work, the primary contributions of this paper are the system architecture and key supporting algorithms for vision-based guidance and control of an HMAV in unknown, indoor environments [8].

II. HARDWARE ARCHITECTURE

The basic hardware structure is outlined in Fig. 1. The airborne system consists of the vehicle, the wireless camera, and reflective infrared markers (dots). The conceptually “onboard” system is composed of a regular desktop computer, a Wi-Fi router for communication with the onboard camera, and a Vicon motion capture system for emulating inertial data. The base station is composed simply of a separate desktop computer workstation. For ease of prototyping, and in an effort to eschew the development of custom vehicles and electronics, as much computation is done offboard as possible. At the same time, extensive process timing analysis is done in order to analyze the feasibility of the core algorithms to ultimately run in real-time on an embedded system. The airborne system and the emulated onboard system are described in detail in the following sections.

Fig. 1. Diagram of the hardware architecture.
A. Vehicle

The HMAV chosen for this research is the Hummingbird quadrotor, which is outfitted with an embedded electronics suite, from Ascending Technologies [9], [10]. An onboard 8-bit microcontroller samples and fuses the data of a 6-axis IMU at 1 kHz to generate a tilt estimate used for onboard tilt stabilization. The digital communication channel (when connected, for example, with an xBee-Pro digital radio) allows external access to the IMU readings and tilt estimates. Any external controller need only specify the desired tilt angle; the vehicle, which receives control signals via a standard 72 MHz R/C receiver, will track this reference with high performance.

B. Imaging Sensor

The imaging sensor selected is a Wi-Fi enabled Panasonic BL-C131A network camera, which hosts 320x240 pixel images that are retrieved at 15–20 Hz through the network as JPEG files and loaded into the image processing system. The 1mm diameter lens has a 41° vertical and 53° horizontal FOV.

Experience has shown that digital wireless cameras are better suited for machine vision applications than their analog counterparts, which suffer indoors from multi-path interference and thereby produce images with distortions and lines of static [11]. The sensor used in this experiment produces images with JPEG compression artifacts that degrade image quality yet are consistent enough to not disturb frame-to-frame processing routines.

C. RAVEN and Emulated Inertial Data

The MIT Real-time indoor Autonomous Vehicle test Environment (RAVEN) [12], [13] provides the hardware foundation for the developments presented. RAVEN combines the tracking capabilities of a Vicon-MX camera system [14] with R/C hardware and an array of networked servers, comprising an ideal rapid-development environment.

The estimation filter of the conceptually onboard system takes IMU data and camera images as inputs. Although the vehicle is equipped with a digital radio to access IMU data, communication dropouts and transmission corruption severely impair the ability of the estimation filter to accurately estimate vehicle ego-motion. Therefore RAVEN is used to emulate inertial data in all flight tests presented. The outputs of RAVEN’s Vicon system are the vehicle position and an attitude quaternion, each of which is passed through an Extended Kalman Filter (EKF) to emulate accelerometer and gyroscope data, respectively.

Figs. 2 and 3 compare the hardware IMU data to filtered Vicon data and indicate the former to be less reliable than the latter, due to the existence of a wireless data link and as typified by the spikes and dropouts in Fig. 2. Despite this, both the hardware and filtered Vicon data have similar nominal behavior. The root mean square (RMS) noise levels of the hardware IMU data are 0.028 m/s² and 0.89 deg/s; the Vicon RMS are slightly higher, at 0.037 m/s² and 0.96 deg/s.

Filtering also incurs an additional lag over hardware data, e.g., 40 ms for gyroscope data. Discretization of the IMU data, clearly shown by the wide gaps between bars in Fig. 3, is due to the fixed precision of the embedded electronics, which evaluates to 0.020 m/s² and 0.61 deg/s in physical terms. However, because the magnitudes of the RMS noise levels and the operational values are larger than the respective fixed precisions, the discretization should not significantly affect the performance of the filter.

Because of the close match of noise characteristics and knowledge of the penalties incurred by using filtered Vicon data, it is expected that the existing ego-motion estimation system likely would operate with similar, if not higher, performance when implemented on an embedded system. Finally, because Vicon has sub-millimeter position accuracy, its unfiltered data can be used off-line as truth for the purposes of measuring estimation errors.

III. IMAGING, ESTIMATION, & MAPPING

A. Image Processing

Visual perception is essential to system performance, as it remains the vehicle’s exclusive capability in sensing the environment and is a direct function of the absolute position and orientation. Conversely, inertial data only provide an indication of rates and tilt.

The primary tasks of image processing are the detection and maintenance of features of interest, or points in space that correspond to locations in the images. To this end, low-level image processing technologies of the Open Computer...
Vision Library (OpenCV) [15] are utilized. The Shi-Tomasi “good features to track” detector [18] finds points with large eigenvalues of the image gradient in multiple directions. In order to achieve feature spread across the image, which is advantageous for the purposes of navigation (i.e., yield a low position dilution of precision [20]), a mask is erected in the vicinity of existing features, within which new features are not detected. The pyramidal Lucas-Kanade optical flow calculator [16], [17] compares pixel patches in successive images in an attempt to accurately maintain the locations of features detected in previous images.

Once tracking is performed, the remaining features are scanned; those that are clumped together, have jumped, or have poorly converging 3D position estimates are discarded and replaced in subsequent calls to the detector. The ultimate result of feature tracking is the delivery of feature locations to the estimation filter and the fast environment mapper, which are discussed next.

B. Ego-Motion Estimation

The process for ego-motion estimation was developed in collaboration with Draper Laboratory [19] building in part on [21], [22]. The EKF used in this implementation accounts for: the position, velocity, and attitude of the vehicle; biases in all axes of the accelerometer and gyroscope; and position of initial sighting, horizontal bearing, vertical bearing, and range of all features.

C. Fast Environment Mapping

While the ego-motion estimation filter explicitly estimates feature locations, it is computationally expensive; thus, only a small number of features can be tracked at once. An alternative method employs a simple least-squares pseudo-intersection algorithm, is well suited for mapping many features, and gives a more complete view of the world than that given by the features in the estimation filter.

Fig. 4 shows the formulation for optimally estimating the pseudo-intersection point $\mathbf{q}_{i,k}$ of obstacle $i$ at time $k$. In the figure, $\mathbf{x}_i$ is the estimated vehicle position, $\mathbf{\hat{o}}_{i,k}$ is the corresponding unit vector bearing estimate from the vehicle to feature $i$, and $h_{i,k}$ is the minimum distance from $\mathbf{q}_{i,k}$ to the estimated vision ray. The two previous bearing estimates at times $k-1$ and $k-2$ show the approximate nature of the intersection.

A simple least-squares algorithm efficiently computes the best-fit intersection point $\mathbf{q}_{i,k}$ by minimizing $\sum h^2$. Originally formulated by Bethke in [23] for simultaneous measurements from multiple vehicles tracking the same target, the algorithm is adapted here to the single vehicle case, where each measurement $i$ is taken at a different point in time. Because the vehicle position estimate tends to drift, an exponential decay of parameter $\alpha$ that slowly attenuates the contributions of older measurements can be incorporated into the transition.

The recursive update equations for obstacle $i$ at the current timestep $k$, with $\mathcal{A}_{i,0}$ and $\mathbf{b}_{i,0}$ initialized to zero, are

$$ \mathcal{A}_{i,k} = \alpha \mathcal{A}_{i,k-1} + (I - \mathbf{\hat{o}}_i \mathbf{\hat{o}}_i^T) \tag{1} $$

$$ \mathbf{b}_{i,k} = \alpha \mathbf{b}_{i,k-1} + (\mathbf{x}_k - (\mathbf{x}_k \cdot \mathbf{\hat{o}}_i) \cdot \mathbf{\hat{o}}_i) \tag{2} $$

$$ \mathbf{q}_{i,k} = \mathcal{A}_{i,k}^{-1} \mathbf{b}_{i,k}. \tag{3} $$

For each feature, we need only store the 3x3 $\mathcal{A}$ matrix and 3-element $\mathbf{b}$ vector and update them with a constant time expression, independent of the number of measurements we have accumulated over time. We can also express each feature’s confidence as $c = \Sigma_{j=1}^{i} ||\Delta \mathbf{x}_j||/||\Delta \mathbf{q}||$ where the estimate jitter $||\Delta \mathbf{q}||$ is simply the magnitude of the difference between the current and previous estimates $\mathbf{q}$. Taking the distance moved since the last update $||\Delta \mathbf{x}_i||$ as approximately $||\mathbf{v} \Delta t||$, the confidence can be recursively updated as

$$ \tilde{c}_{i,k} = \tilde{c}_{i,k-1} + ||\mathbf{v} \Delta t|| \Rightarrow c_{i,k} = \frac{\tilde{c}_{i,k}}{||\mathbf{q}_{i,k} - \mathbf{q}_{i,k-1}||}. \tag{4} $$

Once computed, these confidences determine both whether a feature is used for obstacle avoidance and (in the case of sufficiently low confidence) whether that feature is due to be culled from the map should its confidence not improve during a predefined grace period.

D. Flight Control Issues

The primary difficulty with vision-in-the-loop estimation is that estimate errors decrease control performance, generally via oscillations. These oscillations cause the camera view to change constantly, which forces features to go out of view more often, reducing their persistence. With low-light indoor conditions and low camera quality, the high rate of movement also causes the images to blur. Decreased vision fidelity begets further estimate degradation, and a positive feedback loop is formed as the control performance and estimation subsequently decline.

IV. OBSTACLE AVOIDANCE

A lightweight reactive obstacle avoidance heuristic for the purpose of indoor navigation in complex, unknown environments is now presented. Fast, effective, and natively three-dimensional, the heuristic is easily tractable onboard existing micro flight controllers. As the formulas are of closed form, sampling, which typically requires compromise between completeness and speed (especially when working in three dimensions [24]), is not necessary.

The algorithm takes as inputs the environment map generated using the extended fast-mapping technique (Section III-C), the goal position, and the vehicle’s reference state; it generates as an output a reactive acceleration component $\mathbf{a}_r$. By utilizing the vehicle’s reference state rather than its
estimated state, the algorithm becomes deterministic given a
goal and a set of obstacle locations. Thus, trajectory generation
is decoupled from control and estimation performance; such
coupling would otherwise cause undesirable feedback.
The same methodology was successfully employed by MIT
in the DARPA Urban Challenge [25].

Fig. 5 shows a 2D cutaway diagram of a typical obstacle-avoiding trajectory. The vector from the vehicle at \( x \) to the
goal is \( g \) (normalized to \( \hat{g} \)), and the vector from the vehicle to the
obstacle \( i \) with position estimate \( q_i \) is \( o_i \). The \( i \)th cross-goal vector \( d_i \) is the shortest segment from obstacle \( i \)
to \( g \), such that
\[
d_i = (\hat{g} \cdot o_i) \cdot \hat{g} - o_i \quad \text{with} \quad o_i = q_i - x.
\]

The avoidance buffer of an obstacle \( r_{buf} \), within which the vehicle will try to reverse itself, subsumes the vehicles radius
\( r_{veh} \). If \( \exists i \) such that \( r_{veh} < \|q_i - x\| < r_{buf} \), the vehicle has not crashed but is in the avoidance buffer of a feature,
the reactive acceleration can be immediately set to \( a_r = -a_{max} o_i \), where \( a_{max} \) is a maximum reactive acceleration
magnitude.

The distance \( x_{stop} \) that vehicle will travel towards an
object given the current velocity \( v \) is calculated from simple
kinematics as \( x_{stop} = (v \cdot \hat{o}_i)^2/(2a_{max}) \). At time \( k \), for
the purposes of navigating between \( x_k \) and \( x_{k+1} + g \), we consider
only those obstacles that: have a confidence \( c_i \) above a set
threshold; are within a cylindrical “corridor” of \( g \); are in front
of the vehicle; and are closer than the goal to the vehicle.
Formally, we express the “avoidance criteria,” respectively,
for obstacle \( i \) as
\[
\begin{align*}
c_i &> c_{min} \\
\|d_i\| &< 2(r_{buf} + x_{stop}) \\
\|o_i\| &< \|g\| \\
o_i \cdot g &> 0
\end{align*}
\]
(6)

For an obstacle that meets the avoidance criteria, its contribution
to the reactive acceleration direction vector \( a_i \) is composed of its cross-goal vector \( d_i \), which by construction
is orthogonal to \( g \), and is inversely weighted by the time-to-
impact \( t_i \). If there are \( n \) obstacles that meet the avoidance
criteria, the final reactive acceleration \( a_r \) is aligned with
\( a_i \) and has a magnitude proportional to the velocity to the
nearest obstacle and inversely proportional to an acceleration
time constant \( \tau_r \). More compactly,
\[
t_i = \frac{\|o_i\|}{\hat{a}_i \cdot v} \quad \text{if} \quad \|o_i\| < \|g\|
\]
(7)
\[
a_{c} = \sum_{i=1}^{n} \frac{\|d_i\| \cdot d_i}{t_i} = \left[ \sum_{i=1}^{n} \frac{d_i}{t_i} \right] \quad \text{if} \quad \|o_i\| > \|g\|
\]
(8)
\[
a_r = 1.5 \frac{\|o_i\|}{t_j \tau_r} \hat{a}_o, \quad \text{with} \quad j = \arg\min_{i} t_i.
\]
(9)

This heuristic generates a reference reactive acceleration
\( a_r \) that can be combined with a goal-seeking acceleration
\( a_g \) and an orbital cancelation acceleration \( a_c \perp a_g \) such that
\( a = a_g + a_c + a_r \). The orbital cancelation term can very
simply be found to be
\[
a_c = \frac{\hat{g} \cdot (v \cdot \hat{g}) - v}{\tau_c}
\]
(10)
for some acceleration time constant \( \tau_c \). While \( a_g \) is always
along \( \hat{g} \), the scheduling of its magnitude is less trivial and is
left to the implementer. Note that in the case of an emergency
reverse, the vehicle can use just reactive acceleration references
and impose \( a_c = a_g = 0 \). A simulation of randomly
generated obstacles (merely for ease of presentation, in the
plane \( z = 0 \)) is shown in Fig. 6.

V. FLIGHT TEST RESULTS

A. Drift-Free Hover

When hovering, the vehicle maintains an almost fixed
point of view, maximizing feature persistence and minimizing
drift. Fig. 7 shows data from a typical hover using the
vision-based state estimate for control feedback. Because the
vehicle remains in the same approximate location, no features
are lost, and, thus, the average drift for the 96 seconds the
vehicle is commanded to hover is negligible. The RMS of the
error in the inertial X axis, for example, is 7.61 centimeters.

By contrast, if vision data is not incorporated and the
estimator must rely strictly on the IMU, the position estimate
quickly diverges (i.e., \( |x| \to \infty \)) while the vehicle is still
on the ground and waiting to take off. This is result of the
estimator performing a double integration on biases and noise
in the accelerometer measurements.
B. Vision-Based Control and Obstacle Avoidance

As can be seen in Fig. 8, the environment was set up with the pole, a bar stool, and additional clutter in the center of the room, leaving enough space for the vehicle to maneuver, but with a clear and complex obstacle set blocking the direct path to the goal.

Fig. 9(a) shows two plots from the top (x,y) and side (z,y) of a successful obstacle avoidance flight. The green circle is the take-off point, the crosses are the landing points, and the purple spots represent a snapshot in time of the features used for obstacle avoidance; a cylinder and a cuboid, representing the obstacles, have been superimposed in the operator’s view of Fig. 9(b) for demonstrative purpose. As the vehicle attempted to move toward the goal, the obstacle avoidance commanded to fly the vehicle up, around the pole to the left, over the stool, and down to a smooth touchdown near the goal.

It is of interest to note that during the 5 m flight, the position estimate drifted by approximately 1 m, but the vast majority of error was accumulated just as the vehicle passed over the stool, corresponding with the discarding of all features associated with the obstacle. This would indicate that future trajectory generators should allow the camera to be pointed along a bearing other than the direction of movement, as was imposed in this experiment.

Fig. 10 shows orthographic and operator views of a test flight with the same starting conditions and goal state as with the test previously discussed. However, due to a difference in absolute position at the time the waypoint command was given, the navigation system plans a trajectory to the East, around the pole, avoiding the stool entirely. The data visible in the figure is a clear example of the complimentary offset errors that develop in both the feature estimates and the estimated vehicle position.

VI. Conclusion

This paper has focused on the higher level system architecture and key supporting algorithms for the guidance and control of an HMAV in unknown, GPS-denied environments. The concepts and methodologies for image processing are discussed, and the implementations of the environmental mapping and reactive obstacle avoidance algorithms are presented. Flight results of the autonomous system are analyzed, demonstrating the viability of this solution to achieve its objectives in real environments on physical hardware. Although developed in a testbed for ease of prototyping, many of the issues involving the transition to embedded flight systems are directly addressed, with promising conclusions for the ultimate real-world implementation of these contributions.

REFERENCES

Fig. 9. Flight over the stool from three perspectives.

Fig. 10. Flight around pole from three perspectives.


