Camera Based Localization for Autonomous UAV Formation Flight

Zouhair Mahboubi *
Zico Kolter †
Tao Wang ‡
Geoffrey Bower §
Andrew Y. Ng ¶

This work considers the task of accurate in-air localization for multiple unmanned or autonomous aerial vehicles flying in close formation. The paper describes our experimental setup using two small UAVs and the details of the localization algorithm. The algorithm was implemented on two low-cost, electric powered, remote control aircraft with wing spans of approximately 2 meters. Our control software, running on an onboard x86 CPU, uses LQG control (an LQR controller coupled with an EKF state estimator) and a linearized state space model to control both aircraft to fly synchronized circles.

In addition to its control system, the lead aircraft is outfitted with a known pattern of high-intensity LED lights. The trailing aircraft captures images of these LEDs with a camera and uses the Orthogonal Iteration computer vision algorithm to determine the relative position and orientation of the trailing aircraft with respect to the lead aircraft at 25Hz. The entire process is carried-out in real-time with both vehicles flying autonomously. We note that the camera based system is used for localization, but not yet for closed-loop control.

Although, an absolute quantification of the error for the in-air localization system is difficult as we do not have ground truth positioning data during flight testing, our simulation results analysis and indoor measurements suggest that we can achieve localization accuracy on the order of 10 cm (5% wingspan) when the UAVs are separated by a distance of about 10 meters (5 spans).

I. Introduction

A. Motivation

Recently, there has been increased interest in UAV formation flight for a number of reasons. First, formation flight offers the possibility for significant energy savings due to reductions in induced drag [1]. Such energy savings have been demonstrated for human piloted vehicles [2,3], but due to the high pilot workload required to maintain the necessary spacing tolerances the practice is unlikely to be common until there are robust autonomous formation control systems. Second, autonomous UAV air-to-air refueling, a maneuver that also requires close formation flight, may greatly extend range and/or endurance without human intervention. Such systems could also improve the safely of air-to-air refueling for manned aircraft in low visibility conditions.

Both energy saving formation flight and autonomous air-to-air refueling have a similar and highly critical reliance on accurate in-air localization between aircraft. For instance, to achieve consistent energy savings in formation flight the relative position of the two aircraft must be controlled to within about 5% of the wingspan both horizontally and vertically [4]. For UAVs of relatively small scale, such accuracy is well beyond the capabilities of single-receiver GPS and inertial systems. Thus, additional sensors are needed to accurately localize the multiple UAVs in relation to each other.

In this paper, we present and demonstrate a camera-based tracking system that achieves accurate in-air localization. While a number of localization methods are possible (most notably a between-vehicle differential GPS (DGPS) system, as demonstrated for human piloted aircraft in [2]), we chose to work with a camera-based system for a number of reasons. First, the ubiquity of cheap, low weight and high quality camera systems means that such systems have ideal characteristics for UAVs. In contrast, to the best of our knowledge there are no off-the-shelf DGPS solutions that allow both receivers to be moving over the ground, and the systems that do exist are typically very costly and much heavier than the comparable camera-based setup. Furthermore, DGPS requires an active communication channel between the two aircraft, whereas camera-based localization can be achieved via processing only on the trailing aircraft.

*Aeronautics and Astronautics, Stanford University. AIAA Student Member.
†Computer Science and Artificial Intelligence Laboartory, Massachusetts Institute of Technology.
‡Computer Science, Stanford University.
§Aeronautics and Astronautics, Stanford University. AIAA Student Member.
¶Computer Science, Stanford University

B. Existing Work

There has been some prior research on camera based localization systems for aircraft, especially with applications to aerial refueling[5–7]. However, to the best of our knowledge these studies focus on algorithm development and simulations. To date there have been no published demonstrations of full 6 degree-of-freedom (DOF) in-air localization using such techniques. In this paper we demonstrate such a system applied to two small UAV platforms flying in formation.

In our experience, even if noise is included in simulations, algorithms tend to over-perform in virtual environments. Challenges posed by un-modeled dynamics, external disturbances, and unexpected sensor data are hard to fully capture without realistic experimentation. The main contribution of this paper is in demonstrating the feasibility of the concept of vision based in-air localization and providing realistic data sets that can help other researchers to test their algorithms; therefore laying the foundation for future work in active UAV formation flight control using such approaches. Although we still ultimately obtain quantitative estimates of the localization errors via simulation, we emphasize that the final system is demonstrated on the real aircraft while in flight.

II. Experimental Setup

A. Description

Our experimental setup consists of two remote controlled trainer aircraft. Each aircraft weighs about 4 kg, has a span of 1.8 m, a wing area of 0.6 m^2 and cruises at an airspeed between 11 m/s and 15 m/s. They each carry a 4 cell 3900 mAh LiPo battery which powers a 400 W brushless motor (about 200 W in cruise) which gives us flight times between 10 and 20 minutes. The aircraft were converted to autonomous UAVs by adding a suit of sensors including GPS, an inertial measurement unit (IMU) and an airspeed sensor. Figure 1 shows the aircraft with two team members illustrating the scale.

We used the open-source *paparazzi* autopilot in order to reduce development time — by relying on an existing architecture, it is possible to take advantage of a development tool chain, ground station (Fig. 12), communication protocol, community support etc. However, we have also substantially modified some parts of the autopilot in order to use our own simulation tools and control algorithms. We also developed our own ground station to accommodate the camera data (Fig. 13). Moreover, the paparazzi autopilot traditionally runs on a micro-controller, but due to the high computation power requirements of real-time image processing, we migrated the autopilot to an x86 computer running Linux. We were initially concerned with the control software no longer running in real-time, especially that many people in the UAV community favor real-time operating systems or micro-controllers. However, we did not see any performance degradations from the migration.

The vehicles have three different RF links to the ground in two frequency spectra. A 2.4 GHz Wi-Fi link is used for loading software which is very useful for development. However, it has a fairly limited range, therefore 900 Mhz Digi xTend radios are used as the primary communication link for both telemetry and uplink of commands. Finally, a 2.4Ghz RC transmitter/receiver pair are used as a safety-link to always allow a trained pilot to take control as well as perform complicated maneuvers (such as take-off and landing).

The lead aircraft (named **Batman**) also carries 5 high-intensity LEDs placed at known locations relative to its center of gravity (CG), while the chase aircraft (named **Joker**) carries a camera with a known orientation and location relative to its own CG. Apart from the LEDs and cameras, the two aircraft are quasi-identical, which makes development easier since the same components and codebase can be used on both vehicles.

B. Hardware Architecture

We initially planned to use the Paparazzi Autopilot System[8] as our autopilot unit to realize autonomous flight. Paparazzi is a free and open-source hardware and software project designed at ENAC University (France) for unmanned aerial vehicle development. The hardware of the Paparazzi autopilot system consists of the Tiny v2 control board, a GPS receiver, and two IR sensor boards for attitude measurement. The control board interfaces with the GPS receiver and the radio modem via UART ports, while the IR sensors are connected to the ADC channels. Apart from controlling the servos and motor speed controller directly by outputting its own pulse-width modulation (PWM) signals in autonomous mode, *Paparazzi-Tiny* is also able to receive the pulse-position modulation (PPM) frame from the FM receiver, and decode it into separate (PWM) signals in manual mode, so that a human pilot can takeover and control the airplane via the R/C transmitter during takeoff, landing and emergency situations. However, we identified some issues with the original system:

1. *Paparazzi-Tiny*'s Micro-controller (LPC2148FBD64) has maximum speed of 60MHz, which may not be fast enough to run our desired vision algorithms.

2. *Paparazzi-Tiny* has a built-in multiplexer mechanism to select between the autopilot servo commands and the

Figure 1. Batman and Joker, two RC aircraft converted to UAVs

manual override commands. If the main processor on the board malfunctions, the pilot may lose the ability to take control of the aircraft.

3. The IR sensors have slow reaction time (4Hz). They are also susceptible to humidity and perform poorly in cloudy weather.

For these reasons, we decide to augment the original Paparazzi system with a more powerful x86 computer running Linux (fit-PC2). The *Paparazzi-Tiny* board now only serves as an ADC converter to obtain measurements on airspeed and the propulsion battery current and voltage. It could be replaced by any other I/O board with analog inputs. Apart from providing much higher computation power, the fit-PC2 computer has a number of USB ports via which we can interface with additional sensors and devices. For the manual override system, a Pololu multiplexer selects between the autopilot servo commands and the manual RC commands. Unlike the original system, the multiplexer is powered by the RC receiver directly and is always able to switch into manual override mode even if other parts of the autopilot fail. This solves the safety problem identified with the original system. Additionally, we used a Microstrain 3DM-GX2 inertia measurement unit (IMU) in place of the IR sensors to obtain higher rate measurements for attitude estimation. This makes the attitude measurements no longer dependent on local terrain or weather conditions that affect IR attitude sensors. Figure 2 illustrates the hardware architecture for the autopilot used on both UAVs. The entire system weights about 400 grams, which is heavy for small aircraft, but is within the payload capacity of our vehicles.

III. Localization

One of the primary technical challenges encountered in autonomous formation flight is the task of accurately positioning the chase airplane relative to the lead airplane. As previously mentioned, in order to achieve substantial energy savings from formation flight, the aircraft positioning must be maintained within about 5% of the wingspan both vertically and horizontally. For the small UAVs we consider, this translates to relative positioning accuracy requirements on the order of centimeters. This is well below the precision of code phase positioning used by commercial GPS units. Even for experiments with much larger F-18s, NASA researchers had to developed a custom differential GPS (DPGS) and carrier-phase to obtained precise positioning between the two aircraft. And while DGPS systems do indeed offer a potential method for accurate localization, the cost, weight, and bandwidth issues of such systems make them less well suited to UAV applications. Indeed, prior to the vision approach we describe here, our group worked extensively with a DGPS system, but weight and data bandwidth made it unsuitable to our application; furthermore, all off-the-shelf DGPS solutions that we are aware of require one of the two receivers to be fixed on the ground, severely limiting the range and applicability of such hardware. It is also worth worth noting that while possible to achieve centimeter accuracy with indoor motion capture systems, the usable area they offer is rarely suitable for fixed wing aircraft (which is why the majority of indoor UAV research uses helicopters or quad-rotors).

A. Camera Algorithm

A viable alternative to GPS is using vision based systems for localization. This approach has been proposed by[6,7] and some work has been carried out in simulation, but here we describe the theory behind the system and our algorithm as implemented in our UAV system.

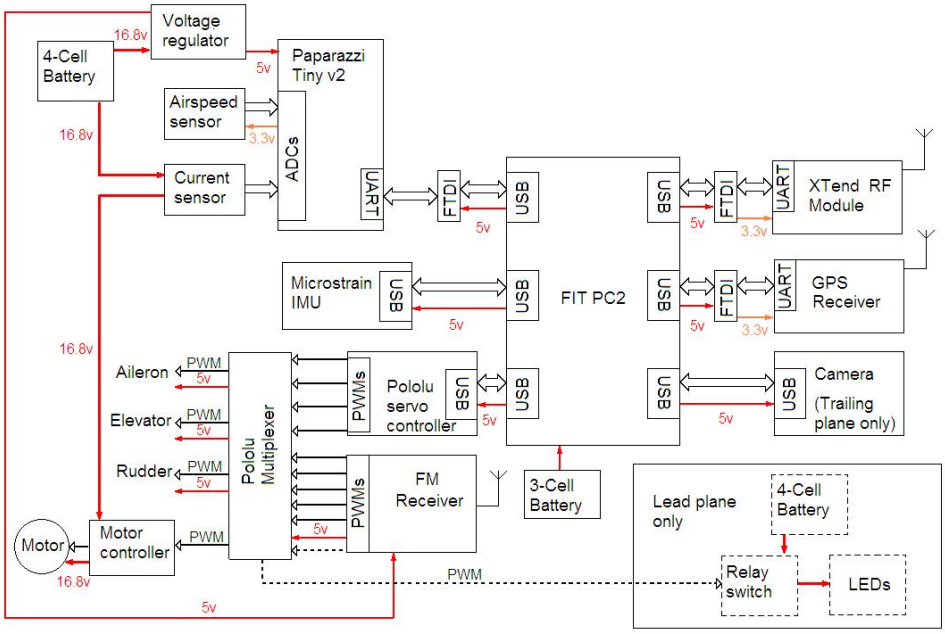

Figure 2. Autopilot hardware architecture used on the UAVs.

The objective of our camera-based localization algorithm is to determine the full 6-DOF transformation between the lead aircraft and the chase aircraft. Formally, we denote this as finding the 4x4 homogeneous transformation ${}^2_1T$ (using the convention that airplane 1 is the leader and airplane 2 is the chase airplane); this matrix has the form

$$ {}^2_1T \equiv \begin{bmatrix} {}^2_1R & {}^2_1p \\ 0 & 1 \end{bmatrix} \tag{1} $$

where ${}^2_1R \in \mathbb{R}^{3\times 3}$ is an orthogonal matrix denoting the rotation of airplane 1 relative to airplane 2 and similarly ${}^2_1p \in \mathbb{R}^3$ is a vector denoting the position of airplane 1 relative to airplane 2. In practice, we are unable to directly measure this transformation, as the algorithm below will provide the position and orientation of the lead airplane relative to the *camera*, i.e. ${}^C_1T$; however, presuming that we know the pose of how the camera is affixed to the chase airplane, ${}^2_CT$ (which we can easily measure directly, or determined via a calibration procedure), then it is trivial to convert between the two, since

$$ {}^2_1T = {}^2_CT\,{}^C_1T \tag{2} $$

or more explicitly,

$$ {}^2_1R = {}^2_CR\,{}^C_1R, \quad {}^2_1p = {}^2_CR\,{}^C_1p + {}^2_Cp. \tag{3} $$

As mentioned briefly above, in order to compute the pose of airplane 1 relative to the camera, we affix a number of LEDs on the lead airplane, in known positions (in order to guarantee uniqueness of the pose, four points are required, though we use 5 to allow for redundancy). Intuitively then, given a camera image of these points, we can find the pose of the lead airplane that minimizes the error between the ideal projected points and the observed image. However, the most straightforward procedures for this problem (such as direct optimization of the projection error), tend to be sensitive to local optima, and there can been much work on more efficient algorithms for this problem in the computer vision community (here this task is known as the perspective n point, or PnP problem). In particular, we employ a method known as the Orthogonal Iteration (OI) algorithm[9], which has been shown to be highly efficient for this task. Although we refer to this paper for a more complete description of the algorithm, briefly, the procedure is as described below.

We suppose that we are given a set of n points, the ith such point denoted $q_i \in \mathbb{R}^3$, fixed relative to the origin of the object (corresponding in our case to the fixed locations of the LEDs relative to the center of the lead airplane). Assume that the homogeneous image coordinate of the ith point is observed to be $z_i \in \mathbb{R}^3 = (u_i, v_i, 1)$, i.e.,

$$ u_i = \frac{x_i - c_x}{f_x}, \quad , v_i = \frac{y_i - c_y}{f_y} \tag{4} $$

where x_i and y_i are the actual points in the camera image, c_x and c_y are the center points of the camera, and f_x and f_y are the respective focal lengths; together, c_x, c_y, f_x, and f_y are known as the *intrinsic parameters* of the camera, and numerous software packages exist to compute these parameters for a given camera (for this work, we use the Caltech

Camera Calibration Toolkit [10]). Given these image points, the OI algorithm seeks to minimize a projected object space collinearity error, given by

$$E(R,p) = \sum_{i=1}^{n} \|(I - Z_i)(Rq_i + p)\|^2, \text{ where } Z_i \equiv \frac{z_i z_i^T}{\|z_i\|^2}. \tag{5}$$

where R and p denote the transformation from the camera to the object (i.e., in our case, they correspond to ${}^c_1R$ and ${}^c_1p$). Although this error cannot be minimized analytically with respect to both R and p, it can be solved analytically for either variable alone, using an SVD or least squares respectively. Thus, to minimize this error, the algorithm iteratively computes

$$\begin{aligned} p &\leftarrow \frac{1}{n}\left(I - \frac{1}{n}\sum_i Z_i\right)^{-1} \sum_i (Z_i - I)Rq_i \\ R &\leftarrow UV^T, \text{ where } M = U\Sigma V^T \text{ is the SVD of } M = \sum_{i=1}^{M}(q_i' - \bar{q}')(q_i - \bar{q}). \end{aligned} \tag{6}$$

where $q_i' \equiv Z_i(Rq_i + p)$, $\bar{q} = \frac{1}{n}\sum_{i=1}^n q_i$, and $\bar{q}' = \frac{1}{n}\sum_{i=1}^n q_i'$. We refer to the paper cited above for a derivation of this algorithm as well as a proof of it's convergence, but this algorithm in particular has been found to be one of the more accurate and efficient methods for the PnP problem.

Finally, we note that a number of other issues are involved with converting, in real-time, from actual camera images to an input format suitable for this algorithm. In our final implementation we use a heavy bandpass lens filter to remove all colors other than those of the LED (see Fig. 11 below), use an optimized version of the libjpeg library to decode images in real-time [11], then look for areas in the image exceeding a certain relative exposure threshold (the best of up to seven of these are are taken to be candidate locations for the LEDs). However, since the OI algorithm requires not only the image points but also the correspondences (i.e., we must specify *which* of the LEDs in the lead airplane each image point corresponds to), we actually run the OI algorithm for all possible permutations of the candidate image points that satisfy certain geometric feasibility constraints (i.e., the lead place cannot be flying upside down, etc). Despite the complexity of these computations, because of the optimized image processing and efficiency of the OI algorithm, we can compute these estimates in real-time (up to the 25fps video rate of the camera), on the 1.6 GHz onboard processor.

B. Fusing camera and other sensor measurements

In order to properly fuse the camera localization information with the other sensing in each airplane, we employ an Extended Kalman Filter. The state of each airplane is represented via 11 state variables: the standard 12 degree of freedom model, with 3-D position (x, y, z), orientation (ϕ, θ, ψ) velocity (in the wind frame) (u, v, w) and angular velocity (p, q, r), but where we assume that v is zero (i.e., we assume zero side-slip angle in the wind frame, since we have no reliable way of measuring this quantity). The EKF employed to track each airplane individually is fairly standard, so we describe it only briefly: we treat the filtered IMU orientation and angular velocity estimates (which themselves combine an accelerometer, magnetometer, and gyroscope) as direct measurements of the airplane orientation and angular velocity with small noise. One minor issue that arises with this method is that the internal filter of the IMU cannot know the true accelerations caused by flying in a circle, and thus the filtered orientation will drift slowly over time; however, for the circles and time scales that were involved in our experiments this was a very minor affect. We use the airspeed measurement as a noisy measurement of u and the GPS to obtain a noisy measurement of w. Finally, comparing the GPS velocity and GPS heading to the airspeed and magnetometer heading allows us to obtain a noisy estimate of the wind; in practice, we estimate the wind using two additional variables in the Kalman filter (with very low variation over time), and treat GPS groundspeed and heading as measurements of the true 2D velocity in the wind frame plus the wind velocity. In this manner, we are able to estimate the full 11-D state of the airplane, plus the wind velocity itself.

Another nice feature of the EKF framework is that it allows us to directly incorporate measurements from the camera system. In particular, although the filter tracks the *absolute* state of each airplane, we can also easily incorporate *relative* measurements between each airplane by running a single filter that contains the state estimates of *both* airplanes. As long as as no camera measurements are introduced, this dual filter operates the same as two independent filters of each airplane's state. However, we can now incorporate measurements of the relative pose between the airplanes.

First recall the standard EKF measurement update. Let $x \in \mathbb{R}^n$ denote the true (unknown) state of the system, $\hat{x} \in \mathbb{R}^n$ the estimated state of the filter, and $\Sigma \in \mathbb{R}^{n\times n}$ its covariance. Then given some measurement $y \in \mathbb{R}^m = h(x) + r$, where we assume that $r \sim \mathcal{N}(0, R)$, the EKF updates the state estimate according to

$$\begin{aligned} K &\leftarrow \Sigma C^T(C\Sigma C^T + R)^{-1} \\ \hat{x} &\leftarrow \hat{x} + K(y - h(\hat{x})) \\ \Sigma &\leftarrow (I - KC)\Sigma \end{aligned} \tag{7}$$

where $C \in \mathbb{R}^{m \times n}$ is the Jacobian of he measurement function evaluated at the current filter estimate

$$C = \frac{\partial h(\hat{x})}{\partial \hat{x}}. \tag{8}$$

Now given a filter where $\hat{x}$ contains state estimates for both airplanes (abusing notation slightly, we will refer to the elements of this matrix as $(\hat{x}_1, \hat{y}_1, \hat{z}_1, \hat{\phi}_1, \hat{\theta}_1, \ldots, \hat{x}_2, \hat{y}_2, \hat{z}_2, \hat{\phi}_2, \ldots)$, since it should be clear from context whether we mean x to refer to the complete state versus just the single position component of the state), it should be clear that we can treat ${}^2_1R$ and ${}^2_1p$ (the measurement returned by the camera system), as functions of the state. In particular, letting ${}^2_1\hat{R}$ and ${}^2_1\hat{p}$ denote the expected relative pose of the two airplanes given the Kalman Filter estimate, we have that

$$\begin{aligned} {}^2_1\hat{R} &= R(\hat{\phi}_2, \hat{\theta}_2, \hat{\psi}_2)^T R(\hat{\phi}_1, \hat{\theta}_1, \hat{\psi}_1) \\ {}^2_1\hat{p} &= R(\hat{\phi}_2, \hat{\theta}_2, \hat{\psi}_2)^T \begin{bmatrix} \hat{x}_1 - \hat{x}_2 \\ \hat{y}_1 - \hat{y}_2 \\ \hat{z}_1 - \hat{z}_2 \end{bmatrix} \end{aligned} \tag{9}$$

where $R(\phi, \theta, \psi)$ denote the rotation matrix formed from Euler angles ϕ, θ, ψ. Thus we can linearize this measurement in the standard form, and directly incorporate the output of the OI algorithm in the EKF. One item to note is that this update technically assumes that each element of the ${}^2_1R$ and ${}^2_1p$ matrices returned by the OI algorithm are corrupted with Gaussian noise, which is a poor assumption since ${}^2_1R$ must span the lower-dimensional manifold of rotation matrices; nonetheless, in practice this has a very small effect on the performance of the filter, since we assume camera measurements to have very low noise relative to GPS measurements. Alternatively one could also directly express the measurements in terms of an Euler angle different between the two system, thus eliminating this potential source of error; in practice, this alternative approach makes virtually no difference, so we present the conceptually simpler approach above.

An alternative approach to what we describe here is to directly incorporate the observed camera images into the EKF, and include the projective geometry of the camera in the measurement update itself, and indeed we did also experiment with this method in preliminary tests. However, while this is certainly a viable approach, this is roughly equivalent to direct non-linear optimization of the projection error for the PnP pose estimation problem, a technique that, as mentioned above, is typically less robust than the more recent computer vision algorithms. Instead, the method we propose allows us to use the full power of recent computer vision algorithms for directly estimating the pose, and allows us to separate out elements such as determining the point correspondences from the EKF itself.

IV. Controls

A. Strategy

Ideally, we would like the lead aircraft to fly in a straight line while holding altitude and airspeed. However this is impractical due to space restrictions at the flying field. Instead, we opted to fly a circle pattern with as large of a radius as possible. This leads to a relatively small steady-state bank angle which approximates level-flight. The chase aircraft flies a similar pattern, but it adjusts its speed, altitude and position relative to the circle based on its best estimate of the location of the lead airplane. The reason we do this is that we expect in the steady-state and noise free condition the chase airplane to converge to the same circle as the lead airplane, therefore we use it as a feed-forward for the chase aircraft controller. Figure 3 illustrates the high-level control strategy, which is effectively an outer loop that produces target altitudes, airspeeds, heading and bank angles. Through successive loop closure, an inner loop controller then commands the control surfaces and power system to achieve these targets.

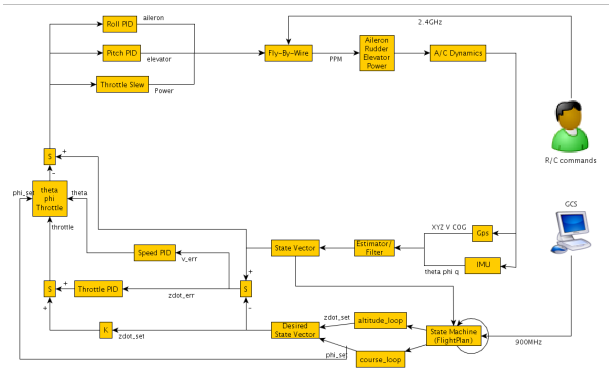

Figure 3. Block diagram of the UAV control architecture.

B. Model-based LQG control

The inner-loops of the controller use an LQG controller; a combination of an LQR controller and EKF estimator. The controller assumes a linear dynamics model for the aircraft. This model is constructed in two steps. First, we estimated the relevant parameters for each airplane by measuring the dimensions, weight, and moments of inertia and by generating a vortex lattice model of the vehicle using AVL to estimate stability derivatives[12] (Fig. 4 shows the airplane modeled in AVL). In parallel to the control development, we integrated JSBSim, a non-linear 6DOF dynamic simulator aimed for aerodynamic simulations, with our flight software. This allowed us to test our control algorithms in both Software-in-the-loop (SITL) as well as hardware-in-the-loop (HITL) simulation environments using the exact same code that the flight computer runs.

The second step in constructing the model uses machine learning to improve the parameter estimates through system identification. On some flights the human RC pilot would manually perform control sweeps and excite the natural modes of the aircraft. The estimated parameters were then varied by a non-linear optimizer to minimize the discrepancy between the predicted and observed trajectories for the same set of pilot control inputs. With the system identification done, synthesizing a controller was straight forward with the help of LQR. That being said, a fair amount of fine tuning of the relative weights had to be done in order to obtain satisfactory responses.

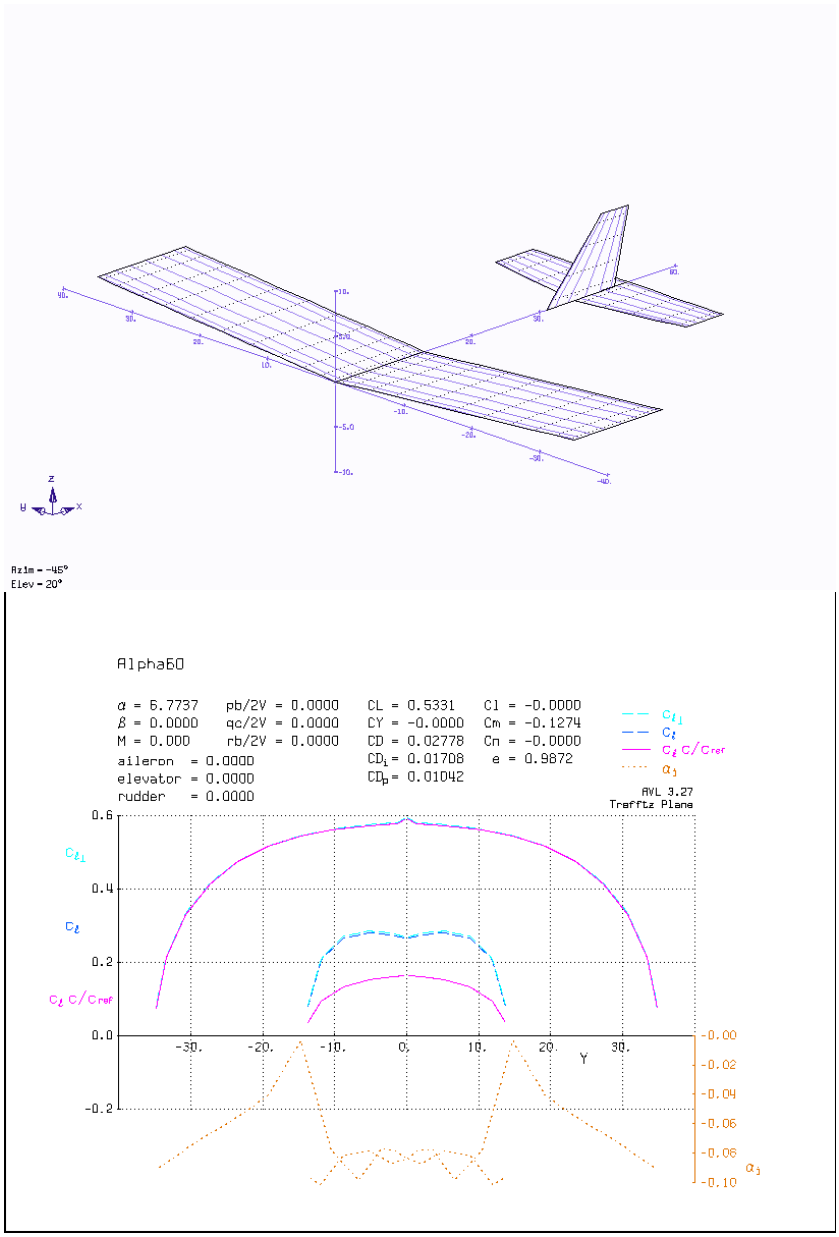

Figure 4. AVL model and cruise analysis of the aircraft.

C. Circle Tracking

While it is relatively straight-forward to fly around in circles, flying a precise circle with a fixed wing airplane of this size turned out to be a non-trivial task, especially in the presence of winds. We tried different algorithms and eventually used an LQR based controller for trajectory tracking. However, in this section we describe another controller which was extensively used initially. This approach is an adaptation of the steering control of the Stanley car in the DARPA Grand Challenge [13] applied to a fixed-wing aircraft.

The main idea behind the controller is to command a bank angle ϕ which the inner-loop controllers described in the previous section can track. By relying on the inner loop controller, we reduce the relative order of the system being controlled. Moreover, this allows us to easily set constraints on commanded bank angles which is difficult to do if commands are directly issued to the control surfaces (ailerons in this case). For brevity, we will assume that we are trying to fly a clockwise circle, although the direction of the controller can be easily adjusted by assigning a sign to the radius and using that as a multiplier for the error terms.

Figure 5 illustrates the problem at hand. We would like the aircraft to be located at a distance R from the center of predefined circle, while flying tangentially to said circle. In other terms, we would like the error e(t) to be 0 and the course ψ to be $\pi - \eta$ (the π is required here since by convention aircraft course uses North as a datum). Given $e(t)$, $\dot{e}(t)$ and $\int e(t)dt$ a simple PID controller can be formulated.

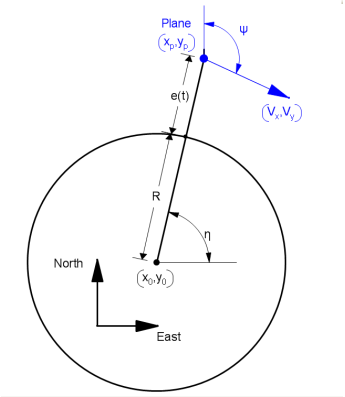

Figure 5. Circle tracking geometry

Assuming that the airplane's coordinates (x_p, y_p) is given in the North-East-Down relative to the circle's origin (x_o, y_o), η and e are readily available from simple geometry:

$$\eta = atan2(y_p, x_p), e = \sqrt{{x_p}^2 + {y_p}^2} - R \tag{10}$$

It is possible to compute the error derivative $\dot{e}$ numerically; however, since we have information about the velocity (V_x, V_y), it is better to compute analytically by simply differentiating the expression for e which yields:

$$\dot{e} = \frac{x_p\, \dot{x_p} + y_p\, \dot{y_p}}{\sqrt{{x_p}^2 + {y_p}^2}} = \frac{x_p\, V_x + y_p\, V_y}{e + R} \tag{11}$$

This can be further simplified by realizing that the numerator of the expression above is a dot product. Using some geometry we can show that:

$$\dot{e} = \frac{\vec{r_p} \cdot \vec{V_p}}{e + R} = \frac{(e + R)V_p cos(\psi + \eta - \pi/2)}{e + R} = V_p\, cos(\psi + \eta - \pi/2) \tag{12}$$

It is worth noting that in the steady-state when $e = 0$ and $\psi = \pi - \eta$ we also have $\dot{e} = 0$. Putting all of this together, we propose the following controller which apart from the integral term, uses data readily available from the estimator:

$$\psi_c = \pi - \eta + k_p\, atan(e) + k_d \dot{e} + k_I \int e\, dt \tag{13}$$

However, this controller commands a heading and our inner-loop controllers are wrapped around bank angle ϕ. This means that an additional loop from course to bank angle is required. Another PID controller was used:

$$\phi_c = \phi_{ss} + k_p\, e_\psi + k_d\, \dot{\psi} + k_I \int e_\psi\, dt \tag{14}$$

where $e_\psi = (\psi_c - \psi(t))$ is the error in course [‖] and $\phi_{ss} = atan(\frac{V}{gR})$ is the steady-state bank angle used as feed-forward.

[‖] Note that we need to pay attention to the wrapping of the course angle at 2π, this is easily handled by letting $e_\psi = atan2(sin(e_\psi), cos(e_\psi))$

V. Results

A. Simulation Results

1. *Circle Tracker Simulation*

In order to verify that the controller proposed for tracking a circle works, a simple simulation in Matlab was put in place. We only simulate the outer-loop controller (i.e. from position to heading) and assume that the inner-loop controller is a perfect coordinated-turn controller. In other words, the commanded bank angle ϕ is achieved immediately and the turn rate is given by $\dot{\psi} = \frac{g\ sin(\phi)}{V}$.

Figure 6 shows the controller operating with no wind in the simulation and a target radius R=100 m, with different initial positions and velocities. Both initial positions inside and outside of the circle were simulated (with radii R/2 and $1.5 \times R$ respectively). The positions are spaced by 60^o around the circle, but in all cases the initial course is towards the North. This allows us to see how the controller behaves for various combinations of initial positions and velocities.

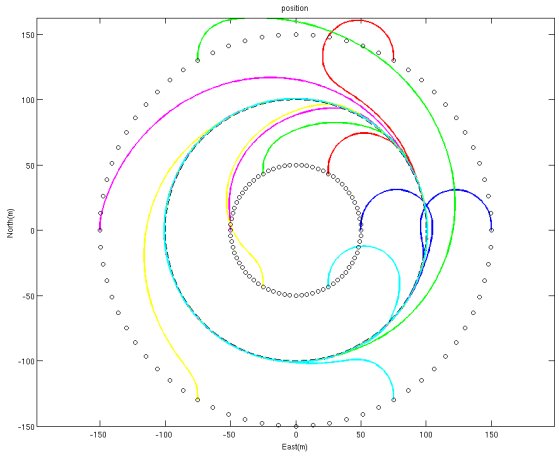

Figure 6. Tracking a circle with R=100m from various initial positions. Initial course is to the North and there is no wind.

Figure 7 shows the details of the simulation of the controller for one initial condition. As can be seen in the commanded bank angle, a maximum of $\pm 40^o$ was imposed.

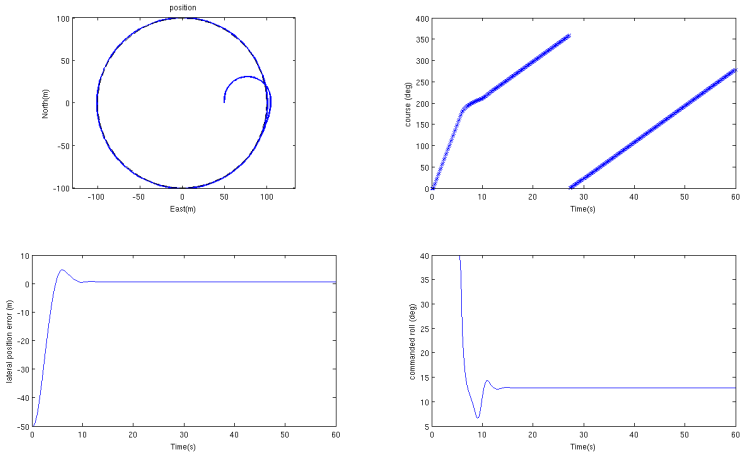

Figure 7. Trajectory, heading, position error and bank angle input for tracking a circle with R=100m. Initial course is to the North with no wind.

We also simulated our controllers with some wind in the JSBSim non-linear simulator, and even adapted them to use wind information to improve the circle tracking. Estimating wind was relatively straightforward given airspeed and GPS measurements, especially that we were constantly flying in circles.

In addition to the real-world experiments we will describe below, we have conducted extensive evaluations of the localization system in simulation. In our simulator we collected data from two airplanes flying in formation, with the chase airplane keeping a distance of two to 15 meters behind the lead airplane. Using the true positions between the two airplanes we simulated LED locations in a camera image, added Gaussian noise to these locations, and then ran the above localization algorithm to estimate the pose between the two airplanes.

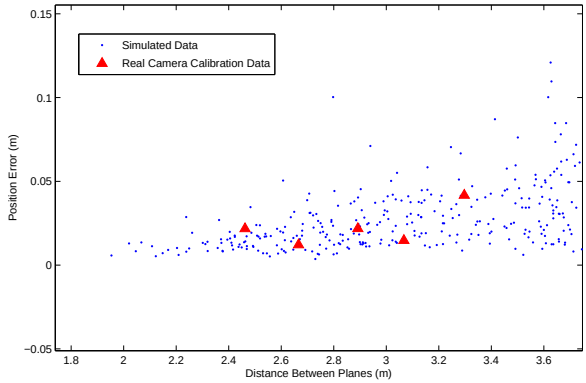

Figure 8. Simulated and real camera pose estimation errors of the lead airplane at different distances from the trailing airplane.

The covariance of the noise we added to the LED locations was estimated to match the real system as closely as possible. Although, as mentioned above, it is difficult to obtain precise accuracies of the camera localization in flight or at long distances (since we have little information about the ground truth positioning of the airplanes in such cases), we tested its accuracy at close range using ground truth data from an indoor Phase-space motion capture system. The errors in these measurements correspond to an error of approximately 3 pixels in the camera display. The blue dots in Figure 8 show the pose estimation error in simulation, using a standard deviation of 3 pixels in the LED location. Similarly, the red triangles illustrate the pose reconstruction error for the real system for five known configurations in the motion capture system. We estimated the standard deviation of three pixels by fitting the resulting errors from simulation to the collected data.

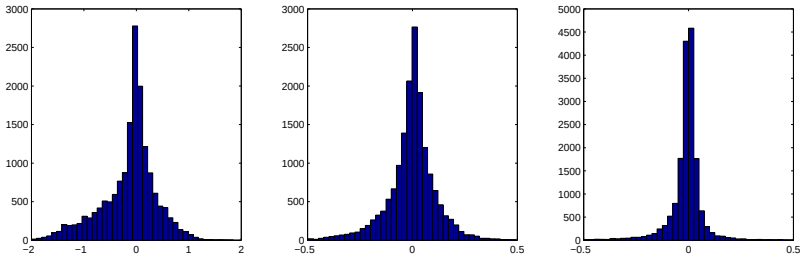

Figure 9. Histograms for simulated pose reconstruction error (in meters) in the longitudinal (left), lateral (center), and vertical (right) directions.

After estimating the noise in this manner we examined the pose reconstruction error for much longer distances in the simulation alone. While this will naturally be only an approximation to the true camera's error, for example due to the fact that LED location errors in the real system will not likely be Gaussian distributed, it still provides a rough estimate of the errors we would expect. Figure 9 shows histograms of the pose estimation errors for 16000 simulated data points, with a distance of two to 15 meters between the two aircraft. As expected, the error in the longitudinal direction is the largest; here, small errors in dot locations, when the lead airplane is far away, can lead to relatively large estimation errors. Fortunately, however, positioning of the tail aircraft in the lateral and vertical directions is significantly more important for formation flight, and in these directions the errors are much lower: after applying the Kalman filer, we obtain a 11.3cm RMSE for the lateral error and 7.9cm RMSE for the vertical error. For example, these are of the order necessary to see significant fuel savings in formation flight for vehicles of this size. This gives an initial indication that such camera localization is a promising approach to in-air localization for formation flight.

B. Experimental Results

So far we have been able to successfully and repeatedly carry autonomous flights with both airplanes. Using GPS only, we were able to get close enough to have Batman's LEDs in the field of vision of Joker's camera. The challenge has been to get the airplanes to fly repeatable circles despite winds and noisy measurements from the sensors. Moreover, we have only been able to hold altitude to within 3 m, which is not accurate enough for the close-range formation task described above. We believe this is caused by the fact that we are relying on GPS alone to estimate altitude and climb rates. Indeed, GPS altitude accuracy is not very reliable, and it is no surprise that we cannot hold altitude more accurately.

Regardless, we have been able to realize flights in which the lead airplane remained in the field of vision for over 5 minutes, and were able to collect extensive in-flight data with the camera. Applying the algorithms above, we were able to localize the lead for this 5 minute time frame for using the in-flight camera data, with results shown in Figure 10, 11, and 13. However, we note that thus far we have not used the localization estimates from the vision system to control the trailing aircraft, since the complete estimator which fuses these estimates with the GPS data has not been completely vetted.

Figure 10. View from the camera on the trailing aircraft. The red dots on the lead aircraft are high power LEDs used for relative positioning.

Figure 11. View from the camera with red filter (helps with detecting the LEDs)

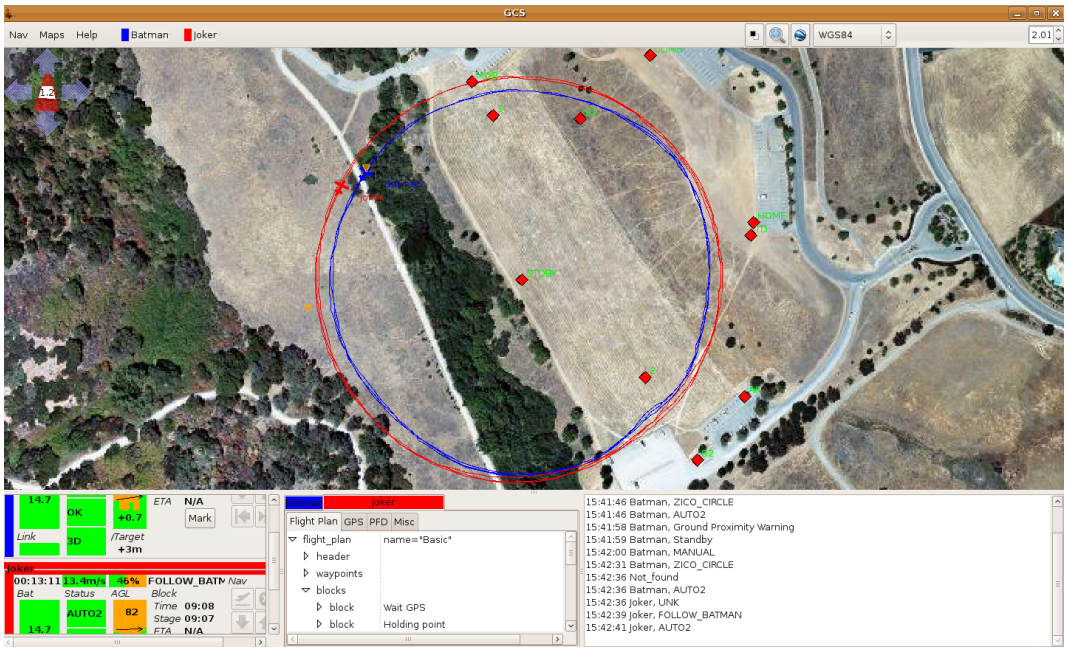

Figure 12. Screenshot of the paparazzi GCS with the groundtracks of the two vehicles

Figure 13. Screenshot of the custom GUI with camera view in the bottom right, filtered image in the bottom left and position estimates in the top (camera view only available in flight log-replay mode)

VI. Conclusion

We have taken two RC airplanes and made them capable of flying autonomously. We have collected data which we used for system identification in order to extract a linear model and synthesize an LQR controller. Using an LQG based control approach, we were able to control the speed and altitude of the airplanes in order to be able to position them relatively close to each other using GPS data only.

Real-time localization using the vision data was shown to work both in the lab and in flight. However, due to the absence of a ground truth measurement system, we are only able to make claims of position accuracy from simulation results based on noise estimated from collected data. Our conclusion is that for longitudinal separations between 2m and 15m, it is possible to achieve in-air localization using a camera with errors in the vertical and lateral direction on the order of 10cm RMSE. This gives at least an initial indication that such camera localization is a promising approach to in-air localization for formation flight.

A. Future Work

The system we present here is admittedly experimental, and there are many additions that could increase the robustness of the overall system. For instance, we could include a tighter coupling of attitude and position estimation. Indeed, we relied on the attitude estimation of the IMU which was not properly fused with position and velocity information from GPS and other sensors. The same thing can be said about altitude estimation, where barometric pressure could be used to reduce noise from gps measurements. In the future, the proposed Extended Kalman Filter could be augmented to include the raw IMU sensors, or a system with full Attitude Heading Reference System (AHRS) could be used instead. Additionally, all of our controllers were designed by linearizing the system around a single trim point. It would be beneficial in the future to implement gain scheduling based on dynamic pressure at different velocities and air densities.

B. Acknowledgments

We thank our pilot Garett, whose skills have avoided some imminent crashes in the early stages of the flight-testing. We also thank Pieter Abbeel and Tim Hunter for helpful work and discussions on early portions of this project.

References

[1]King, R. and Gopalarathnam, A., "Ideal Aerodynamics of Ground Effect and Formation Flight," *Journal of Aircraft*, Vol. 42, No. 5, 2005, pp. 1188.

[2]Ronald, J. R. et al., "Flight Test Techniques Used to Evaluate Performance Benefits During Formation Flight," Nasa/tp-2002-210730, NASA Dryden Flight Research Center, Edwards, CA, 2002.

[3]Hummel, D., "The use of aircraft wakes to achieve power reductions in formation flight," *Proceedings of the AGARD Conference*, AGARD, Dresden, Germany, 1996, pp. 1–13.

[4]Ning, A., Tristan, F., and Kroo, I., "Aerodynamic Performance of Extended Formation Flight," *Proceedings of 48th AIAA Aerospace Sciences Meeting*, AIAA, 2010.

[5]Giampero, C. et al., "Machine Vision/GPS Integration Using EKF for the UAV Aerial Refueling Problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 38, IEEE, 2008, pp. 791–801.

[6]Giampero, C. et al., "Simulation Environment for Machine Vision Based Aerial Refueling for UAVs," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 45, IEEE, 2009, pp. 138–151.

[7]Pollini, L., Innocenti, M., and Mati, R., "Vision Algorithms for Formation Flight and Aerial Refueling with Optimal Marker Labeling," *Proceedings of AIAA Modeling and Simulation Technologies Conference and Exhibit*, AIAA, 2005.

[8]"Paparazzi: The Free Autopilot," http://paparazzi.enac.fr/.

[9]Lu, C.-P., Hager, G. D., and Mjolsnes, E., "Fast and globally convergent pose estimation from video images," *IEEE Transations on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 6, 2000, pp. 610–622.

[10]Bouguet, J.-Y., "Camera Calibration Toolbox for Matlab," http://www.vision.caltech.edu/bouguetj/calib_doc/.

[11]Project, T. V., "libjpeg-turbo," http://libjpeg-turbo.virtualgl.org/.

[12]Drela, M., "Extended Vortex-Lattice Model," http://web.mit.edu/drela/Public/web/avl/.

[13]Thrun, S. et al., "Stanley: The Robot that Won the DARPA Grand Challenge," *Journal of Field Robotics*, Vol. 23, No. 9, 2006, pp. 684–685.