2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

Autonomous Vision-based Landing and Terrain Mapping Using an MPC-controlled Unmanned Rotorcraft

Todd Templeton, David Hyunchul Shim, Christopher Geyer, and S. Shankar Sastry*

Abstract

In this paper, we present a vision-based terrain mapping and analysis system, and a model predictive control (MPC)-based flight control system, for autonomous landing of a helicopter-based unmanned aerial vehicle (UAV) in unknown terrain. The vision system is centered around Geyer et al.'s Recursive Multi-Frame Planar Parallax algorithm [1], which accurately estimates 3D structure using georeferenced images from a single camera, as well as a modular and efficient mapping and terrain analysis module. The vision system determines the best trajectory to cover large areas of terrain or to perform closer inspection of potential landing sites, and the flight control system guides the vehicle through the requested flight pattern by tracking the reference trajectory as computed by a real-time MPC-based optimization. This trajectory layer, which uses a constrained system model, provides an abstraction between the vision system and the vehicle. Both vision and flight control results are given from flight tests with an electric UAV.

1 Introduction

The concept of a high-endurance, situation-aware unmanned aerial vehicle (UAV) demands an onboard system that will actively sense its surroundings and make intelligent decisions over an extended period of time to accomplish its mission, with minimum intervention from remotely located human operators. In missions such as perch-and-stare reconnaissance, where a UAV lands at a strategic location and probes for certain events using its onboard sensors, VTOL-capable UAVs such as helicopters and tilt-rotor wings offer many advantages. For such missions, as well as in emergency situations, it is essential for an intelligent UAV to be able to autonomously locate, and then land at, a suitable landing site.

*Todd Templeton and S. Shankar Sastry are with the Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA: {ttemplet,sastry}@eecs.berkeley.edu. David Shim is with the Dept. of Aerospace Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea: hcshim@kaist.ac.kr. Christopher Geyer is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA: cgeyer@cs.cmu.edu.

Figure 1: An electrically-powered, MPC-controlled UAV testbed for vision-based landing and terrain mapping.

To execute a safe landing, a UAV first needs to find a level space that is free of debris. It is usually impossible or impractical to rely on a preloaded digital elevation map (DEM) due to its limited resolution and possible inaccuracy due to obstacle movement. UAVs are typically no larger than manned helicopters, and even if it could be assumed that the terrain would not significantly change a DEM accurate enough for such an application would have to have resolution to within a few inches. It is deemed unlikely that such an accurate DEM would be available *a priori*, and, even if one did exist, that such a DEM large enough to cover the mission area would be small enough to carry as part of the avionics. Therefore, it is desired that the UAV builds a DEM limited to the area of interest on the fly using its onboard sensors. Given UAVs' flying accuracy and their efficiency for the repetitive task of scanning a given area, aerial mapping using UAVs is also a useful capability in its own right.

For aerial ranging, laser scanners have been found to be both reliable and accurate sensors. When ranging at a high altitude, however, use of a laser scanner would require a high-energy beam. Such a sensor could be easily detected, and would impose unreasonable weight and power burdens on a small vehicle. Therefore, we choose to sense the terrain passively using cameras; in fact, due to our use of Geyer et al.'s Recursive Multi-Frame Planar Parallax (RMFPP) algorithm [1], we are able reconstruct terrain using only a single camera.

1-4244-0602-1/07/$20.00 ©2007 IEEE.

The vision landing problem has been addressed in many previous research projects, although many, including [2], [3], [4], [5], and [6], require an easily-recognizable landing target. No landing target is used in [7], although it is assumed that the visual axis of the camera is perpendicular to the ground and that the image contrast is higher at the boundary of obstacles than anywhere else. The approach most similar to ours is that of A. Johnson et al. at JPL [8], although their use of only two images at a time (a wide-baseline stereo pair over time from a single camera) restricts their 3D reconstruction accuracy at high altitudes.

When combined with the vision system, the MPC-based trajectory layer serves as a 'filter' to find a plausible trajectory that yields minimal cost from the reference trajectory. The vision and flight control systems are implemented on a UAV testbed (see Figure 1) based on a radio-controlled electric helicopter. Flight tests have shown that components of the vision-based landing and terrain mapping perform well when combined with the high-precision flight control system.

2 Vision-based Terrain Mapping and Analysis

The computer vision problem that we address in this project is one of 3D terrain reconstruction and analysis. In particular, we are trying to find suitable landing locations, i.e. regions that are large enough to safely land the helicopter that are flat and free of debris, have a slope of no more than 4 degrees, have been confidently mapped, (possibly) are similar to a desired constant appearance, and (optionally, not used in the experiments in this paper) contain a distinctive landing target[1]. Despite the availability of high-accuracy active technologies such as radar and LIDAR, we use a camera for this task because it is passive (and hence difficult to detect), and because such active technologies require high-powered beams that are energy- and weight-intensive for operation at high altitude.

The vision system (see Figure 2) consists of a feature tracking thread, which tracks distinctive image points through the image sequence and stores them in the feature repository; a motion stamping thread, which uses GPS/INS data and feature tracks to estimate the global position and orientation of the camera when each image was captured and to estimate the 3D locations of the tracked features (the latter of which are used to choose the best reference plane for the Recursive Multi-Frame Planar Parallax algorithm); and the mapping thread, which adds 3D points to its modular elevation and appearance map using the Recursive Multi-Frame Planar Parallax algorithm. The vision system also includes two interchangeable sets of external interfaces: in flight mode, it uses a custom Firewire capture thread, which stores timestamped captured images in a frame repository, and an external communication thread, which receives GPS/INS and other vehicle state data from, and sends desired trajectory information to, the flight control computer; in simulation/replay mode the Firewire capture thread is replaced by a custom simulation/replay thread, and all communication through the external communication thread is redirected to the simulation/replay thread.

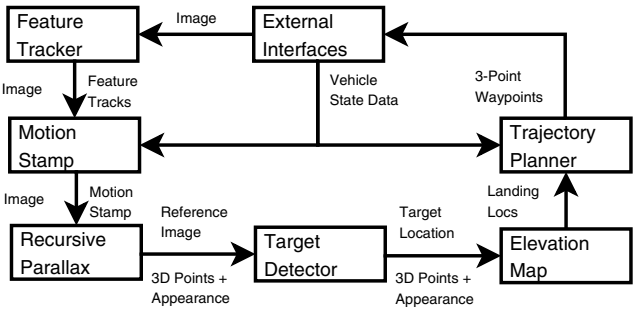

Figure 2: Vision system architecture. Note that the target detector is optional and was not used in these experiments.

2.1 The Recursive Multi-Frame Planar Parallax Algorithm

The cornerstone of our approach is Geyer et al.'s Recursive[2] Multi-Frame Planar Parallax (RMFPP) algorithm [1]. The RMFPP algorithm is a direct[3] method for obtaining dense[4] structure (terrain, in our case) estimates with corresponding appearance online in real time by using a single moving camera whose motion has been accurately estimated. We choose to use this single-camera method because of the inaccuracy inherent in estimating distant terrain using a stereo camera pair with a baseline that is attainable on the vehicle (see discussion in [1]), while using multiple images as the camera moves through space allows the RMFPP algorithm to attain expected range error that increases between linearly and with the square root of the range.

Suppose a camera takes images $i = 1, \ldots, m$ of a rigid scene, where image 1 is the reference view in which range will be estimated for each pixel. Then the homographies H_i

[1] The landing target is only used to enable the operator to choose a specific location if many locations are suitable—the vision system always ensures that all other requirements are satisfied.

[2] The cost of incorporating measurements from a new image depends only on the number of pixels in the image and does *not* depend on the number of images already seen.

[3] The algorithm expresses a cost function directly in terms of the image rather than depending on feature matching, and gradients of the cost function are calculated by linearization of the brightness constancy constraint (see pro: [9], con: [10]).

[4] The algorithm provides a depth estimate for every pixel that is within a sufficiently textured region.

that transfer the i-th view to the reference view via a chosen reference plane are given by:

$$\mathsf{H_i} \;=\; \mathsf{K}\left(\mathsf{R_i} - \frac{1}{d}\boldsymbol{T_i}\boldsymbol{N}^T\right)^{-1}\mathsf{K}^{-1} \in \mathbb{R}^{3\times 3}, \quad (1)$$

where $(\boldsymbol{N} \in \mathbb{R}^3, d \in \mathbb{R})$ are the unit normal of the reference plane in the coordinate system of the first camera and the perpendicular distance of the first viewpoint from the reference plane, $(\mathsf{R_i} \in SO(3), \boldsymbol{T_i} \in \mathbb{R}^3)$ are the rotation and translation from first camera coordinate system to the i-th one, and $\mathsf{K} \in SL(3)$ is the constant intrinsic calibration matrix of the camera.

Suppose that $\boldsymbol{X} \in \mathbb{R}^3$ is a point in space in the coordinate system of the first camera. Let $\boldsymbol{p_i} = (x_i, y_i)$ for $i = 1, .., m$ be the projection of $\boldsymbol{X}$ into each image, $\pi(x, y, z) = (x/z, y/z)$, and $\pi^*(x, y) = (x, y, 1)$. The quantity

$$\boldsymbol{p_1} \;-\; \underbrace{\pi\left(\mathsf{H_i}\,\pi^*(\,\boldsymbol{p_i}\,)\right)}_{\boldsymbol{p_i}'} \quad (2)$$

is called planar parallax, and is zero if $\boldsymbol{X}$ lies on the reference plane. The RMFPP algorithm uses planar parallax, which is small for small movements if $\boldsymbol{X}$ is close to the reference plane and increases with increased camera motion, to recursively estimate the quantity $\gamma = h/z$ for each pixel $\boldsymbol{p_1}$ in the reference image, where z is the range of $\boldsymbol{X}$ in the first view and $h = \boldsymbol{N}^T\boldsymbol{X} + d$ is the signed perpendicular distance of $\boldsymbol{X}$ from the reference plane. We then recover the range z using $z = -d/(\boldsymbol{N}^T\,\mathsf{K}^{-1}\,\pi^*(\,\boldsymbol{p_1}\,) - \gamma)$.

2.2 Modular Elevation and Appearance Map, and Landing Site Quality

The mapping system consumes the filtered (see discussion in [1]) 3D points generated by the RMFPP terrain reconstruction algorithm, and creates a consistent map in a world coordinate system, while simultaneously identifying candidate landing sites. The mapping system takes into account the following requirements: (1) efficient integration of incoming points (on the order of $50,000$ points per update, which occur at 0.375 Hz), each with a corresponding elevation variance (expected squared error) and appearance; (2) efficient recalculation of candidate landing sites after each update; (3) sufficient resolution for evaluating landing sites while fitting a memory budget and not hampering exploration (e.g. a fixed grid in memory is inadequate); (4) the ability to integrate and analyze data at a wide range of resolutions (due to the expected wide range of altitudes); and (5) minimal map recentering costs.

To obviate the need for map recentering and to increase memory efficiency with low overhead, the mapping system is modular in the sense that a fixed set of blocks are allocated for the map representation, and least recently accessed blocks are recycled as necessary. We achieve efficiency during updates by using weighted (by inverse elevation variance) sufficient statistics that represent the aggregate data, and with which we can quickly compute local planar fits and other metrics to rapidly evaluate candidate landing sites.

To efficiently store and analyze, at a wide range of resolutions, the terrain elevation, terrain elevation variance (expected squared error), and appearance data available for each 3D point, the map is represented as a 2D (x, y) grid with three layers (one for each type of data) at each of multiple resolutions. All blocks contain all resolutions; since blocks represent fixed areas of 2D space, higher resolutions in each block contain more pixels in each layer than lower resolutions. For operations that require only a single resolution, such as calculating landing quality and exporting maps, each map block independently chooses its highest resolution where at least a fixed percentage of pixels have known value (or its lowest resolution if none of its resolutions have enough pixels with known value).

The modular elevation and appearance map is designed to be robust. Because the scene is likely to change over long periods of time, blocks are reinitialized when they are revisited after no updates for a fixed period of time. To reduce the update and creation of blocks due to outliers, a fixed number of points from a given RMFPP update must be contained in an existing or potential block for it to be updated or created. To reduce the number of landing candidates generated by a suitable region, only block centers are considered as possible landing sites. To eliminate the trade-off between requiring large landing sites and having many mostly-unexplored blocks, and to allow more dense possible landing sites, the landing quality score of a given block is calculated over itself and a given radius of its neighbor blocks; this is implemented efficiently for all blocks in the area covered by a batch of points from the RMFPP algorithm by using integral images [11] over the sufficient statistics contained in the blocks.

To allow the efficient retrieval of arbitrary map blocks, all of the blocks that are in use are maintained in a hash table over their (x, y) centers; however, during normal operation, most blocks are retrieved using links from one of their neighbors (in addition to being accessible through the hash table, each block is bidirectionally linked to each of its immediate neighbors). All existing blocks that are required for the given list of 3D terrain and appearance points produced by the RMFPP algorithm are retrieved immediately upon determination of the points' 2D bounding box. Adding each 3D point to the map involves creating or locating the proper map block in the prefetched grid and then updating the closest 2D map block pixel at each resolution, i.e. optimally updating the pixel at each layer based on the exist-

ing elevation variance at the pixel and the elevation variance for the new 3D point as provided by the RMFPP algorithm. When a 2D pixel is updated in a given block and resolution, its previous value is subtracted from, and its new value is added to, each statistic that is maintained for that block and resolution.

The landing quality score for each modified block (combined with its radius of neighbors) is a linear combination of the angle of the best-fit plane from horizontal, the plane fit error, the percentage of grid squares in the block with unknown value, the difference between the average appearance and a given desired appearance, the appearance variance, and the average target quality (optional, not used in the experiments in this paper). A block is considered to be a landing candidate if it is below given thresholds on each element of the landing quality equation, and landing candidates are maintained in a priority queue so that, all other things being equal, the better (lower 'landing quality' value) candidates are considered first.

2.3 High-level Planner

Concurrently with the above vision algorithms, the vision system executes a high-level planner that operates directly on the vehicle state data. The default plan (when no landing site candidates have been identified within a given maximum radius of the current location) is an outwardly-expanding box search centered around the point, and at the altitude of, where the planner is initially enabled (see Figure 3). When a landing site candidate is identified that is within the given maximum radius, the planner enters a mode where it directs a descending spiral toward a point a fixed distance directly over the candidate site. The candidate site is examined whenever it is visible during the downward spiral, and all other visible locations are also examined at closer range during this process. At any time during the spiral, the vision system may determine that the site is unsuitable, or the human operator may signal that the site is unsuitable, and the planner will switch to a different candidate site (or return to the default box search plan if there are no nearby candidate sites). Once the helicopter reaches a point a fixed distance directly over the candidate site, the human operator may approve an autonomous landing, at which time the planner directs a constant-speed vertical descent to a fixed lower altitude AGL, followed by a slower constant-speed vertical descent to the ground.

2.4 A Note on Motion Stamping

Although several algorithms, such as Chiuso et al.'s '3-D Motion and Structure from 2-D Motion Causally Integrated over Time' [12], are available to estimate camera attitude and position in real time, the accuracy of these algorithms, even when extended to incorporate (noisy) camera attitude and position measurements from the vehicle state data[5], has thus far proven to be insufficient for our needs. While we continue to experiment with alternative motion-stamping methods, for the purposes of this paper we separated the experiment into three phases to showcase the performance of the other components: In the first phase, the vision high-level planner directed the helicopter to perform the box search while it collected image data. In the second phase, we performed SIFT [13] feature tracking and Sparse Bundle Adjustment (SBA) [14] for camera localization, which we initialized using the previous GPS/INS datapoint, followed by RMFPP running on similar hardware to the helicopter vision computer in better than real time. In the third phase, we executed the closer inspection and landing maneuver using a premade elevation and appearance map.

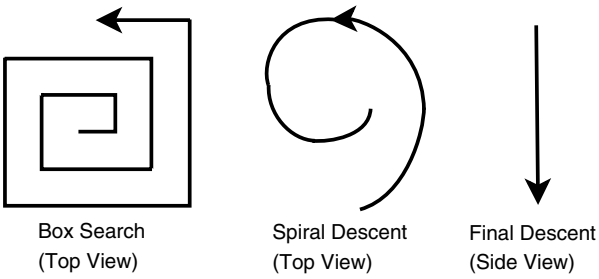

Figure 3: Plans for the high-level planner.

3 Flight Control System

The flight control system consists of two hierarchical layers: the trajectory generator and the tracking controller. The trajectory generator is based on model predictive control (MPC), which solves for the optimal control input and the associated vehicle trajectory. The tracking controller guides the vehicle to follow the given trajectory with minimal error using the output from the trajectory generation layer.

3.1 MPC-based Trajectory Generation

In [15, 16, 17], it is shown that model predictive control using penalty functions for state constraints and explicit input saturation is a viable approach to address the guidance and control problems of UAVs at a reasonable computational load for real-time operation. In [16], an MPC-based control system is shown to have outstanding tracking performance in the presence of coupled dynamic modes and substantial

[5]The vehicle state data contains measurements of the GPS/INS attitude and position, which we adjust by the constant rotation and translation between the coordinate system of the GPS/INS system and the coordinate system of the camera. Note that the state data measurement is not synchronized with the imagery.

model mismatch. It has also been demonstrated that MPC-based optimization can be formulated to implement a higher level of autonomy, such as real-time aerial collision avoidance [15], and obstacle avoidance in an urban environment using an onboard laser scanner [17]. In [18], an MPC-based trajectory planner is implemented as the feedforward control part of a two-degree-of-freedom control system. Here, as in [18], we use a full vehicle kinodynamic model with input saturation.

Suppose we are given a time-invariant nonlinear dynamic system

$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{3}$$

where $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^{n_x}$ and $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^{n_u}$. The optimal control sequence over the finite receding horizon N is found by solving the nonlinear programming problem

$$V(\boldsymbol{x}, k, \boldsymbol{u}) = \sum_{i=k}^{k+N-1} L(\boldsymbol{x}(i), \boldsymbol{u}(i) + F(\boldsymbol{x}(k+N)), \tag{4}$$

where L is a positive definite cost function term and F is the terminal cost. When applied to the vision-based landing problem, L contains a term that penalizes the deviation from the desired trajectory. Suppose $\boldsymbol{u}^*(\boldsymbol{x}, k)$ is the optimal control sequence that minimizes $V(\boldsymbol{x}, k, \boldsymbol{u})$ such that $V^*(\boldsymbol{x}, k) = V(\boldsymbol{x}, k, \boldsymbol{u}^*(\boldsymbol{x}, k))$, where $V^*(\boldsymbol{x}, k) \leq V(\boldsymbol{x}, k, \boldsymbol{u})$, $\forall \boldsymbol{u} \in \mathbb{U}$. With $\boldsymbol{u}^*(k)$, we can find $\boldsymbol{x}^*(k)$, $k = i, \cdots, i+N-1$ by recursively solving the given nonlinear dynamics with $\boldsymbol{x}(i) = \boldsymbol{x}_0(i)$ as the initial condition. The obtained $\{\boldsymbol{x}^*(k), \boldsymbol{u}^*(k)\}$ are used as the reference trajectory and feedforward control input, respectively, in the following control law:

$$\boldsymbol{u}(k) = \boldsymbol{u}^*(k) + K(\boldsymbol{x}^*(k) - \boldsymbol{x}(k)). \tag{5}$$

For the feedback control gain K, a multi-loop proportional-differential (MLPD) loop is implemented as depicted in Figure 4, a control strategy similar to that in [19]. If the dynamic model used for solving the optimization problem perfectly matches the actual dynamics and the initial condition without any disturbance or model mismatch, there should not be any tracking error. In the real world, such an assumption cannot be satisfied. Using this approach, with a tracking feedback controller in the feedback loop, the system can track the given trajectory reliably in the presence of disturbance or modeling error.

In the following subsection, we will show how the MPC-based trajectory generator is integrated with the high-level planner.

3.2 Flight Control Strategy for Autonomous Vision-based Landing

For automatic surveying, the control system should be able to guide the vehicle through the requested waypoints with minimal deviation while keeping the vehicle within its dynamic performance boundary. The flight control should not induce any excessive vibration or rapid turns that may cause motion blur in the camera image.

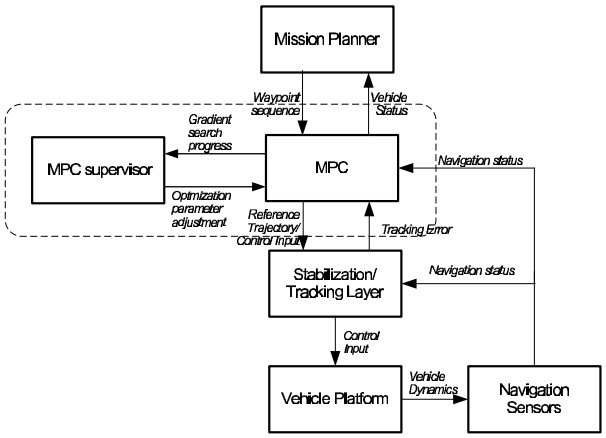

Figure 4: Flight control system architecture with MPC-based trajectory generator.

The vision system requests one of three types of flight patterns as described in Figure 3. During the box search for coarse DEM construction, the vision system sends waypoints that are the vertices of piecewise linear segments. The vehicle is expected to fly along the linear segments with minimal deviation while making bank-to-turn maneuvers around each vertex at a constant horizontal cruise speed and altitude. During the spiral descent, the vision unit sends a series of waypoints with much finer resolution. In final descent mode, the vision unit sends the landing coordinates to the flight control system.

Since the vision system requires a smooth flight for high-quality image acquisition, the vehicle needs to fly at a constant velocity and a reasonable yaw rate to avoid motion blur of the image. In particular, in order to achieve a smooth transition around a waypoint with a constant cruise speed, we have to know the next waypoint *a priori* while the vehicle is approaching the current waypoint so that the flight control system can prepare for the bank-to-turn maneuver without any abrupt changes in heading or cruise velocity. Therefore, the trajectory planner needs to know the next waypoint as well as the current waypoint so that it can plan ahead around the waypoint. For additional flexibility, the high-level planner may also specify the previous waypoint, which is not required to have been the actual previous waypoint; it is merely a point that defines a vector of approach to the current waypoint. When a new waypoint request is received from the high level planner, regardless of the flight mode it requests, the given waypoints are initially connected with linear segments. The high level planner also specifies the reference horizontal and vertical velocity, which is

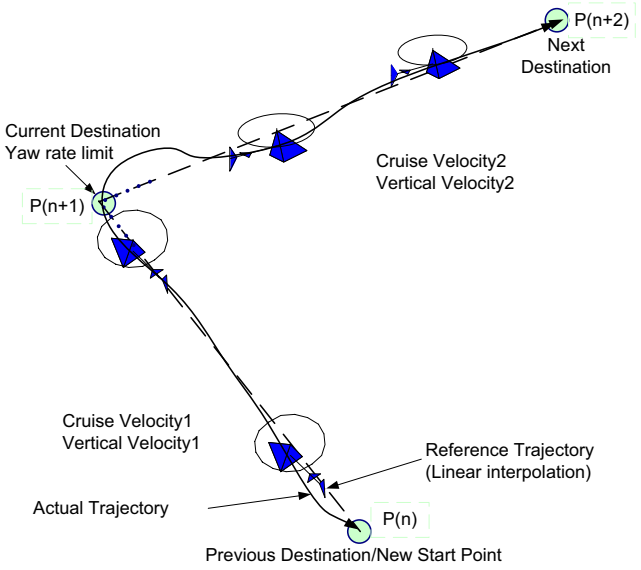

Figure 5: Three-point waypoint specification and MPC-based trajectory generation.

used for sampling the reference points over the linear segments. The resulting reference trajectory is a sequence of (x, y, z) coordinates associated with the reference heading. The given reference points are used to compute the cost function in Equation (4), which is solved in real time using a highly efficient optimization algorithm [20] at every sample time. The resulting reference trajectory and the optimal control input are sent to the tracking controller (see Figure 4).

To facilitate the guidance algorithm introduced above, the vision computer periodically sends the following to the flight computer over a wired RS-232 channel: past/current/future waypoints (see Figure 5), horizontal/vertical speed limits, heading rate limit, flight mode (Abort, Box Search, Spiral, Landing), and a time tag.

Although the MPC-based approach in this paper creates a heavy numerical load, the algorithm is simple and straightforward in principle, and it protects the vehicle from any infeasible waypoint requests from the higher level planner that might outherwise push the vehicle beyond its dynamic capability. With careful design of the multi-process flight control software, and with careful selection of the horizon length, iteration gain, and step size of the gradient search as discussed in [18], the algorithm comfortably runs in real time on a Pentium III 700MHz machine.

4 System Setup

As described above, the vision system requests appropriate flight patterns as it performs real-time terrain mapping and analysis at varying resolutions. The flight control system

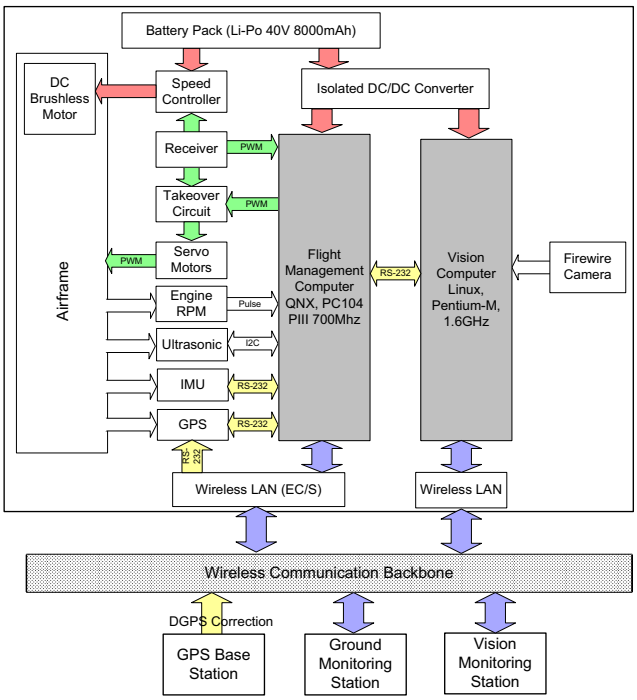

Figure 6: System architecture of the UAV testbed for vision-based landing and mapping.

is responsible for guiding the vehicle through the requested waypoints with an acceptable accuracy. At a constant rate of 10 Hz, the flight control system reports time-stamped navigation data such as position and attitude/heading, which are necessary to reconstruct the terrain with respect to an inertial reference frame (see Figure 6), to the vision system.

4.1 Vision Processing Unit

The vision-based terrain mapping and analysis system is implemented on a PC104 form factor Pentium M computer running Linux. The CPU is interfaced with a 2GB Compact Flash drive, a Firewire board, and PCMCIA wireless Ethernet. As shown in Figure 1, the vision computer is installed in the nose of the vehicle in a dedicated aluminum enclosure for modularity and EMI shielding. A Firewire camera is installed in a forward- and downward-looking direction to capture the ground ahead of and below the vehicle. The vision system receives vehicle state data from, and sends trajectory commands to, the flight computer through the RS-232 serial interface.

4.2 UAV Hardware

The testbed used in this research is based on an electrically powered RC helicopter, whose detailed specifications are given in the Table 1. The DC brushless motor with

Base platform	Electric Helicopter (Maxi-Joker)
Dimensions	0.26 m (W) x 2.2 m (L) x 0.41 m (H)
Rotor Diameter	1.8 m
Weight	4.5 kg (no onboard electronics) 7.5 kg (fully instrumented)
Powerplant	Actro 32-4 motor (1740W max at 75A) Lithium-Ion-Polymer (10S4P; 40V 8Ah)
Operation Time	Up to 15 minutes
Avionics	Navigation: DGPS-aided INS GPS: NovAtel OEM4-MillenRT2 IMU: Inertial Science ISIS-IMU Flight Computer: PC104 Pentium III 700MHz Communication: IEEE 802.11b with RS-232 multiplexing Vision Computer: PC104 Pentium M 1.6GHz
Autonomy	Waypoint navigation with automatic VTOL Position-tracking servo mode MPC-enabled dynamic path planning with collision avoidance Stability-augmentation system

Table 1: Specification of the UAV testbed.

high-capacity Lithium-polymer batteries allows more than 10 minutes of continuous flight with the ease of fully automatic start-stop operation. The onboard components are designed and integrated with an emphasis on weight reduction for longer flight time, reliability, and maneuverability. The vehicle is controlled by a PC104 form factor Pentium III 700MHz CPU with a custom servo interfacing board, an inertial measurement unit (IMU), a high-precision carrier-phase differential global positioning system, and an IEEE 802.11b device (see Figure 6). The flight control system communicates with the ground station over the wireless channel for receiving commands and sending navigation status data and system vital signs such as the battery level and the health of onboard components.

5 Experimental Results

In this section, we present experimental results for the vision-based landing and terrain mapping experiment. Figure 7 shows the result of terrain reconstruction using experimental aerial imagery that was geo-registered offline and then processed using the RMFPP algorithm in real time on hardware similar to that in the vehicle vision computer. Although much of the left side of the view is flat and uneventful (as it should be), the trees in the middle of the view and the road on the right are clearly visible.

In order to build the terrain map or investigate a candidate landing site, the vision computer commands the vehicle to perform a box search or closer inspection spiral by requesting waypoints following the format defined in Section 4.2. The actual flight trajectory and other navigation states are presented in Figure 8. The vehicle initially follows the waypoints in a box pattern, and then, upon discovering a possible landing zone, the vehicle is commanded to fly in a spiral pattern at a very low velocity. After completing the closer inspection, the vision computer commands the vehicle to abort the spiral and resume the box search at normal flight speed. As shown in the figure, the helicopter follows this flight sequence with acceptable accuracy.

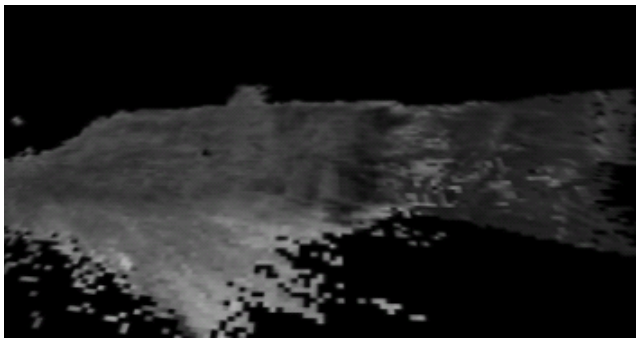

Figure 7: Real images vision experiment: The reconstructed appearance draped over the reconstructed 3D terrain.

6 Conclusion

This paper has presented a vision-based terrain mapping and analysis system, and a hierarchical flight control system based on the model predictive control (MPC) approach, for autonomous rotorcraft landing. The algorithms were implemented using COTS components on an electrically-powered helicopter for experimental validation. We observed that the RMFPP algorithm required higher camera localization accuracy than we were able to achieve using existing algorithms, but we were able to show accurate real-time results for other components of the vision system when the camera localization was computed offline. The flight control algorithm based on real-time MPC was shown to be a viable approach for producing plausible trajectories in real time using the identified vehicle model, and the feedback control law was shown to provide added protection against disturbance and model mismatch.

7 Acknowledgement

The authors would like to thank Travis Pynn, Hoam Chung, Jonathan Sprinkle, and Mikael Eklund for their the contributions in support on the experiments in this paper. This work was funded by the following grants: ARO DAAD 19-02-1-0383 and Boeing SEC BAI-Z40705R.

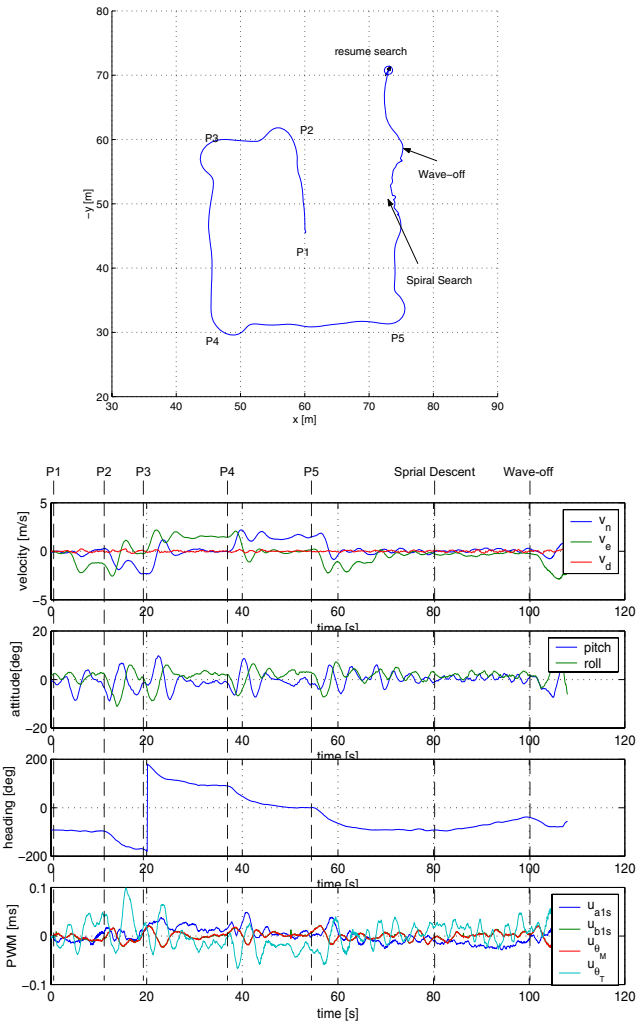

Figure 8: Flight test result for terrain reconstruction.

References

[1] C. Geyer, T. Templeton, M. Meingast, and S. Sastry, "The recursive multi-frame planar parallax algorithm," in *Proceedings of Third International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.

[2] O. Shakernia, Y. Ma, T. Koo, and S. Sastry, "Landing an unmanned air vehicle: Vision based motion estimation and nonlinear control," 1999.

[3] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Visually-guided landing of an unmanned aerial vehicle," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 371–381, Jun 2003.

[4] S. Saripalli and G. S. Sukhatme, "Landing on a moving target using an autonomous helicopter," in *Proceedings of the International Conference on Field and Service Robotics*, Jul 2003.

[5] S. Saripalli, G. S. Sukhatme, and J. F. Montgomery, "An experimental study of the autonomous helicopter landing problem," in *Proceedings, International Symposium on Experimental Robotics*, (Sant'Angelo d'Ischia, Italy), 2002.

[6] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle," in *IEEE International Conference on Robotics and Automation*, pp. 2799–2804, 2002.

[7] P. J. Garcia-Pardo, G. S. Sukhatme, and J. F. Montgomery, "Towards vision-based safe landing for an autonomous helicopter," *Robotics and Autonomous Systems*, vol. 38, no. 1, pp. 19–29, 2001.

[8] A. Johnson, J. Montgomery, and L. Matthies, "Vision guided landing of an autonomous helicopter in hazardous terrain," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2005.

[9] M. Irani and P. Anandan, "About direct methods," in *Proc. International Workshop on Vision Algorithms*, September 1999.

[10] P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Proc. International Workshop on Vision Algorithms*, September 1999.

[11] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of Computer Vision and Pattern Recognition*, 2001.

[12] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "3-d motion and structure from 2-d motion causally integrated over time: Implementation," in *Proceedings of European Conference on Computer Vision*, June 2000.

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, 2004.

[14] M. Lourakis and A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm," Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[15] D. H. Shim, H. J. Kim, and S. Sastry, "Decentralized nonlinear model predictive control of multiple flying robots," in *IEEE Conference on Decision and Control*, December 2003.

[16] D. H. Shim, H. J. Kim, and S. Sastry, "Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles," in *American Control Conference*, May 2002.

[17] D. H. Shim, H. Chung, and S. Sastry, "Conflict-free navigation in unknown urban environments," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 27–33, September 2006.

[18] D. H. Shim and S. Sastry, "A situation-aware flight control system design using real-time model predictive control for unmanned autonomous helicopters," in *AIAA Guidance, Navigation, and Control Conference*, August 2006.

[19] D. H. Shim, *Hierarchical Control System Synthesis for Rotorcraft-based Unmanned Aerial Vehicles*. PhD thesis, University of California, Berkeley, 2000.

[20] G. J. Sutton and R. R. Bitmead, "Computational implementation of NMPC to nonlinear submarine," *Nonlinear Model Predictive Control*, pp. 461–471, 2000.