Monocular Visual Mapping for Obstacle Avoidance on UAVs

Daniel Magree, John G. Mooney, Eric N. Johnson

***Abstract*— An unmanned aerial vehicle requires adequate knowledge of its surroundings in order to operate in close proximity to obstacles. UAVs also have strict payload and power constraints which limit the number and variety of sensors available to gather this information. It is desirable, therefore, to enable a UAV to gather information about potential obstacles or interesting landmarks using common and lightweight sensor systems. This paper presents a method of fast terrain mapping with a monocular camera. Features are extracted from camera images and used to update a sequential extended Kalman filter. The features locations are parameterized in inverse depth to enable fast depth convergence. Converged features are added to a persistent terrain map which can be used for obstacle avoidance and additional vehicle guidance. Simulation results and results from recorded flight test data are presented to validate the algorithm.**

I. Introduction

Unmanned aerial vehicles (UAVs) have the potential to help with a variety of complex tasks, such as search and rescue, assessing disaster areas, and gathering intelligence. To be successful at these tasks, UAVs must operate and interact in challenging environments. Environments such as groups of buildings, urban cityscapes, rugged mountainous terrain, and building interiors, pose collision threats to unmanned aerial vehicles, as well as make it difficult to accomplish mission objectives. UAVs must gather information about the environment with its limited sensor payload in order to accomplish mission goals in these cases. This information could take many forms, but one of the more useful forms is that of a terrain map of the area of interest. The terrain map of the environment not only enables obstacle avoidance but also helps in accomplishing mission goals, such as disaster assessment, and search and rescue.

Monocular cameras are standard payload of many UAVs, and so it is natural to turn to these sensors when considering the problem of terrain mapping. Cameras provide a wealth of information about the environment, but processing this information can have a high computational cost. Efficient algorithms are necessary to enable realtime processing.

This paper proposes an efficient monocular vision-based feature estimation algorithm for fast terrain mapping. Feature measurements from the camera images are used to update the feature states in an extended Kalman filter (EKF). The algorithm uses inverse depth parameterization of the feature points for fast depth convergence, and stores converged points in an altitude map of the local area. The altitude map is used in a obstacle avoidance routine.

The system presented here is the mapping portion of the full simultaneous localization and mapping (SLAM) problem. Monocular vision-only SLAM was first developed by Davison in [1]. The slam problem was formulated as an extended Kalman filter which propagated the entire covariance matrix for all stored features. Improvements have been made to the original algorithm, most importantly the use of inverse depth parameterization of the feature points [2]. The inverse depth parameterization stores the features as a six-element vector consisting of an anchor point, azimuth and elevation to the feature, and inverse distance to the feature. The inverse depth formulation has the dual advantage of being more nearly gaussian in uncertainty for points of unknown depth, and being able to handle points at infinite distance.

Other mapping paradigms exist besides the EKF

Graduate Research Assistant, Georgia Institute of Technology, dmagree@gatech.edu.

Graduate Research Assistant, Georgia Institute of Technology, john.g.mooney@gatech.edu.

Associate Professor, Georgia Institute of Technology, eric.johnson@ae.gatech.edu.

framework. In particular, the SLAM algorithm presented in [3] operates by processing a full bundle adjustment solution in parallel with feature-point tracking and has been very successful in producing large maps over great distances. This algorithm has been implemented on a quadrotor vehicle in [4]. However, they report that during bundle adjustment step, long delays may occur as the map grows. We think that to avoid potentially disastrous delays, it is critical that map updates occur in constant time, as happens in the EKF formulation we present here.

Previous work in monocular vision-based obstacle avoidance have tackled the problem in a variety of ways. In [5], it is assumed that a depth map is given from the camera and image processing. Pixels are grouped using K-mean clustering, and then obstacles are estimated in a vehicle fixed frame using an EKF. Tracking discrete obstacles reduces the number of states needed in the estimator but may cause small obstacles to be overlooked. Another approach was taken in [6], in which the optical flow and bearing is used as a measurement for sectors of an image, and depth is estimated based on a linearized model.

The use of stereo camera sensors allows range to be computed from a single frame, thereby simplifying the estimation process. Several recent papers have developed obstacle avoidance systems using such a sensor. In [7] a combination of stereo cameras and optical flow is used to allow a vehicle to avoid obstacles in urban canyon environments. The paper presents results using a small helicopter which successfully identifies obstacles in front. Other work presented in [8] use stereo vision in conjunction with laser range finders to navigate riverine environments in the presence of intermittent GPS. The use of stereo cameras are convenient for this problem, but such a payload is not as common as a single camera on UAVs. Furthermore, the range measurement is limited by the baseline of the stereo pair, and beyond that distance the system is equivalent to a monocular system.

The work in the paper is influenced by our previous work in vision-based obstacle avoidance and navigation. In [9], a EKF formulation with Cartesian point parameterization is used to estimate the location of a small number of obstacles and initiate avoidance. Our work extends these results by allowing a greater number and variety of points to be tracked in a given frame and the results to be stored in a persistent altitude map for later use. In [10], a fully independent vision-based navigation system was developed with extended flight test results, demonstrating the ability to simultaneously map and localize a vehicle using the EKF framework. This was accomplished using a downward facing camera in a scenario in which uncertainty in depth was less important than bounding vehicle position. In the current work we require depth to converge quickly and are concerned with objects in the path of the vehicle, and therefore have greater uncertainty in depth at the same time as a greater need for depth to converge quickly. For this reason inverse depth parameterization has been implemented on this system. The obstacle avoidance method used in this system is developed in [11], and the terrain mapping is developed in [12].

The rest of the paper is organized as follows. First, the terrain mapping system is presented. Details of the inverse depth parameterization are described, and the sequential EKF is presented. Next, feature initialization is presented and the role of the terrain map is explained. Finally, simulation results and results using recorded flight test data are shown which validate the algorithm and compare its performance to Cartesian parameterized estimator.

II. EKF Terrain Mapping

This section presents an overview of the vision-based mapping algorithm.

The vision-based mapping system uses an extended Kalman filter to estimate the location of image features in the environment. During operation, images are captured and features are extracted from the images. The features from the first image are used to initialize a database of point locations. When each subsequent image is captured, features are extracted and matched to the points stored in the database, and the error between their predicted and measured location is used to update their location via the extended Kalman filter update equations. Points that have converged in depth are

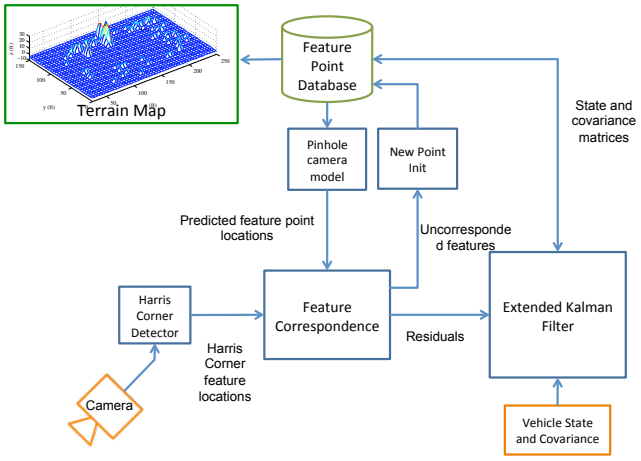


Fig. 1. A block diagram describing the vision-based mapping system. Pixel location of features are corresponded to predicted locations of database points. The resulting residual is fused with vehicle state and covariance to estimate the feature locations and associated covariance matrices for the features. Converged features are used to update the terrain map. Uncorresponded features can be used to update the database as current points leave the field of view.

used to update the terrain map. Points that are no longer visible due to vehicle motion are discarded from the estimator, and unmatched measurements are initialized into the database. A schematic of the algorithm is shown in Figure 1.

A. *Inverse depth parameterization*

Key to the success of the vision-based mapping algorithm for obstacle avoidance is fast convergence of tracked points in depth. Without fast convergence, a UAV does not have time to take action to avoid the approaching obstacle safely. The use of inverse depth parameterization enables convergence at a faster rate than Cartesian parameterization because the uncertainty in inverse depth is more nearly Gaussian than standard depth. The discrepancy between actual uncertainty and modeled uncertainty in standard parameterization requires that measurements are assumed to have larger covariances than is actually the case, slowing estimation. By using a more accurate uncertainty model, points converge faster to their true value.

The parameterization used in this system is based on the work in [2]. Database features are defined by a six-state vector. The vector is composed of an anchor point, given by a Cartesian location in the local inertial frame, $[x_a, y_a, z_a]^{\mathrm{T}}$, the azimuth, ψ, and elevation, θ, and the inverse distance, ρ, from the feature to the anchor point. The anchor is chosen to be the location of the vehicle coordinate system at the time of database point initialization. The full estimated state vector for a feature is given by

$$\hat{y} = \begin{bmatrix} \hat{x}_a & \hat{y}_a & \hat{z}_a & \hat{\psi} & \hat{\theta} & \hat{\rho} \end{bmatrix} \tag{1}$$

where $\hat{\cdot}$ indicates estimated quantities. Angles (ψ, θ) are measured with respect to a plane and axis in $\mathbb{R}^3$, which in this work was chosen to be the inertial y-z, with $(0, 0)$ lying along the z axis.

The vehicle state vector is expressed using a minimal representation of the attitude quaternion (see [10], [13]) and is given by.

$$\hat{x} = \begin{bmatrix} \hat{p} & \hat{v} & \hat{R} & \hat{s}_b & \hat{\omega}_b \end{bmatrix}^{\mathrm{T}} \tag{2}$$

where p is the position of the vehicle in the inertial frame, v is the velocity, R is the quaternion error given by $\delta q = q \otimes \hat{q}^{-1} = \begin{bmatrix} 1 & \hat{R} \end{bmatrix}^{\mathrm{T}}$, and s_b and ω_b are the accelerometer and gyroscope biases, respectively.

Within the measurement model, the location of the point is expressed as a vector normalized by length in terms of the vehicle state and feature state:

$$\hat{h}^c = L_{ci} \left[\hat{\rho} \left(\begin{bmatrix} \hat{x}_a \\ \hat{y}_a \\ \hat{z}_a \end{bmatrix} - \begin{bmatrix} \hat{p}_x \\ \hat{p}_y \\ \hat{p}_z \end{bmatrix} \right) + m(\hat{\psi}, \hat{\theta}) \right] \tag{3}$$

where $\hat{p} = \begin{bmatrix} \hat{p}_x & \hat{p}_y & \hat{p}_z \end{bmatrix}^{\mathrm{T}}$ is the estimated location of the vehicle, $m(\hat{\psi}, \hat{\theta})$ is the unit vector in the direction of the feature from the anchor point, and L_{ci} is the rotation matrix from inertial to camera frame. If the camera is not located at the origin of the vehicle reference frame, an appropriate translation is also applied. This formulation allows for the estimator to handle features at infinite depth, in which case $\hat{\rho} = 0$.

The measurement model is the standard pinhole camera model, but expressed in terms of the normalized direction vector $h^c = \begin{bmatrix} h_x & h_y & h_z \end{bmatrix}^{\mathrm{T}}$:

$$z = \begin{bmatrix} u \\ v \end{bmatrix} = g(x, y) = \begin{bmatrix} f_u \frac{h_y}{h_x} + \eta_u \\ f_v \frac{h_z}{h_x} + \eta_v \end{bmatrix} \tag{4}$$

where u and v give the image location, f_u and f_v are the horizontal and vertical focal lengths,

respectively, and it is assumed $\eta_u \sim N(0, R_u)$ and $\eta_v \sim N(0, R_v)$. Note $R_{img} = \text{diag}(R_u, R_v)$. An undistortion transform is applied to the image features before being used in the algorithm, and so a simple camera model is appropriate.

B. State and covariance update

The mapping algorithm performs the state update in a sequential manner. Each feature point is treated as an independent measurement, and correlations between the measurement and state are ignored. At each time step, a subset of the feature database is observed and used to update the feature state. During each feature update j, the Jacobian of the measurement model, C_k, is calculated at the current state estimate:

$$C_{j,k} = \left[\begin{array}{cc} \frac{\partial z_j}{\partial \hat{x}} & \frac{\partial z_j}{\partial \hat{y}_j} \end{array}\right]_{(\hat{x},\hat{y}_j)=(\hat{x},\hat{y}_j)_{k|k-1}} \tag{5}$$

The state and covariance update is calculated according to the familiar EKF update equations. The combined covariance for the feature point j and vehicle state is given by

$$P_j = \begin{bmatrix} P_x & 0 \\ 0 & P_{y_j} \end{bmatrix} \tag{6}$$

where P_x is the covariance matrix for the state vector, provided by the navigation solution, and P_y is the covariance of the feature state. Then,

$$K_{j,k} = P_{y_j,k|k-1} C_{j,k}^{\mathrm{T}} \left(C_{j,k} P_{jk|k-1} C_{j,k}^{\mathrm{T}} + R_{img}\right)^{-1} \tag{7}$$

$$\hat{y}_{j,k|k} = \hat{y}_{j,k|k-1} + K_{j,k}(z_j - g(\hat{x}, \hat{y}_j)_{k|k-1}) \tag{8}$$

$$P_{y_j,k|k} = (I - K_{j,k} C_{j,k}) P_{y_j,k|k-1} \tag{9}$$

C. Point Matching

Before extracted features can be used in the EKF, it must first be determined which database point they correspond to. A maximum-likelihood approach is used, where database points are corresponded to the measurement with the closest Mahalanobis distance. While other more robust methods, matching of point descriptors may be used, we have found that this method provides adequate matching results for the estimator.

The feature detector is a modified Harris corner detector, which extracts points based on the image gradient in orthogonal directions[14]. To ensure good feature separation, the detector partitions the image into bins, and sets a maximum number of features in each bin. Also, a minimum distance between features can be set. The feature locations are passed to the mapping algorithm in order of their Harris corner score, up to a maximum number.

The search region is determined using the estimated location uncertainty:

$$S^j = (C_j P_j C_j^{\mathrm{T}} + R_{img}) \tag{10}$$

The statistical distance between measurement z_i and the image plane location of point $\hat{y}_j$ is given by

$$e_{ij} \triangleq z_i - g(\hat{x}, \hat{y}_j) \tag{11}$$

$$Z_{ij} = e_{ij}^{\mathrm{T}} (S^j)^{-1} e_{ij} \tag{12}$$

A maximum value, Z_{lim}, is set for Z_{ij} for measurement i and database point j to be considered matches. If they are below this threshold and database point j is not closer to any other measurement, i is considered to be a measurement of j.

If a measurement z_i does not meet the conditions to match any database point, it is considered to be a measurement of no currently stored point and is uncorresponded. Uncorresponded points can be used to initialize new database points, as described in the following section.

D. Initialization of new points

Initialization of new points into the database is performed by setting the anchor point to be the current vehicle position, and deriving the azimuth and elevation from straightforward trigonometric relations of the pinhole camera model. The inverse depth is set such that the database point is initialized at the best estimate of the altitude of the terrain under the vehicle.

$$\begin{bmatrix} x_a & y_a & z_a \end{bmatrix}^{\mathrm{T}} = p \tag{13}$$

$$\theta = \text{atan2}\left(h_x^i, \sqrt{(h_y^i)^2 + (h_z^i)^2}\right) \tag{14}$$

$$\psi = \text{atan2}(h_y^i, h_z^i) \tag{15}$$

$$\rho = \frac{1}{||r||} = \frac{|h_z^i|}{|p_z|} \tag{16}$$

where $h^i = L_{ic} h^c$.

Initialization of the covariance of the point is dependent on the measurement covariance R_{img}, the state covariance P_x, and the depth covariance prior σ_ρ^2.

$$P_{y_j} = \frac{\partial y_j}{\partial z_i} R_{img} \frac{\partial y_j}{\partial z_i}^{\mathrm{T}} + \frac{\partial y_j}{\partial \rho} \sigma_\rho^2 \frac{\partial y_j}{\partial \rho}^{\mathrm{T}} + \frac{\partial y_j}{\partial x} P_x \frac{\partial y_j}{\partial x}^{\mathrm{T}} \tag{17}$$

Feature removal from the database is determined by comparing a "confidence index" of the current database features with the initialization confidence index of a new feature. The confidence index for current database features is limited to a minimum and maximum of 0 and 100, and is incremented for every iteration a feature is corresponded to a measurement, and decremented if it is not corresponded. The index can be thought of as a measure of how often a database feature has been used (corresponded) for navigation in the recent past. The initialization confidence index is related to the number of database features which were corresponded on the current iteration. A database feature is replaced if its confidence index falls below the initialization confidence index. When few features are corresponded, the initialization confidence index is high, and features are replaced quickly. When many features are corresponded, the initialization confidence index is low and features are replaced slowly. The initialization confidence index represents a dynamic point removal threshold modified by the number of correspondences in the current iteration.

E. Terrain map

A key element of the success of this algorithm is the dynamic terrain map. The EKF has a limited number of features it can estimate at a time, due to the increasing computational burden. In this work, the number of features estimated in the filter is limited to 50, far too few to make a large map of an area. To overcome this problem, the location of converged points in the estimator are used to update a terrain map, which retains their information even after the points are discarded from the estimator.

The terrain map is stored as an evidence grid of altitudes. For each location in the x-y plane, the altitude of the location is updated based on point data from the estimator. When a converged point is received, the height of the location in the grid is raised or lowered. More details on the terrain map can be found in [12].

III. Obstacle Avoidance

The terrain map generated by the filter is used in the obstacle avoidance algorithm. The obstacle avoidance algorithm operates by considering the section of the map in the direction of the velocity vector in the horizontal plane. The altitude of the map at each grid square out to a specified distance is compared to the specified minimum altitude. If this minimum altitude will be violated in the current trajectory, then a smooth pull-up maneuver is performed. In a similar way, the algorithm can also cause the vehicle to quickly return to the desired altitude after passing the obstacle.

The height restriction h_{min_i} imposed by each grid square i in the map along the vehicle path is given by the desired clearance altitude minus altitude which can be reached in the time to get to the grid location.

$$h_{min_i} = h_i + h_c - 0.5 a_c \Delta t_i^2, \tag{18}$$

where h_i is the height of location i, h_c is the clearance height, a_c is the vertical acceleration for avoidance, and Δt_i is the time until the vehicle is within the specified horizontal miss distance of location i, i.e. the time to collision.

The vertical speed is calculated such that a smooth pull-up and push-over is achieved:

$$\dot{h}_{min_i} = -\sqrt{4 a_c (h - h_i - h_c) + 2 (a_c \Delta t_i)^2} + a_c \Delta t_i, \tag{19}$$

where h is the current altitude of the helicopter. If the aircraft is low enough that the aircraft cannot smoothly pull up at the required acceleration level, the commanded velocity is set to the maximum allowable at the desired acceleration:

$$\dot{h}_{min_i} = -a_c \Delta t_i \tag{20}$$

Any nominal trajectory can be modified using these equations. More details on the development of the algorithm can be found in [11].

IV. Simulation Results

The evaluation of the system was conducted using the Georgia Tech UAV Simulation Toolbox (GUST). The GUST software package that combines a high-fidelity vehicle and environment model, onboard flight control software, and ground station software. GUST may be operated in hardware in the loop (HITL) mode or software in the loop (SITL) mode. In HITL mode, the flight control software and ground station interface with physical sensors, actuators, and communication links. In SITL mode, the flight control software and ground station interface with the vehicle model and simulated communication links. This design ensures that the same software is used in simulation and in flight. The vehicle model is a six rigid body degree of freedom model and simulates sensor noise, delay, location, orientation, and actuator dynamics and saturation. The vehicle model can also simulate external disturbances such as turbulence and wind.

The results presented in this section were obtained using a simulated RMAX helicopter. The RMAX has a rotor diameter of 10 ft, and weighs 150 lbs. A camera is mounted on the nose of the vehicle at a 45 deg. downward angle. The camera has a 42 deg. vertical field of view, and captures images at 640×480 resolution. The simulation is run on a desktop computer with a Core i7 processor, which allows images to be processed and feature location updates to be run at 16 Hz. It should be noted that the algorithm also has been tested on the computer installed on the RMAX helicopter, and has been found to be able to acheive the same frame rate. Therefore this algorithm is computationally capable of operating in real time on flight hardware.

To illustrate the effectiveness of the proposed method, two simulations were performed, one mapping using inverse depth parameterization, the other using Cartesian parameterization. The helicopter was made to fly a circuit at 50 ft above ground and 20 fps which would pass over a 40 ft tall structure. The obstacle avoidance algorithm was assigned to keep the helicopter at least 50 ft above the terrain.

Figure 2 shows the vehicle path after each of three laps of the circuit and the final map generated by the algorithm. It can be seen that on the first lap, the vehicle identifies the obstacle but at a lower than full height. After subsequent passes over the obstacle, the height more closely approximates the full height of the obstacle, and the final map shows an obstacle approaching 30 ft. A possible explanation for why the full 40ft height is not achieved is that the image processor relies on image texture to extract points. In this scenario, texture is found on the corners of the windows, which are at 30 ft. The simulation also illustrates the effectiveness of the terrain map in allowing the vehicle to retain knowledge about previous areas even though it has remove the points from the estimator. Figure 2(c) shows that avoidance gets better with each pass, even though the estimator must reinitialize points on the building when it see them each time.

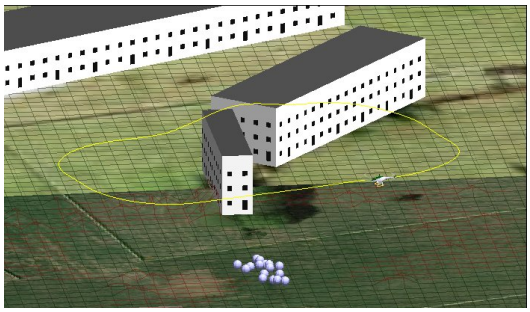
For comparison, Figure 3 shows the vehicle path and final map using standard Cartesian parameterization. This is equivalent to the system presented in [10] but without the vehicle state update. It can be seen that the vehicle does a poorer job maintaining 50 ft separation from the terrain, and that the terrain map is generally more noisy. Furthermore, the terrain map does not identify the structure, due to the slow convergence of the estimator. The estimator does not converge to a solution before the vehicle is past the obstacle.

These simulations illustrate the capabilities of monocular terrain mapping for obstacle avoidance, and highlight the benefits of the inverse depth parameterization in allowing fast convergence of the estimator.

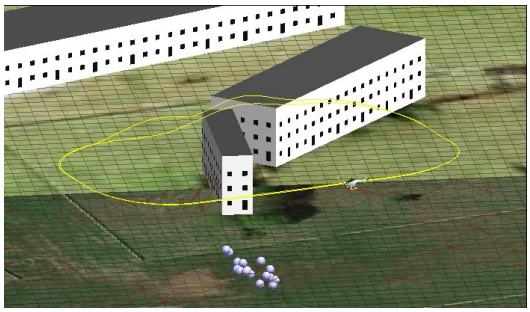
V. Results using Recorded Flight Test Data

The mapping system presented in this paper was further evaluated using recorded data from a test flight of a autonomous helicopter. The helicopter was flown in a circuit at an altitude of 100 ft which took the vehicle over a 40 ft tree line. The vehicle IMU, GPS, magnetometer and raw camera images were recorded and time stamped during the flight. These sensor inputs were then replayed in the simulation to generate the navigation solution and terrain map from the flight. This method of testing allowed rapid evaluation of the mapping system under a variety image processing settings.

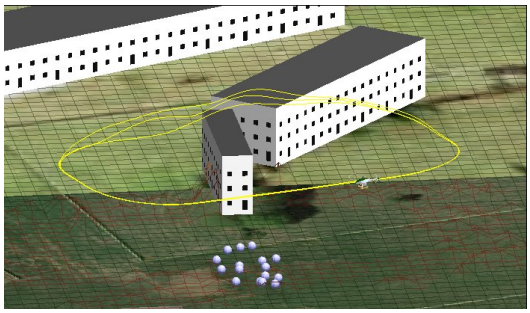
The helicopter was configured similarly to that



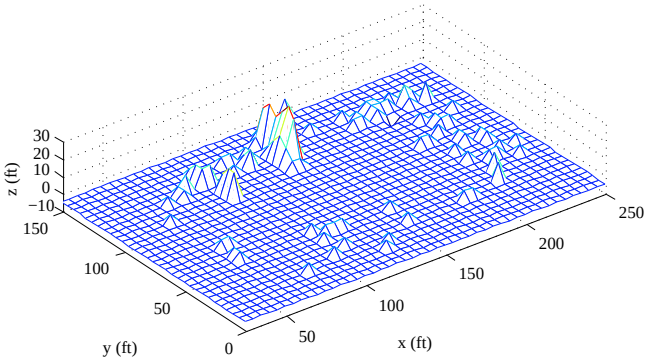
(a)



(b)

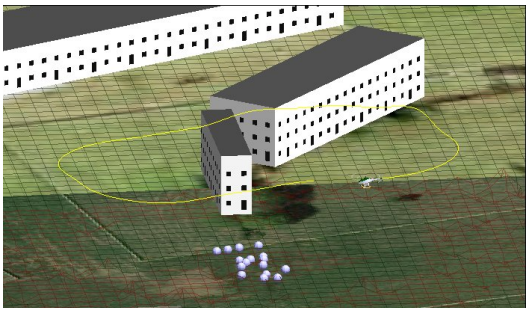


(c)

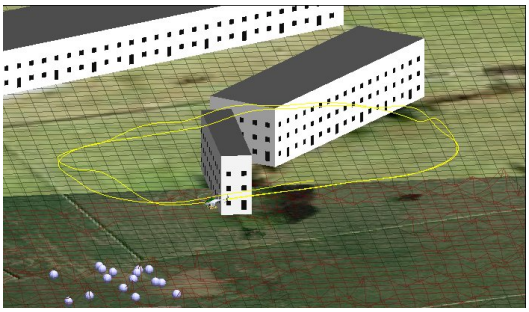


(d)

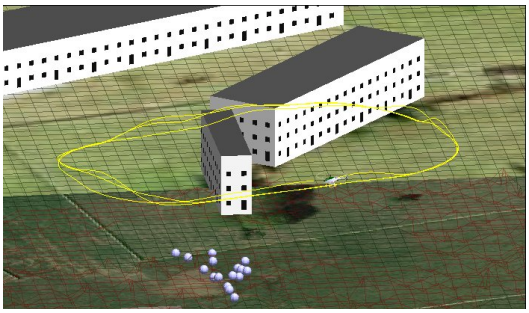
Fig. 2. Images from three circuits over a 40 ft structure, and the resulting terrain map. It can be seen that storing converged points in the terrain map improves obstacle avoidance for locations viewed multiple times. Also, terrain map accurately depicts the height of the texture on the structure, thought higher, untextured structure is not observed.



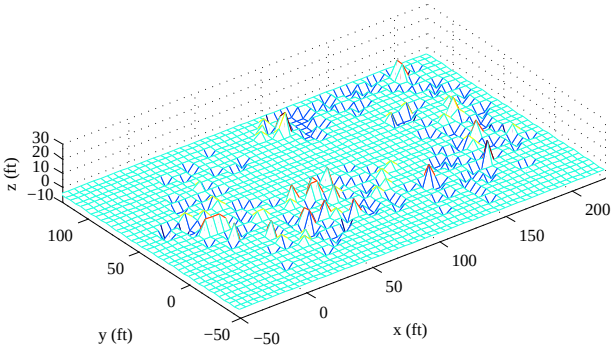
(a)



(b)



(c)



(d)

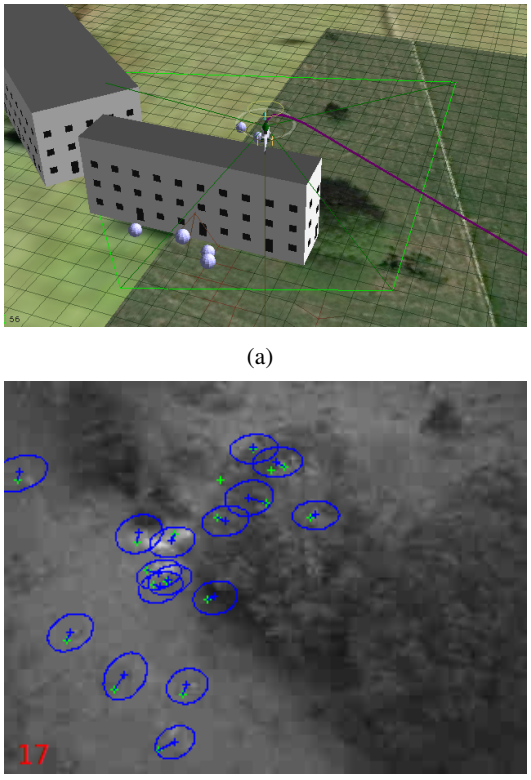
Fig. 3. For comparison, images from three circuits over a 40 ft structure, and the resulting terrain map, while using Cartesian parameterization are shown. The avoidance is much poorer, and the obstacle is not present in the map. The map is generally more noisy than when using inverse depth parameterization.

of the simulation, but images were recored a 320x240 resolution in order to maintain approximately 10 fps. Note that the low resolution and frame rate is a limitation of recording data, a step that is unnecessary during normal operation. IMU, GPS, and magnetometer data were used to generate the full navigation solution, including the state covariance necessary for feature location estimation. The flight path of the vehicle was also similar to that of simulation, with the simulated buildings in the approximate location of the tree line. Figure 4 shows a screenshot from the recorded sensor data being processed in the simulator.

Figure 5 shows the results using the recorded data. The terrain map identifies the tree line as the 40 ft spike in the middle of the map. Also visible are several fence lines, the longest along the bottom of the image. The fence contributes to several mis-mapped points, due to correspondence errors. Additional noisy map data can be seen behind the tree line, where there was an area where feature points appeared inconsistently. As can be seen in Figure 4(b), the tree line itself had very low contrast and it was very difficult to get consistent features from the image processing.

VI. Conclusion

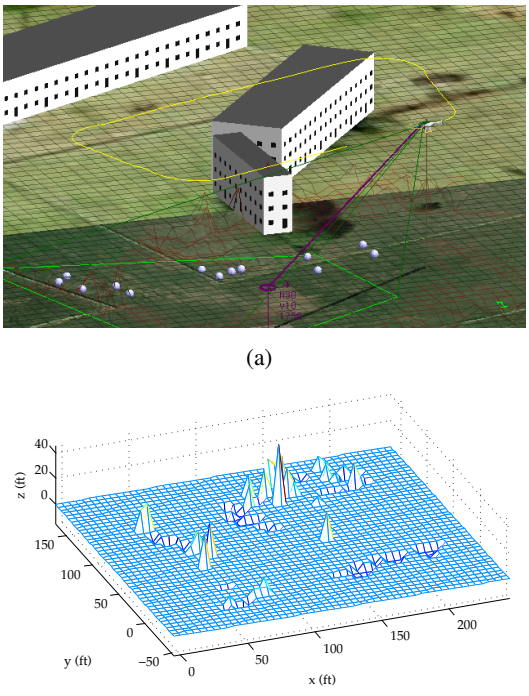
This paper presents a monocular terrain mapping system which enables obstacle avoidance for UAVs. Key advantages of this system are the use of inverse depth parameterization of the database point states, which enables fast estimator convergences. Converged points are added to a terrain map of the area of interest. Simulation results presented in the paper illustrate the effectiveness of the algorithm for obstacle avoidance, and show the benefit of using inverse depth parameterization over standard Cartesian parameterization. Implementation on data recorded during a flight test showed the mapping algorithm is able to generate a terrain map with real sensor data as input.



(a)

(b)

Fig. 4. Image from processing of the recoded flight test data in the simulator. The top figure shows the navigation solution and the camera rhombus, and the bottom figure is an image from the recorded data processed at that time step. The bottom figure also shows extracted features from the image in green and corresponding database points in blue.



(a)

(b)

Fig. 5. Full flight trajectory of the recorded data and resulting terrain map. Obstacle avoidance was not tested since the terrain map was generated from recorded data. The terrain map shows an obstacle at the tree line at 40 ft, approximately the height of the tree. The fence line is also clearly visible in the terrain map, occasionally mapping to incorrect heights.

REFERENCES

[1] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, oct. 2003, pp. 1403 –1410 vol.2.

[2] J. Civera, A. Davison, and J. Montiel, "Inverse depth parametrization for monocular slam," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 932 –945, oct. 2008.

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, nov. 2007, pp. 225 –234.

[4] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slambased navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011. [Online]. Available: http://dx.doi.org/10.1002/rob.20412

[5] H. Yu and R. Beard, "A vision-based collision avoidance technique for micro air vehicles using local-level frame mapping and path planning," *Autonomous Robots*, vol. 34, no. 1-2, pp. 93–109, 2013.

[6] S. Q. Marlow and J. W. Langelaan, "Local terrain mapping for obstacle avoidance using monocular vision," *Journal of the American Helicopter Society*, vol. 56, no. 2, pp. 22 007–1–22 007–14, 2011. [Online]. Available: http://www.ingentaconnect.com/content/ahs/jahs/2011/00000056/00000002/art00007

[7] S. Hrabar and G. Sukhatme, "Vision-based navigation through urban canyons," *Journal of Field Robotics*, vol. 26, no. 5, pp. 431–452, 2009. [Online]. Available: http://dx.doi.org/10.1002/rob.20284

[8] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: state estimation, river detection, and obstacle mapping," *Autonomous Robots*, vol. 33, pp. 189–214, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10514-012-9293-0

[9] Y. Watanabe, A. J. Calise, and E. N. Johnson, "Vision-based obstacle avoidance for uavs," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007.

[10] G. Chawdhary, E. N. Johnson, D. Magree, A. Wu, and A. Shein, "Gps-denied indoor and outdoor monocular vision aided navigation and control of unmanned aircraft," *Journal of Field Robotics*, Accepted, 2013.

[11] E. N. Johnson, J. G. Mooney, C. Ong, V. Sahasrabudhe, and J. Hartman, "Flight testing of nap-of-the-earth unmanned helicopter systems," in *67th American Helicopter Society International Annual Forum*. Virginia Beach, Virginia, May 2011.

[12] E. N. Johnson, J. G. Mooney, M. White, V. Sahasrabudhe, and J. Hartman, "Terrain height evidence sharing for collaborative autonomous rotorcraft operation," in *5th International Specialists' Meeting on Unmanned Rotorcraft and Network Centric Operations*. Scottsdale, Arizona, January 2013.

[13] E. J. Lefferts, F. L. Markley, and M. Shuster, "Kalman filtering for spacecraft attitude estimation," *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, 1982.

[14] C. Harris and M. Stephens, *A combined corner and edge detector*. Manchester, UK, 1988, vol. 15, no. Manchester, pp. 147–151. [Online]. Available: http://www.cis.rit.edu/ cnspci/references/dip/harris1988.pdf