AIAA Guidance, Navigation, and Control Conference
10 - 13 August 2009, Chicago, Illinois

AIAA 2009-5678

3D Obstacle Detection Using a Single Camera

Syed Irtiza Ali Shah[1] and Eric N Johnson[2]
Georgia Institute of Technology, Atlanta, GA 30332

This paper aims at detecting obstacles using a single camera in an unknown three dimensional world, for 3D motion of an unmanned air vehicle. Obstacle detection is a pre-requisite for collision-free motion of a UAV through 3D space. Most research towards vision based obstacle detection and avoidance has been done for 2D planar motion of ground robots and using active sensors like laser range finders, sonar, radar etc. Passive camera based research has mostly been done, either using stereo vision (multiple cameras) or, by developing a prior expectation map of the world and its comparison with the new image data. In this paper, an attempt has been made to find a 3D solution of the obstacle detection problem using a single camera in an unknown world. The equations developed and the simulations results presented here, show that a 3D model of the scene can be generated from 2D image information from a single camera flying through a very small arc of lateral flight around the object, without the need of capturing images from all sides as in a typical 'structures from motion' problem. The forward flight simulation results show that the depth extracted from forward motion is in fact usable for large part of the image, which is a significant contribution of this work.

I. Introduction

There has been an extensive literature addressing obstacle detection and avoidance, particularly for ground robots. The most common approach to obstacle detection and avoidance is that of use of multiple sensors. Thus for example, David Coombs and Karen Roberts (Ref 1) propose two cameras looking obliquely to steer between objects. The left and right proximities have been compared to steer through the gap. Another similar development is a vision system capable of guiding a robot through corridor-like environments by Argyros and Bergholm (Ref 2). It uses three cameras, one for central forward vision and the other two for peripheral vision. The main principle is to implement a honey-bee-like reactive centering behavior by controlling the movement in a way that the optical flow on both sideward-looking cameras is equal. The normal flow for all three cameras is computed by an intensity-based algorithm, after which, the depth to obstacles visible in the periphery cameras is extracted, by using the central camera to compensate for the rotational component of the ego-motion. It may be seen that the hardware requirements for this approach are that of three cameras and two workstations in order to compute the three optical flows. Analogous approaches have been proposed and successfully applied for various robotic platforms. Representative examples are Ref 3 for Stereo Vision (most common for ground robots) and Ref 4 for fusing Radar and Vision for obstacle avoidance on cars.

In his PhD Thesis (Ref 5), Randal C Nelson proposes the use of certain measures of flow field divergence as a qualitative cue for obstacle avoidance. It has been shown that directional divergence of the 2D motion field indicates the presence of obstacles in the visual field of an observer, undergoing generalized rotational and translational motion. Divergence information has been calculated from image sequences, based on the directional separation of optical flow components and the temporal accumulation of information. The use of the system to navigate between obstacles has been demonstrated by experimental results. This approach essentially does not do obstacle detection in 3D space, but instead successfully comes up with a 'No-Go' direction approach, skipping directly to the obstacle avoidance part.

---

[1] Graduate Research Assistant, School of Aerospace Engineering, 270 Ferst Drive Atlanta GA 30332-0150, Member AIAA.

[2] Lockheed Martin Associate Professor of Avionics Integration, School of Aerospace Engineering, 270 Ferst Drive Atlanta GA 30332-0150, Member AIAA.

In their paper[6], Young et. Al. present an approach to obstacle detection, using optical flow without recovering range information. A linear relationship, plotted as a line called reference flow line, has been used to detect discrete obstacles above or below the reference terrain. The parameters of the reference flow line are estimated using the optical flow of a specific part of the picture that is assumed to be obstacle-free. Slopes of surface regions have also been computed. Objects that intersect with the reference space line and occlude it, cause different flow values than the reference line and can thus be detected. It may be seen that this approach may work effectively for ground robots in general, and for UAVs during landing, but does not seem very useful in normal 3D flight of a robotic UAV, primarily because of absence of any reference or obstacle free terrain data in completely unknown flying environments. Ref 7 and 8 also present approaches for obstacle detection and avoidance in either structured, or partially known environments.

Nicholas Hatsopoulos and James Anderson[9] also use optical flow, but instead calculate time to contact, which is an optical property. However, they describe in the paper that this approach, which has been proposed for collision avoidance in cars, is not effective in realistic driving environments, when the surfaces are not very flat and are not perpendicular to the center of camera axis.

Nakao et al[10] present a method of 3D shape reconstruction of objects for a camera mounted on a robotic arm with the object being modeled on the turn table. This approach effectively uses a single camera and an Extended Kalman Filter for 3D shape reconstruction. However, this paper does not seem to address the correspondence problem in detail, (probably because there are very few feature points in the scene in such structured environment).

Besides, there had been a lot of literature under the heading of 'Structure from Motion' problem. The problem at hand may be considered as an improved solution approach to such a problem.

The problem attempted in this paper is that of a single sensor, which is a camera and the solution being sought here in this paper is that for a 3D problem in perfectly unknown world for potential test vehicle as GTMax UAV (Figure 1). The details of the proposed approach are described in the following section.

Figure 1. GeorgiaTech GTMax UAV – Potential test vehicle for approach developed here

II. Proposed 3D Obstacle Detection: Lateral Flight

A. Equations of Camera Motion

For the present problem, it is supposed that a camera is capturing 2D images and is mounted on a UAV. Immediately after the detection of feature points in the scene, UAV stops its forward flight and instead starts flying around the object, following a circular path, where the flight path is tangent to the radial vector to the object. UAV flies in a radius of flight 'r', with angular velocity 'ω' at a constant altitude 'h' . The relative position of the camera in 3D space is 'x', 'y', 'z' and its orientation is 'φ', 'θ' and 'ψ' (Refer Figure 2 below). This is an extreme case of obstacle avoidance maneuver selected to maximize predicted ability to generate the 3D map.

With the vehicle frame of reference as North-East-Down (NED), the following states and their rates are obtained for the camera

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\sin\omega t \\ r\cos\omega t \\ -h \end{bmatrix} + \Delta Position \qquad \text{and} \qquad \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} r\omega\cos\omega t \\ -r\omega\sin\omega t \\ 0 \end{bmatrix} + \Delta Velocity \qquad (1)$$

where *x,y,z* are the position states, with dot notation specifying the rate and *ΔPosition* and *ΔVelocity* are the error values for position and velocity vectors modeled as Gaussian noise vector of size 3x1, respectively. (Values of the noise covariances have been chosen keeping in view similar calculations e.g. in Ref 11).

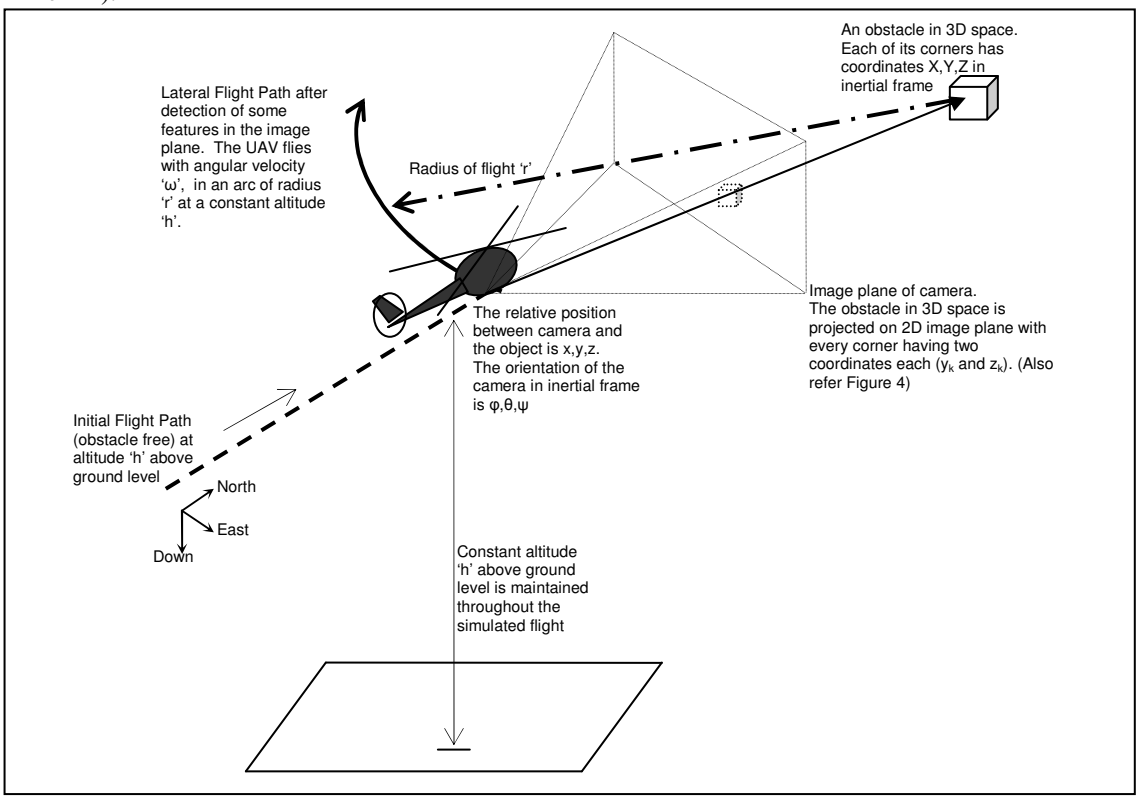


Figure 2. Camera Mounted on a UAV with a Detected Object in the Scene

The orientation and orientation rates of the camera are given by

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix}=\begin{bmatrix}\phi_c\\ \theta_0\\ -\omega t\end{bmatrix}+\Delta Orientation \quad \text{and} \quad \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\begin{bmatrix}0\\ 0\\ -\omega\end{bmatrix}+\Delta AngRate \tag{2}$$

where Φ, θ, ψ define the orientation of the camera on the UAV, Φ_c is the installation angle of camera on UAV, dot notation specifies the rate and *ΔOrientation* & *ΔAngVelocity* are the noise values for Orientation and orientation rates, modeled as Gaussian noise vectors of size 3x1, respectively.

For conversion between body frame and vehicle frame, the rotation matrix is as follows

$$L_{bv}=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\phi & \sin\phi\\ 0 & -\sin\phi & \cos\phi\end{bmatrix}\begin{bmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{bmatrix}\begin{bmatrix}\cos\psi & \sin\psi & 0\\ -\sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3}$$

and

$$L_{vb}=L_{bv}{}^{T} \tag{4}$$

B. Z-test for Correspondence

Statistical Z-test method has been used to solve the correspondence problem between the estimated corners from database and the measurements. The Z-test has been taken for a certain error index (*J*) and is defined as the square of this index divided by its variance (*C*) i.e Ztest value=J^2/C. Both the estimation error covariance (matrix *P*) and the measurement error covariance (matrix *R*) have been taken into account while calculating *C* (equations 5 & 6). Then the Z-test value is inversely related to the likelihood of an event that a given measurement corresponds to the corner point chosen. Thus for example, if there is a

large error between the measurement and the image data, but the measurement also has a large uncertainty, then the probability of its correspondence should be higher than the case in which, the measurement has a small uncertainty. Thus each corner point is to be assigned to a point, which attains the least Z-test value, meaning thereby, the highest likelihood.

For $\boldsymbol{Z}$ being the projected measurement vector onto image plane and $\boldsymbol{X}$ being the relative position vector in 3D space, the error index J and its variance C for calculating Z-test value are defined as

$$J = dx^2 + dy^2 \qquad C = C_X P C_X{}^T + C_Z R C_Z{}^T \tag{5}$$

where

$$C_X = \frac{\partial J}{\partial X_v} \quad \text{and} \quad C_Z = \frac{\partial J}{\partial Z} \tag{6}$$

are the two components of the variance C of the error index J.

In the figure 3, $\mathbf{Z}$ is the projected measurement vector onto image plane and $\boldsymbol{x_k}$ is the projected database corner vector onto the image plane. Hence, it may be noted that the residual vector is

$$\boldsymbol{d} = \boldsymbol{Z} - \boldsymbol{x_k} \tag{7}$$

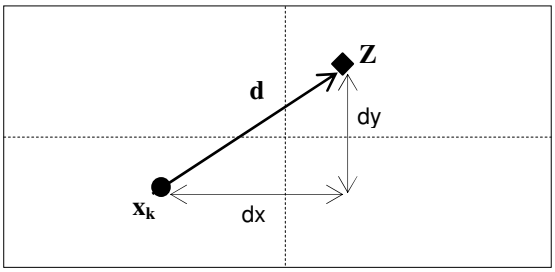


Figure 3. The Residual Vector

C. Pin-Hole Camera Model

Assuming that the camera is mounted at the center of gravity of the vehicle, let L_{bv} denote a known camera attitude represented by a rotation matrix from inertial to the camera frame. A camera frame is taken so that the camera's optical axis aligns with its $X_c(t)$axis. Then the relative position in camera frame will be as follows

$$\boldsymbol{X} = \boldsymbol{X_v} - \boldsymbol{X_p} \quad \text{(in inertial frame)} \tag{8}$$

$$\boldsymbol{X_c} = L_{bv}\,\boldsymbol{X} \quad \text{(in camera frame)} \tag{9}$$

where

$$\boldsymbol{X_c} = [X_c(t) \quad Y_c(t) \quad Z_c(t)]^{\mathrm{T}} \tag{10}$$

and the subscript 'v' is used for vehicle position vector, the subscript 'p' is used for the the object position vector, subscript 'c' is used for the camera and upper case (bold-italic) $\boldsymbol{X}$ indicates a 3x1 relative position vector in 3D space.

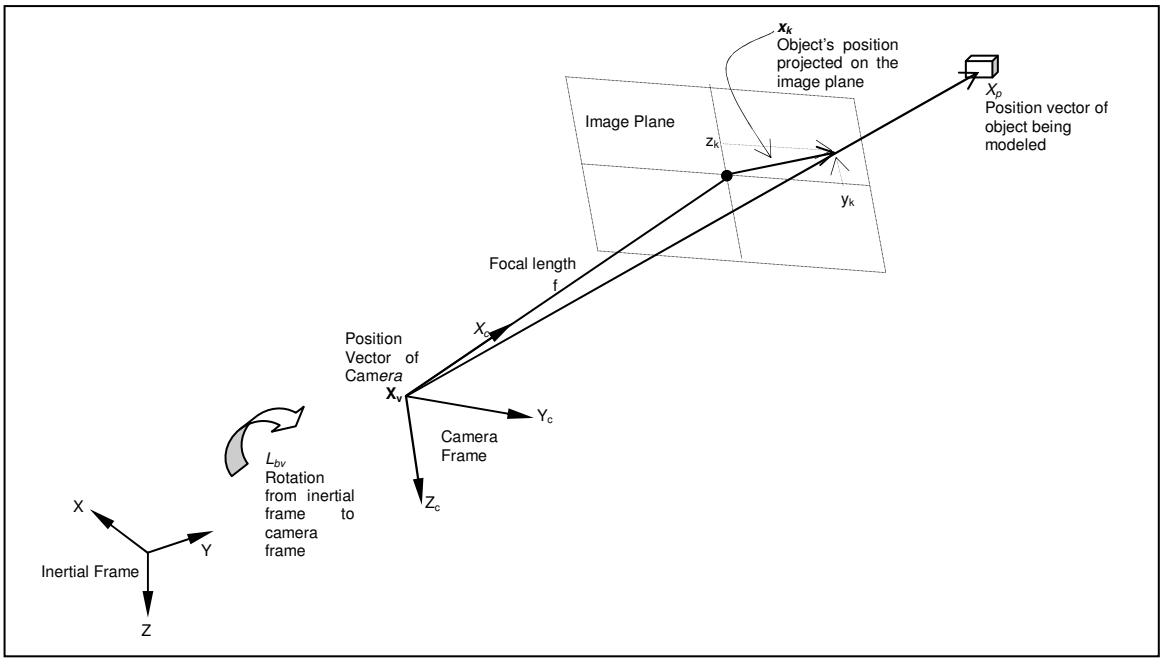


Figure 4. A Standard Pin-Hole Camera Model[11]

Assuming a pin-hole camera model as shown in the Fig 4, the object position in the image at a time step t_k is given by (lower case $\boldsymbol{x_k}$ is a 2x1 vector in the image plane)

$$x_k = \begin{bmatrix} y_k \\ z_k \end{bmatrix} = \frac{f}{X_{ck}} \cdot \begin{bmatrix} Y_{ck} \\ Z_{ck} \end{bmatrix} \tag{11}$$

This equation is non-linear with respect to the relative state. (Hence an Extended Kalman Filter has been used here).

In the implementation, focal length '*f*' of the camera has been assumed to be unity without loss of generality.

Following Eqns 5, 6 and 7, we can now write expressions for the components of Variance matrix C as C_x and C_z, using chain rule as follows:

$$C_X = \frac{\partial J}{\partial X_v} = \frac{\partial J}{\partial d} \cdot \frac{\partial d}{\partial x_k} \cdot \frac{\partial x_k}{\partial X_v} \qquad \text{and} \qquad C_Z = \frac{\partial J}{\partial Z} = \frac{\partial J}{\partial d} \cdot \frac{\partial d}{\partial Z} \tag{12}$$

D. Extended Kalman Filter

Predict:

The predicted stage (before the new measurement) of the Extended Kalman filter is defined by

$$X_k^- = f(X^-_{k-1}, U_k)$$

which, for this case of no dynamnics and no input for the feature point being modeled, simplifies to

$$X_k^- = X^-_{k-1} \tag{13}$$

The estimation covariance matrix is defined by

$$P_k^- = F_k P^-_{k-1} F_k^T + Q_k$$

which for no dynamics case, simplifies to

$$P_k^- = P^-_{k-1} + Q_k \tag{14}$$

Update:

The Kalman gain, the state and the estimation covariance for update stage (after the new measurement) are given by

$$K_k = P^-_k H_k^T (H_k P^-_k H_k^T + R_k)^T \tag{15}$$

$$X_k = X^-_k + \sum K_k d \tag{16}$$

$$P_k = P_k^- - \sum K_k H_k P_k^- \tag{17}$$

where R_k is the measurement error covariance matrix. The observation matrix Jacobian H_k , which is defined as partial derivative of the residual vector (d) with respect to partial derivative of the state ($\boldsymbol{X}$) is calculated here as

$$H_k = \frac{\partial d}{\partial X} = \frac{\partial d}{\partial x_k} \cdot \frac{\partial x_k}{\partial X} \tag{18}$$

For the vectors $\boldsymbol{d}$, $\boldsymbol{X}$ and $\boldsymbol{x_k}$ defined above, this is evaluated as (where I_2 is a 2x2 identity matrix)

$$\frac{\partial d}{\partial x_k} = -I_2 \qquad \text{and} \qquad \frac{\partial x_k}{\partial X} = \frac{1}{X_{ck}} \begin{bmatrix} -x_k & I_2 \end{bmatrix} \tag{19}$$

so that H_k turns out to be:

$$H_k = -\frac{1}{X_{ck}} \cdot I_2 \cdot \begin{bmatrix} -x_k & I_2 \end{bmatrix} \tag{20}$$

Similarly the covariance matrix R_k (measurement error covariance) is defined as

$$R_k = \frac{\partial d}{\partial Z} \cdot R \cdot \left(\frac{\partial d}{\partial Z} \right)^T \tag{21}$$

which is evaluated as:

$$R_k = I_2 \cdot R \cdot I_2 \tag{22}$$

where R is the measurement noise covariance.

It may be seen now that the components of Variance matrix *C,* as C_x and C_z given above, may be evaluated as:

$$C_X = \frac{\partial J}{\partial X_v} = \frac{\partial J}{\partial d} \cdot H_k = 2\, d^T \cdot H_k \tag{23}$$

$$C_Z = \frac{\partial J}{\partial Z} = \frac{\partial J}{\partial d} \cdot \frac{\partial d}{\partial Z} = 2\, d^T \cdot I_2 \tag{24}$$

E. 3D Modeling Algorithm

The above equations have been implemented in the 3D scene modeling algorithm as shown in Fig 5. The algorithm starts when a feature point is detected in the scene. This information is being fed to the program by the frame grabber after image processing and segmentation. Further, the camera calibration information is also being fed to the program by GTMax onboard systems, which includes its location in 3D space and installation angle on the UAV, besides the knowledge of its FOV (Field of View) and its image plane size. The UAV then starts flying in a circular path of radius 'r'. Equations 1 – 4 give the position, orientation and respective rates for the camera and equations 8, 9 and 10 give the position information in camera and inertial frame. For the first iteration, as the estimation database is empty, all the feature points as measured in the image frame, go into the estimation database without establishing any correspondence. Since this was only 2D information from the image plane, the third dimension is unknown and is supposed to be zero i.e. all points are supposed to lie on the ground plane initially. When the subsequent image information is received, the estimation points in the database are projected onto image plane (via equation 11) and the residual vector is calculated between the new measurement and the estimated points on image plane (equation 7). Z-test correspondence is done to establish which measurement corresponds to which estimated value (equations 5, 6, 7 and 12) and the new values are updated with the extended Kalman filter (equations 13 to 24). If correspondence is not established between a measured feature points in the image, with any of the estimated feature points, this feature point is recognized as a new point. Conversely if an estimated feature point existed, for which there was no corresponding measurement in the new image, this is marked for deletion. However, it is actually not deleted unless it remains without correspondence for next consecutive N images. This is done to ensure, that if a feature point temporarily goes out of view, it is not deleted immediately, otherwise the whole simulation time would increase, if it came back into the view later on and was instead recognized as a new feature requiring new estimation starting from ground plane.

F. Simulation Results

As a first case, a cube was selected with eight corners (or eight feature points). This known model of the cube was used to verify the ability of the algorithm to successfully generate its 3D model using the 2D image information captured from the camera. The simulation results are as presented in Fig 6. In this figure, the solid (magenta) lines indicate the object to be modeled, the blue diamonds indicate the progressive outcome of corner estimation from the proposed algorithm, whereas the wavy black arcs indicate the flight path of the camera. The final figure (at 60 sec) shows that the blue diamonds approach the actual corners of the object being modeled, indicating a successful 3D obstacle detection for this case.

As a next case, a scene comprising of 35 feature points was chosen, as various corners of high-risers in a typical urban scenario. The simulation results for lateral flight path are shown in Fig 7. In this figure also, the solid (magenta) lines indicate the object to be modeled, the blue diamonds indicate the progressive outcome of corner estimation from the proposed algorithm, whereas the wavy black arcs indicate the flight path of the camera. The final figure (at 100 sec) shows that the blue diamonds approach the actual corners of the object being modeled, indicating a successful 3D obstacle detection for this case as well.

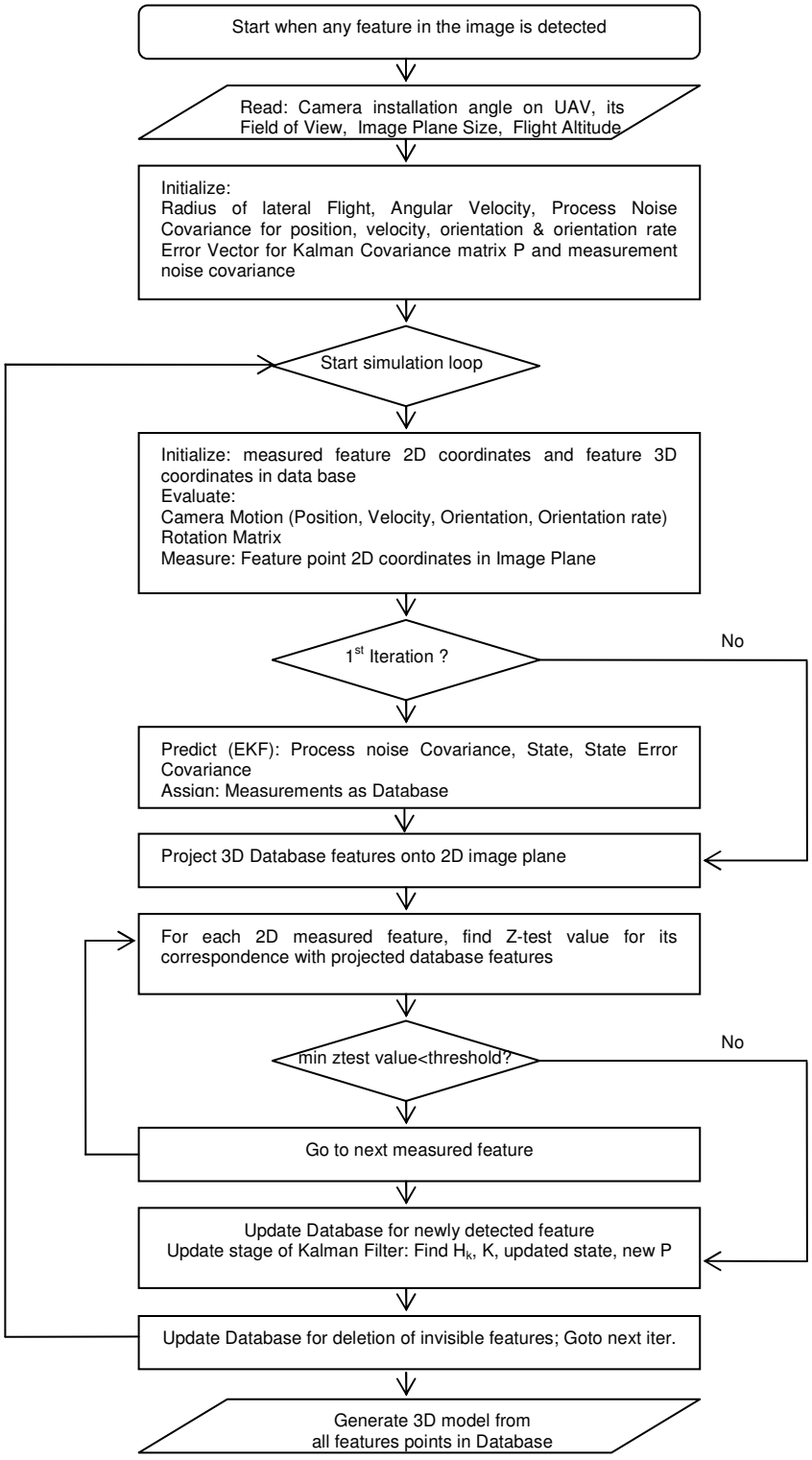
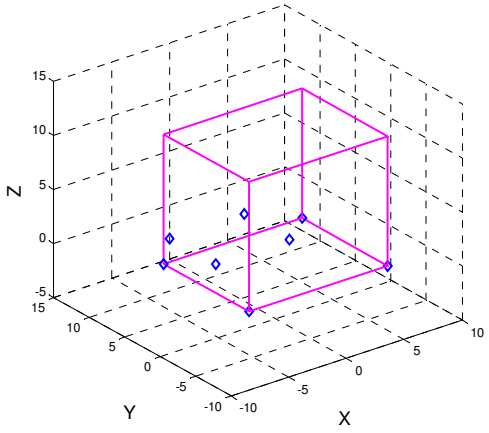
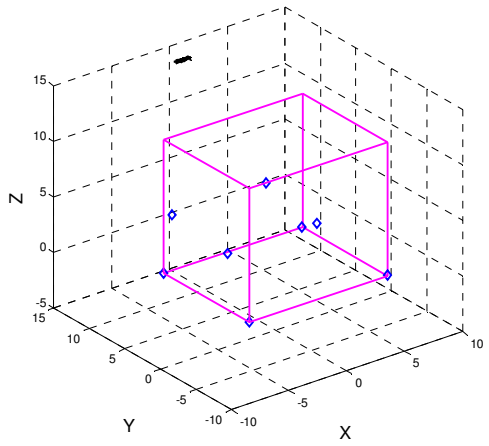


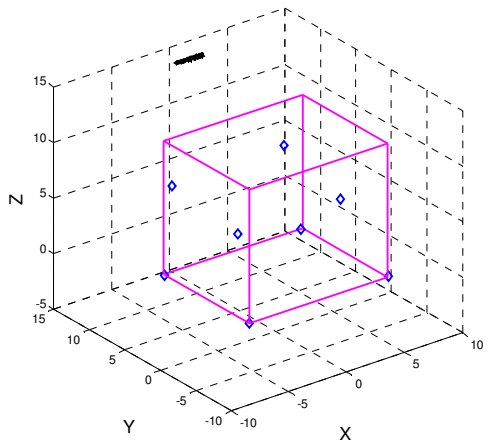
Figure 5. The Proposed Algorithm



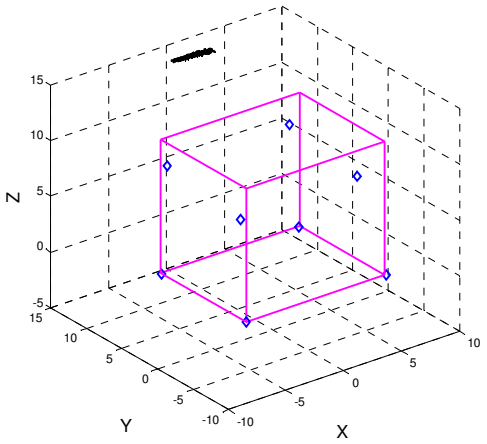
a) At time 0 sec



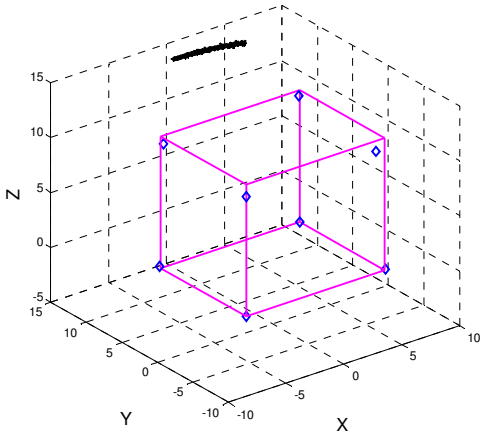
b) At time 12 sec



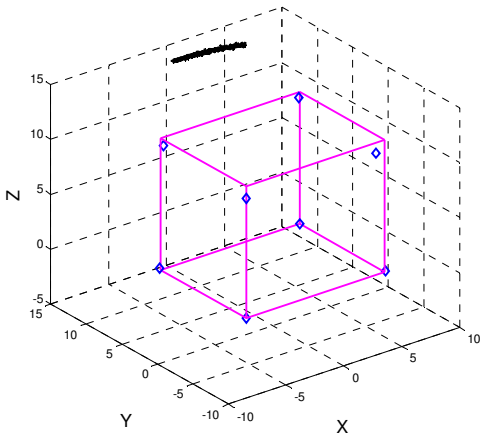
c) At time 24 sec



d) At time 36 sec

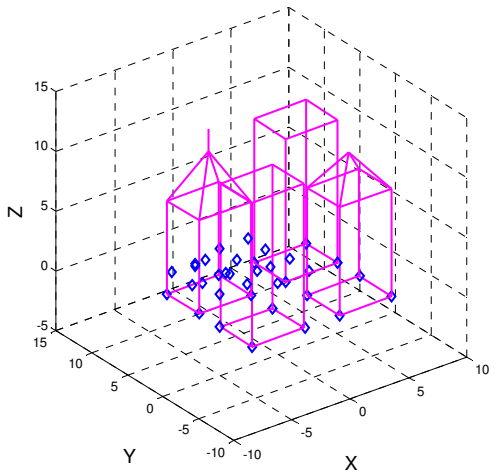


e) At time 48 sec

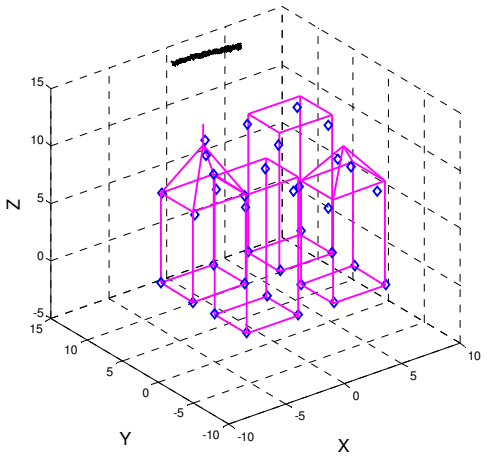


f) At time 60 sec

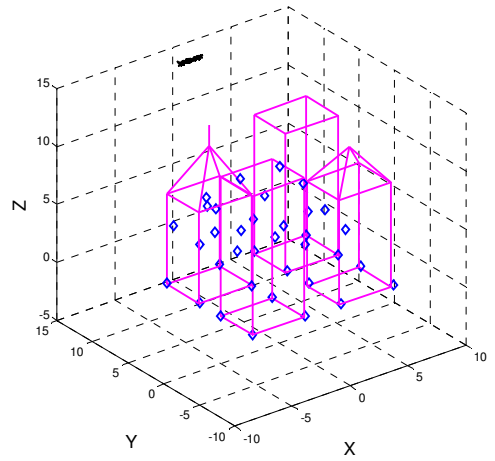
Figure 6. Flight Simulation results with 8 feature points. Image processing is updated at 10 frames / sec. Convergence is good at 60 sec, traveling 25 deg around the object.



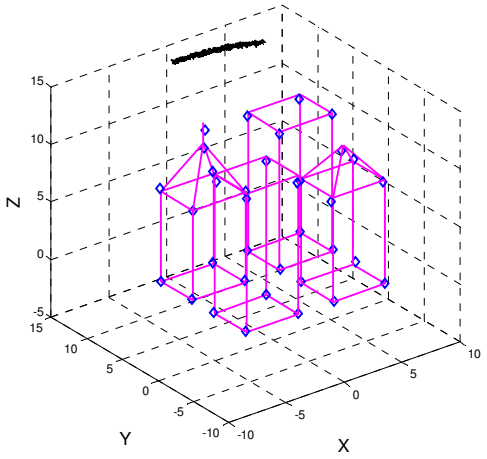
a) At time: 0sec



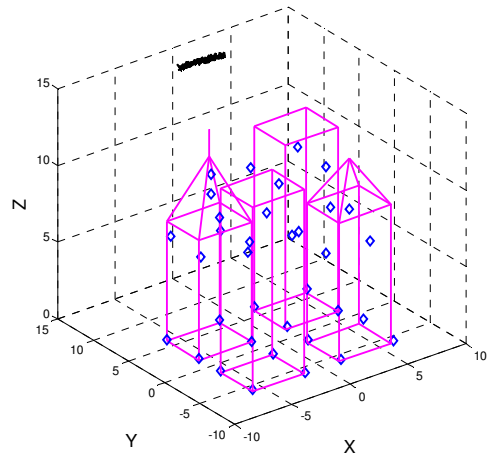
d) At time: 60sec



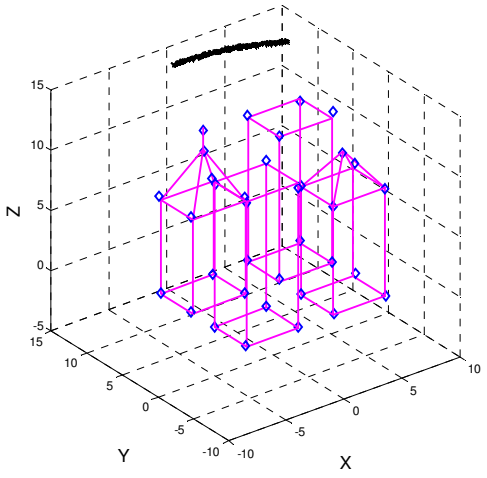
b) At time: 20 sec



e) At Time: 80 sec



c) At time: 40sec



f) At time: 100sec

Figure 7. Flight Simulation results with 35 feature points. Image processing is updated at 10 frames / sec. Convergence is good at 100 sec, traveling 40 deg around the object.

Table 1 below gives the values used for the simulation.

Table 1. Values used for simulation: Lateral Flight

Flight altitude above ground level	140 ft
Radius of flight about the object	140 ft
Angular velocity around the object	0.36 deg/sec
Camera field of view	30 deg
Position error in all three states each	1%
Velocity error in all three states each	1%
Orientation error in all three states each	0.01%
Angular velocity error in all three states each	0.01%

The simulation results of Figure 6 and 7 show that the proposed algorithm can successfully generate a 3D model of the scene, from 2D image information. This modeling only requires one camera as the sensor. These results have been achieved for an unknown world and no constraints were put on the environment being modeled. No attributes of the environments were provided to the system, except for the 2D images being captured by the camera. The scene modeling has been achieved in 60 seconds of flight for 8 feature points, and 100 seconds of flight for 35 feature points. The successful 3D scene modeling required flying through a very small arc of lateral flight, as compared to the size of object being modeled. There had been no need to capture images from all sides of the objects being modeled. Thus the approach is more practical than a typical 'Structure from Motion' problem, which requires right, left, top or other views of the object, in order to generate its 3D model. The algorithm does require some feature points in the scene. Hence if no feature points are detected in the scene, the algorithm implies that there are no obstacles to be avoided and the initial flight path of the UAV may be continued without any disruption. This is almost always true in real world scenarios. However, there could be one exception of that of a flat wall in front, which is discussed as 'Future Work' in Section V.

III. 3D Obstacle Detection : Forward Flight

A. Introduction

In their research paper[12], Matthies, Kanade and Szeliski present Kalman Filter-based Algorithms for Estimated Depth from Image Sequences. Besides other conclusions, they have shown that

1) For a translating camera, the accuracy of depth estimates increases with increasing distance of image features from the focus of expansion (FOE, which is a point in the image where translation vector pierces the image).
2) Best translations are parallel to the image plane and the worst are forward along the camera axis.
3) For practical fields of view, the accuracy of depth extracted from forward motion will be effectively unusable for a large part of the image. Thus for practical depth estimation, forward motion is effectively unusable compared with lateral motion.

B. Proposed Approach

Section II of this paper demonstrated that Lateral flight gives good 3D scene modeling of objects from 2D image data. (In fact depth is just one coordinate of any feature point in 3D space). This substantiates deduction 2 above. This however, is apparently an awkward flight maneuver form a practical perspective in the sense that a UAV, which was supposed to fly forward, has to start flying laterally, as soon as some object is detected in the scene, in order to estimate its depth or 3D location in space. Hence, here an attempt was made to do depth estimation while flying forward, which is in conflict to what was recommended in conclusion 3 above. However, two facts are important here.

Firstly, estimation of 3D positions of those objects is attempted, which do not lie exactly at focus of expansion, because if the features exactly lie at FOE, there is no solution to the problem. This is in accordance with the first deduction mentioned above. We propose that if the features are not at FOE, even flying forward could give

reasonable depth estimation. Of course the accuracy would improve with increasing distance of features from FOE, as stated in Ref 12 above.

Secondly, it may be noted that the conclusions in the above-referred paper were arrived at by linearizing the system equations and using a Kalman Filter. In this paper however, it is investigated, whether the use of non-linear Extended Kalman Filter instead of a regular Kalman Filter, can provide good results for forward flight of a UAV.

Accordingly it is proposed in this work that, subject to the two considerations just mentioned above, flying forward will give depth estimation, which is of practical use, as opposed to deduction 3 of above referred paper.

Implementation of this 3D obstacle detection in forward flight, changes only the equations of motion of camera i.e. equations 1 for position and velocity of Section II above. All other equations presented for Lateral flight in Section II above, remain valid in this forward flight case as well. This also applies to fig 3 (Residual Vector), fig 4 (Pin-hole Camera Model) and fig 5 (Proposed Algorithm). The changes required in equations 1 for position and velocity are: $\omega=0$ (for no lateral flight), 2nd (forward) component of position vector added with a factor '$a \times t$', where 'a' is forward velocity and t is time. Also second component of velocity vector is added with this constant factor 'a' (forward velocity). Hence the new equations are

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ r + a \cdot t \\ -h \end{bmatrix} + \Delta Position \qquad \text{and} \qquad \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ a \\ 0 \end{bmatrix} + \Delta Velocity \qquad (1)'$$

To avoid the obstacles in FOE, the speed of flight is a critical factor. If it is too high, the images of the objects, which are enlarging, in this case as the motion is towards them, will quickly occupy almost whole of the image, including FOE as well. Hence the 3D scene modeling would not be possible. On the contrary if the speed of flight is too low, there is less variation in the subsequent images, and hence less new information in those images. This will in turn prolong the simulation time to an unacceptable extent. In this case of 35 feature points, the optimum speed of flight was found by iterations in repeated simulations, so as to achieve the correct 3D modeling at a relatively high speed.

C. Forward Flight Simulation Results

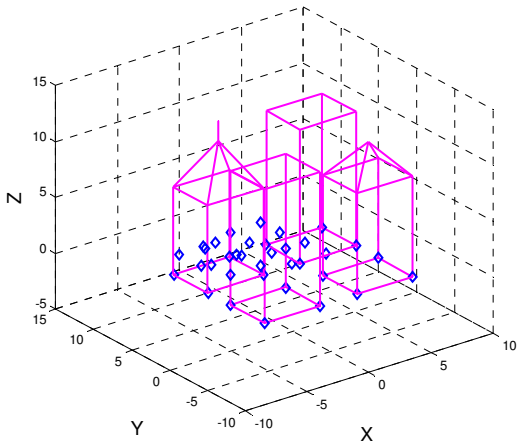
The simulation results for 3D obstacle detection in forward flight are presented in Figure 8. In this figure, the solid (magenta) lines indicate the object to be modeled, the blue diamonds indicate the progressive outcome of corner estimation from the proposed algorithm, whereas the wavy black line indicates the forward flight path of the camera. The final figure (at 125 sec) shows that the blue diamonds approach the actual corners of the object being modeled, indicating a successful 3D obstacle detection for this case.

The table below gives the values used for the simulation

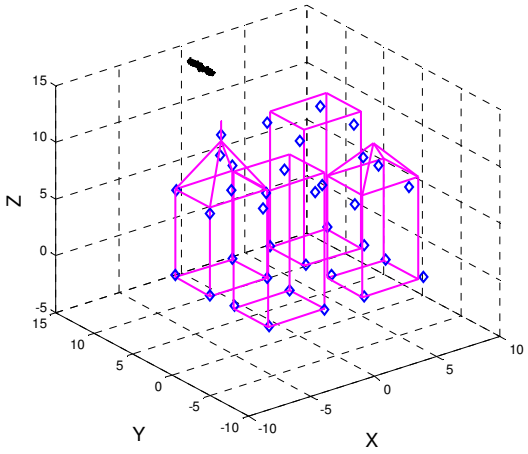
Table 2. Values used for simulation : Forward Flight

Flight altitude above ground level	140 ft
Forward Velocity	1.4 ft/sec
Camera field of view	30 deg
Position error in all three states each	1%
Velocity error in all three states each	1%
Orientation error in all three states each	0.01%
Angular velocity error in all three states each	0.01%

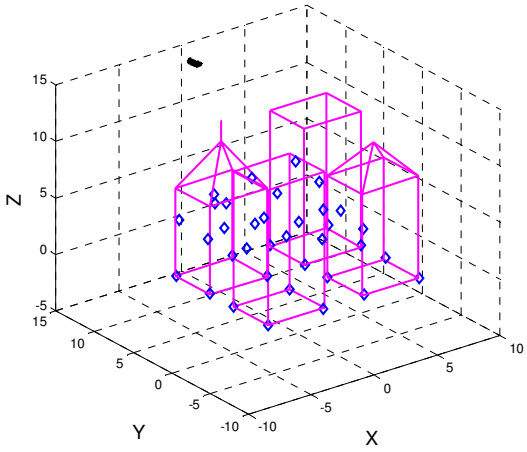
The results in figure 8 show that the proposed algorithm can successfully generate a 3D model of the scene, from 2D image information while flying forward towards the obstacle. The speed of flight is critical, as with too high a speed, obstacles will overlap the FOE. Too low a speed, on the contrary, will give very less new information for the update. Successful 3D modeling will not be possible in both such cases. Comparing the results of lateral flight simulations (Section II) and the results presented here, it can be said that, flight duration required to generate a 3D model of the scene while flying forward, was 25% more than the duration of flight required for lateral flight case. Subject to the two conditions of features not exactly at FOE and using EKF for non-linearities, the simulation results show that for practical fields of view, the depth extracted from forward motion is indeed usable for a large part of the image.



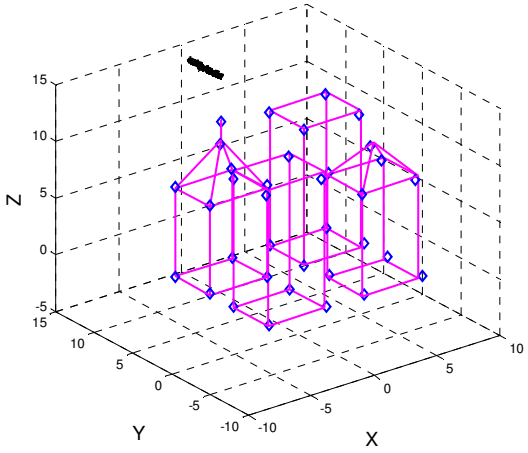
a) At time: 0 sec



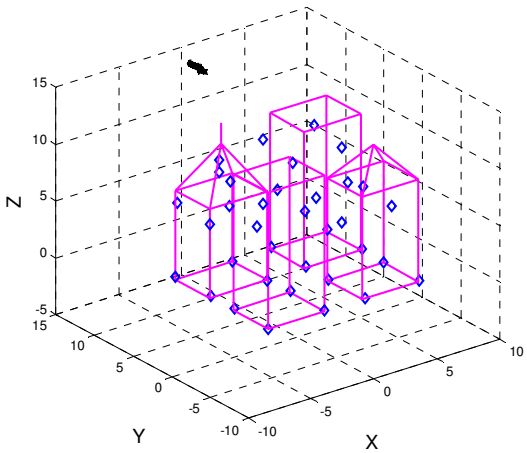
d) At time: 75 sec



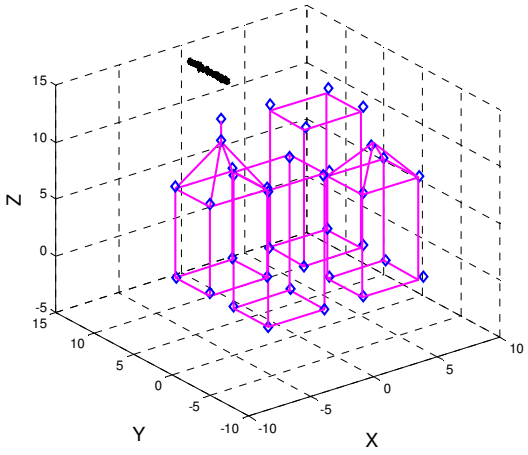
b) At time: 25 sec



e) At time: 100 sec



c) At time: 50 sec



f) At time: 125 sec

Figure 8. Forward Flight Simulation results with 35 feature points. Image processing is updated at 10 frames/sec and UAV is flying forward at 1.4ft/sec. Convergence is good at 125 sec.

IV. Analysis of Results

It may be realized that for detecting N_X number of feature points (in 3D space), we have N_X*N_Z correspondences to be established, where the N_Z is the number of feature points picked up (observed) in every image. This holds for every iteration except the first one, when the database is empty and there is no correspondence to be done. So if the frame rate is 'f' frames per second and the simulation gives satisfactory results after 't' seconds, then the total number of images we use in the simulation are

$$\text{Number of images utilized: } N_I = f*t$$

Hence the total number of correspondences, the algorithm has to establish is given by

$$\text{No of Correspondences} = N_X*N_Z*N_I - 1$$

It may be seen that as the number of feature points in the scene increases (i.e. as N_Z increases), the number of points in the database N_X also increases accordingly (as the scene feature points eventually end up as points in the database, once the correspondence is established and 3D coordinates are found). Hence the computational effort increases tremendously, which in turn results in a need for much more simulation time, as well as, many more number of images required, in order to satisfactorily do the 3D obstacle detection/modeling. For the simulations above, the computational effort increased by about 14.7 times with an increase in number of feature points by 4.4 times (precisely from 4.054 seconds of computer time required to simulate 8 feature points vis-à-vis 60.201 seconds required to detect 35 feature points to within 2.5% of accuracy) for lateral flight. This further increased by yet another 25% for forward flight.

For the above analysis, we supposed a frame rate of 10 frames per second. Hence if the simulation ran for 60 seconds, 600 images were used (case of Fig 6 above). If a better frame grabber / image processor is available so that, about 30 frames per second rate is available, the simulation time will reduce to 20 seconds, which in turn means, lesser duration of lateral flight required, to correctly model the scene.

Figure 9 presents with error analysis, for the simulation results presented in Figures 6, 7 and 8. One sample point randomly has been chosen for each of the three cases. All the three estimated coordinates of the selected feature points in the database has been compared with the actual value of coordinates. The results in Fig 9 show that all cases converge to the actual point locations, to within about +3%. This indicates successful convergence of features points to their actual locations in 3D space.

In the algorithm however, simulation is stopped when the average error from each of the three coordinates from all the feature points in the scene are successfully modeled (to within about +3% of the actual locations in 3D space). This means that for a case of 35 feature points, there are 35x3=105 coordinates to be estimated correctly.

It may be said that the obvious merit of this approach in both lateral and forward flight, is that of providing a capability of 3D obstacle detection and modeling by using only one camera. This is of significant importance for future miniature UAVs, which might not be capable of carrying any other sensor, except for a single camera. The information that is obtained as a result of this algorithm, is that of a full scale 3D model of the scene, which may be directly utilized for any mission planning, as desired. On the contrary, the algorithm has an obvious constraint of tremendous increase in computational effort with an increase in number of feature points.

Further, the lateral flight pattern for such obstacle detection may also seem as a constraint, at least to a mission, which was that of moving forward towards the goal/target. The forward flight does overcome this constraint but increases computational effort by another 25%, and comes with a constraint of having no feature points at FOE. Yet another constraint is that of at least having some feature points at all, in the scene. If the UAV takes-off e.g in front of a flat wall, there are hardly any feature points to be detected and modeled. Such a problem, in fact, is a recommended subject of future work, as discussed in Section V.

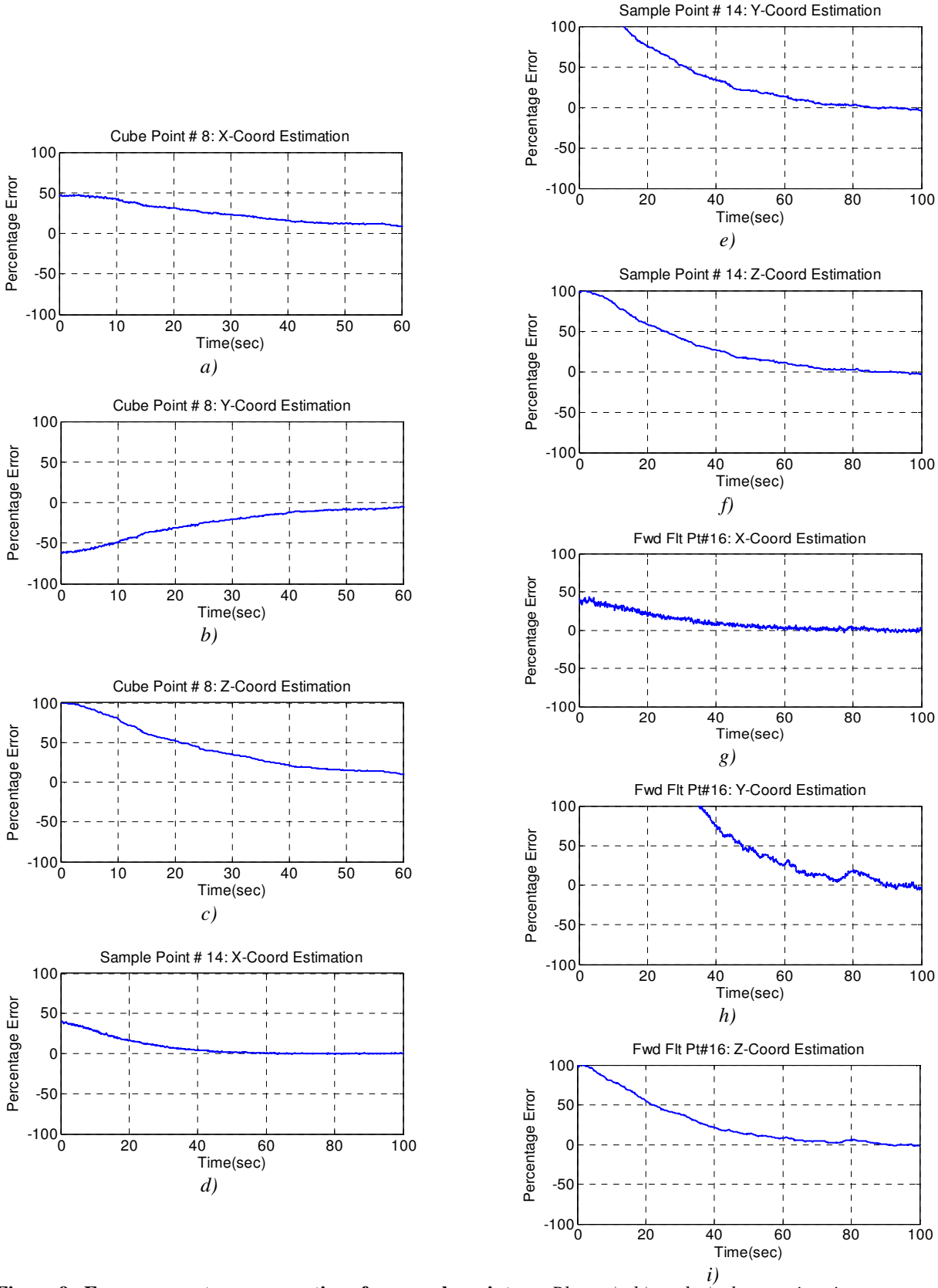


Figure 9. Error percentage versus time for sample points. *Plots a), b) and c) show estimation error progression in X, Y & Z coordinates, respectively, for a sample point from figure 6. Plots d), e) & f) are similar plots for a sample point from figure 7, whereas plots g), h) & i) are for figure 8 upto 100seconds.*

V. Conclusions & Future Work

From the proposed algorithm and associated simulations it is concluded that, the proposed algorithm can successfully generate a 3D model of the scene, from 2D image information. This modeling only requires one camera as the sensor. The results were achieved for an unknown world and no constraints were put on the environment being modeled. No attributes of the environments were provided to the system, except for the 2D images being captured by the camera. The 3D scene model gives information of size and location of all obstacles in the scene. This information is sufficient to initiate an obstacle avoidance maneuver in 3D space. In the case of lateral flight the scene modeling has been achieved (to within +3% of actual 3D locations of the feature points) in 60 sec of flight for 8 feature points, and 100 sec of flight for 35 feature points. Successful 3D scene modeling required flying through a very small arc of lateral flight, as compared to the size of object being modeled. There had been no need to capture images from all sides of the objects being modeled. Thus the approach is much better as compared to a typical 'Structure from Motion' problem, which requires right, left, top or other views of the object. In the case of forward flight, the speed of flight is critical, as with too high a speed, obstacles will overlap the FOE. Too low a speed, on the contrary, will give very less new information for the update. Successful 3D modeling will not be possible in both such cases, while flying forward. Comparing the results of lateral flight simulations with that of forward flight, it can be said that, flight duration required to generate a 3D model of the scene while flying forward, was 25% more than the duration of flight required for lateral flight case. Subject to the two conditions of features not exactly at FOE and using EKF for non-linearities, the simulation results for forward flight show that for practical fields of view, the depth extracted from forward motion is indeed usable for a large part of the image. The algorithm does require some feature points in the scene, both in case of lateral flight as well as forward flight. Hence if no feature points are detected in the scene, the algorithm implies that there are no obstacles to be avoided and the initial flight path of the UAV may be continued without any disruption. This is almost always true in real world scenarios. However, there could be one exception of that of a flat wall in front, which is discussed below.

In future, it is planned to integrate some of the single sensor based approaches similar to Flow Field Divergence concept and/or Optical flow concept (as referred in Section I above) with the algorithm proposed in this paper. To handle the case of having no feature points in the scene (e.g a flat wall in front), a non-scanning (and hence a low weight) laser range finder may also be integrated with this set-up. The final approach is planned to be implemented on GTMax UAV.

References

[1]Coombs, D., Roberts, K., "Bee-Bot: Using Peripheral Optical Flow to Avoid Obstacles" *Intelligent Robots and Computer Vision XI*, , SPIE Vol 1825 1992.

[2]Argyros, A., A., and Bergholm, F., "Combining Central and Peripheral Vision for Reactive Robot Navigation" *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages II:646-651, IEEE Computer Society Press, 1999.

[3]Takeno, J., Shinogi, Y., Nishiyama, S., Mizuguchi, N., Sorimat, K., "Realization of a 3D Vision Mobile Robot that can Avoid Collision with Moving Obstacles," *IEEE International Conference on Robotics and Automation,* Sacramento, California April 1991

[4]Langer, D., and Jochem, T., "Fusing Radar and Vision for Detecting, Classifying and Avoiding Roadway Obstacles", *IEEE Intelligent Vehicles Symposium,* 1996 pp333-338.

[5]Nelson, R. C., "Visual Navigation", PhD Thesis, University of Maryland, August 1988

[6]Young, G. S., Hong, T. H., Herman, M., Yang, J. C. S., "Obstacle detection for a vehicle using optical flow", *Proceedings of the Intelligent Vehicles '92 Symposium (Cat. No.92TH0468-9)*, 1990, p 185-90.

[7]Mikawa, M., Yoshida, K., Tanno, M., and Matsumoto, M., "Vision-based redundancy control of robot manipulators for obstacle avoidance", *IECON Proceedings (Industrial Electronics Conference)*, v 3, 1997, p 1373-1378.

[8]Lenser, S., Veloso, M., "Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision", *IEEE International Conference on Intelligent Robots and Systems*, v 1, 2003, p 886-891.

[9]Hatsopoulos, N., Anderson, J.A., "Collision-avoidance system based on optical flow", *Proceedings of the Intelligent Vehicles '92 Symposium (Cat. No.92TH0468-9)*, 1990, p 79-84

[10]Nakao, K., Kondo, K., Kobashi, S., Hata, Y., Yagi, T., "Shape Reconstruction Using Extended Kalman Filter with an Active Camera", *IEEE International Symposium on Communications and Information Technologies 2004 (IEEE Cat. No.04EX897)*, Oct 2004, Japan, p 857-60 vol.2.

[11]Watanabe, Y., "Stochastically Optimized Monocular Vision-Based Navigation and Guidance", *PhD Thesis*, Georgia Institute of Technology, April 2008.

[12]Matthies, L., Kanade, T., Szeliski, R., "Kalman Filter-Basd Algorithms for Estimating Depth from Image Sequences", *International Journal of Computer Vision,* v3, n3, 1989, p209-238