Technical Report

IISc/SID/AE/RP/AOARD/2010/01

Nonlinear Geometric and Differential Geometric Guidance of UAVs with Vision Sensing for Reactive Obstacle Avoidance

Ankush Gupta
Project Associate

Radhakant Padhi
Associate Professor

Department of Aerospace Engineering
Indian Institute of Science
Bangalore, India.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
29 SEP 2010	**FInal**	**23-06-2009 to 22-07-2010**

4. TITLE AND SUBTITLE
Reconfigurable Co-operative Flying of UAVs at Low Altitude in Urban Environment

5a. CONTRACT NUMBER
FA23860914076

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Radhakant Padhi

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Indian Institute of Science,Bangalore 560-012,India,India,IN,560-012

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AOARD, UNIT 45002, APO, AP, 96337-5002

10. SPONSOR/MONITOR'S ACRONYM(S)
AOARD

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AOARD-094076

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This paper describes the implementation of the guidance strategies Nonlinear Geometric Guidance (NGG) and Differential Geometric Guidance (DGG) while estimating the position of the obstacles with the noisy measurements from a 2D passive vision sensor with Extended Kalman Filter (EKF). The reactive obstacle avoidance algorithm has been validated from a number of simulation studies in three-dimensional scenario for UAV applications.

15. SUBJECT TERMS
Aircraft Control, Flight Control, UAV, Navigation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**80**	

Table of Contents

List of Figures

Abstract

Many of the UAV missions require them to fly in urban setting over low altitudes making them vulnerable to the collision with objects in their vicinity such as trees, buildings, power lines etc. So it is very crucial for UAVs to have autonomous obstacle detection and avoidance capability to complete their missions safely and successfully. In order to find a solution to this problem, first an elaborate literature survey has been conducted. This literature survey led to a fairly thorough understanding of the state-of-art techniques in the area of reactive obstacle avoidance algorithms for UAVs. For reactive obstacle avoidance, it is essential to sense the environment around UAV closely and continuously with some on-board sensor. A video camera is very popular such sensor. It is a 2D passive (vision) sensor and has many advantages such as compact size, low weight, low cost, and passive sensing capability. However, the data from vision sensor is usually noisy and inaccurate, hence must be filtered. So an Extended Kalman Filter (EKF) has been developed to get the best possible estimate of relative position of the obstacles and target with respect to the UAV based on noisy measurements from the vision sensor. After getting the obstacle position estimate from EKF, the Collision Cone approach has been applied to detect any incoming obstacle and to find a new aiming point in order to avoid it. Once the aiming point is known, the obstacle avoidance problem reduces to a guidance problem. Next, two recently proposed nonlinear guidance laws, Nonlinear Geometric Guidance (NGG) and Differential Geometric Guidance (DGG), have been applied for guidance purpose. This paper describes the implementation of the guidance strategies NGG and DGG while estimating the position of the obstacles with the noisy measurements from a 2D passive vision sensor with EKF. The algorithm developed has been validated from a number of simulation studies in three-dimensional scenario.

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs) are expected to be ubiquitous in the near future, autonomously performing complex military and civilian missions such as reconnaissance, environmental monitoring, border patrol, search and rescue operations, disaster relief, traffic monitoring etc [1].

Many of these applications require the UAV to fly at low altitudes in proximity with man made and natural structures such as buildings, trees, power lines etc. A collision with such structure would be fatal and would result in mission failure with the loss of the vehicle. Therefore, UAVs should have some sense of "situation awareness" [2] in order to successfully sense and avoid obstacles. At the same time they should be able to go ahead to pursue their original mission. This requires robust and computationally feasible obstacle avoidance algorithms to be implemented on-board the UAV. However, UAVs have limited computational resources. Following limitations exist for any system to be implemented onboard.

1. Size, weight and power consumption of onboard processor is limited. The processing power and memory available to the algorithm are also limited. Therefore, the algorithm must be computationally very efficient.
2. The algorithm must respond quickly, because it needs to be implemented in an online navigation system.
3. The payload of UAV is extremely limited. Therefore the sensor used to gain information from the surroundings must be lightweight. Also it should be energy efficient and economical.
4. UAVs may need to avoid detection for certain military missions like enemy reconnaissance. Thus active sensors, which send out energy into the environment, are not suitable.

In view of limitations (3) and (4), vision based navigation is a very suitable option. It is compact, lightweight, economical as well as a passive sensor. Therefore the obstacle avoidance algorithms should be able to perform well with vision information. However, the information from vision sensor is usually noisy and inaccurate. Hence algorithm must filter out any noise or at least minimize noise in order to get best possible estimate of the obstacle/target position so that reactive obstacle avoidance can work properly. At the same time algorithm must be computationally efficient to satisfy the above requirements as well.

UAV obstacle avoidance approaches can be classified into deliberative global path planning and reactive local obstacle avoidance algorithms. Global path planning algorithms assume complete knowledge of the environment beforehand and calculate an obstacle-free trajectory to the goal point. These methods take into account the dynamics of the UAV, and avoid known obstacles like buildings, trees etc. Additionally, the UAV constraints such as turn radius and kinodynamic constraints are also taken into account. Usually they attempt to find optimal paths. However, these methods are computationally expensive hence cannot be implemented online. Also they require complete knowledge of the surrounding environment which is not the case always. Local methods are reactive in nature, i.e. an onboard sensor detects the possibility of collision and then an alternative route is planned online. These consist of computationally efficient algorithms and are able to react quickly to changes in the environment. However these methods cannot guarantee optimality. Local navigation methods do not retain global information and continually react to changing environments. It is essential that local planning methods are fast and reliable, since the speed and reliability of the UAV is limited by them.

Recent obstacle avoidance approaches implement multi-layer architectures. A global planning layer first finds a dynamically feasible obstacle-free optimal path to the goal, and then a local obstacle avoidance layer reacts to changes in the environment and computes an alternate, obstacle-free path online. This scheme reduces the computation resources required since the amount of online computation carried out is much less. Optimality of the path may also be guaranteed, except for brief periods when the UAV

maneuvers away from the planned trajectory to avoid a local obstacle. This paper reviews some recently developed ideas of reactive obstacle avoidance algorithms. A literature survey of evolving philosophies on obstacle avoidance of UAVs is presented in Chapter 2.

Obstacle avoidance is often considered as another side of the target interception problem [3]. This is because once an incoming obstacle is detected, an alternate aiming point must be found in order to avoid it. After this, the problem reduces to one of finding a path which satisfies terminal conditions. This constitutes a guidance problem. The problem considered in this paper involves collision avoidance with stationary obstacles as they appear during flight. A goal point is considered after the obstacle region (which may be a way-point decided by a global planner). The objective is to avoid obstacles along the way and then reach the goal point.

The developed algorithm here, applies vision based sensor (video camera) to sense the surrounding environment and detect any incoming obstacle as well as to look for the target point or destination. During this study we assumed that an onboard image processor is available capable of detecting multiple obstacles simultaneously and distinguishing them. Since the measurements from a vision sensor are inaccurate, it is necessary to filter the noise out and get a very good estimate of the relative position of the obstacle before the guidance can be applied. The filtering technique has to be fast, computationally efficient, and implemented over the onboard processor. Since the measurement equation from a vision sensor is often nonlinear, an Extended Kalman Filter (EKF) is a very suitable technique for such application. It is recursive in nature hence can be implemented online with very efficient computing. On the other hand, EKF is "fragile" is nature, so care must be taken for the tuning of its parameter before any application. The fundamental assumption with this filter is that the true state is always sufficiently close to the estimated state. Hence the error dynamics can be represented fairly accurately by the linearized system about the estimated state. In other words, if initial estimation error is significantly high then it is very difficult to converge. Nonetheless, the EKF has been applied successfully for a range of problems over the past

decades. It is easy to implement, computationally efficient and works quite well with most of the problems. Once the obstacle position is estimated, next is to find out any potential collision with an obstacle and apply some fast guidance strategy in order to avoid it.

Two recently developed nonlinear guidance laws [4], named as Nonlinear Geometric Guidance (NGG) and Differential Geometric Guidance (DGG), are implemented here for guidance purpose. This navigation laws apply collision cone approach [3] to detect any possible collision and, if necessary, compute an alternate aiming point in order to avoid it. The guidance algorithms then attempt to quickly align the velocity vector of the vehicle along the aiming point within a part of the available time-to-go, which ensures quick reaction and hence safety of the vehicle. The main advantage of these guidance laws is that they effect high maneuvering at the beginning, causing the velocity vector of the UAV to align with the aiming point direction quickly and then settle along it. Therefore, there is no need to maneuver all the way until the aiming point is reached. Such a strategy ensures quick reaction and hence safety of the vehicle by quickly avoiding obstacle. Note that after the obstacles are avoided, the destination point also serves as another aiming point and hence the same guidance is also applied for reaching the destination when the path is free of obstacles. Therefore these guidance laws achieve reactive obstacle avoidance as well as destination-seeking.

The obstacle position estimation technique with visual information developed in this paper has been integrated with the recently developed NGG and DGG navigation laws. This paper validates these guidance strategies with vision sensing from a number of simulation studies in various three-dimensional scenarios.

Chapter 2

Literature Survey

This chapter presents a literature survey on recently evolving ideas for reactive obstacle avoidance for UAVs. Unlike Global Path Planning Algorithms, reactive obstacle avoidance algorithms deal only with the problem of avoiding collisions with obstacles as and when they appear during the flight. These algorithms do not require global information i.e. they do not require knowledge of the entire environment, or the initial and goal points. The information about the immediate environment and nearby obstacles is provided to the algorithm by onboard sensors. This information is often sufficient to compute an avoidance maneuver for the UAV. It must be emphasized that these algorithms can be imbedded into any global path planning algorithms, under the assumption that after avoiding the obstacle, the UAV comes back to the global path as soon as possible.

2.1 Nonlinear Model Predictive Control Approach

Model Predictive Control (MPC) [5] has gained popularity in recent years as a control approach for nonlinear dynamical systems. This approach handles realistic system constraints such as input saturation and state constraints and is found to be suitable for path-planning problems in complex environments [6]. An MPC scheme is implemented by [7] to achieve online obstacle avoidance in UAVs. Since MPC performs online optimization over a finite receding horizon, it can account for future environment changes. Obstacle avoidance is built into the optimization problem, which performs reference trajectory tracking and obstacle avoidance in a single module. When a collision is predicted to occur, a safe trajectory that avoids the impending collision is computed (Figure 2.1). In the MPC formulation, an optimal control input sequence over a finite receding horizon N must be found that minimizes a cost function [8] that is to find

$$u(t_k), k = i, i+1, i+N-1 \tag{2.1}$$

such that

$$u(t_k) = \arg\min V(x, t_k, u) \tag{2.2}$$

where

$$V(x,t_k,u)=\sum_{i=t_k}^{t_{k+N-1}} L(x(i),u(i))+F(x(t_{k+N})) \tag{2.3}$$

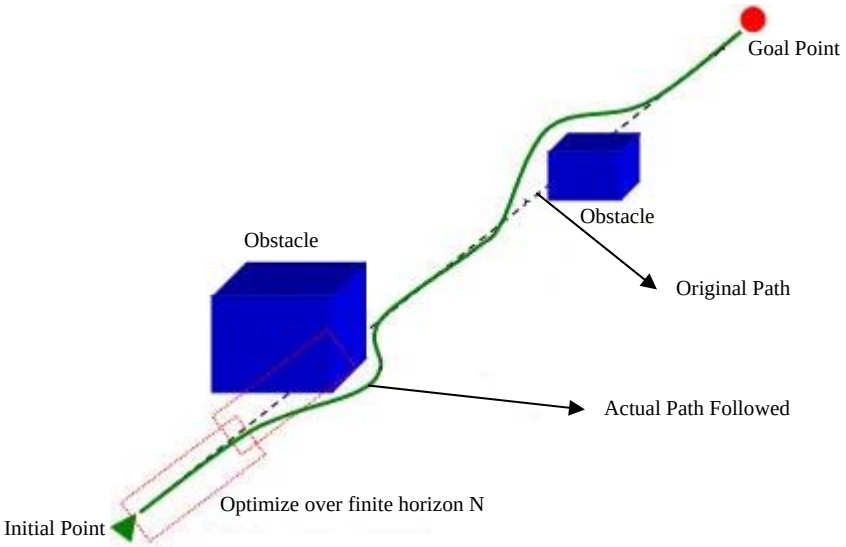

Figure 2.1: Trajectory tracking and re-planning using MPC

L is a positive definite cost function and F is the terminal cost. The cost function L is chosen as:

$$L(x,u)=\frac{1}{2}(x_r-x)^T Q(x_r-x)+\frac{1}{2}u_r^T R u_r+S(x)+\sum_{l=1}^{O} P(x_v,\eta_l) \tag{2.4}$$

The first term in the cost function ensures that any deviation from the reference state results in a large value of cost, and therefore is penalized. Similarly the second term penalizes high control inputs. The term *S(x)* penalizes states that are outside the allowable range. The last term is a potential function term to be included for obstacle avoidance. It is chosen as follows:

$$P(x_0,\eta_l)=\frac{1}{(x_v-\eta_l)^T G(x_v-\eta_l)+\varepsilon} \tag{2.5}$$

where $x_v \in R^3$ is the position of the UAV, and η_l is the position of the l^{th} obstacle out of O obstacles. This penalty function increases as $\| x_v-\eta_l \|$ decreases. G is positive definite and ε is a quantity that is kept positive to prevent P from being singular when $\| x_v-\eta_l \| \to 0$. The potential function term may be chosen to be enabled only if $\| x_v-\eta_l \| < \sigma_{\min}$, a minimum safety distance to be maintained. A new trajectory is then planned. Including collision avoidance into the optimization step reduces the risk of the

UAV falling into a local minimum, since MPC looks ahead and optimizes over a finite horizon.

The obstacle's predicted position after $k-N-1$ steps is required (N is the horizon) in order to avoid the obstacle. If the current position and velocity v_l of the obstacle may be estimated, the position of the obstacle after N_p steps (prediction horizon) may be found at the k^{th} step:

$$\eta_l(t_{k+i}) = \eta_l(t_k) + \Delta t v_l(t_k)(t_{i-1}) \tag{2.6}$$

Control saturation is taken into account during the online optimization process. Additionally a tracking feedback controller in the loop will track the reference without any error in the presence of modeling uncertainties.

A dual mode strategy is followed in order to track a reference trajectory, as well as avoid collisions. In normal flight, parameters are chosen so as to achieve tracking performance and good stability. In the emergency evasive maneuver case, the parameters are chosen so as to generate a trajectory that will avoid collision at all cost. During evasion, large control effort and large deviations from reference are allowed. Results of simulations in [8] indicate that this method is capable of avoiding hostile obstacles flying at high speeds (100 KM/H) and with different heading angles. This method was implemented successfully in unmanned helicopters. A disadvantage of this method is that MPC is quite computation intensive and may not be suitable for online implementation on small UAVs. An extension of this approach can be towards collision avoidance with maneuvering obstacles. This would require an estimator and some knowledge of the dynamics of the obstacle.

2.2 Vision Based Approach

Vision based navigation, guidance and control has become one of the most focused research area in automation of robots and UAVs. It is efficient to use a videos sensor since it is small, light-weight, economical and power efficient. Increasing computational

power of small processors and consequent improvement in quality of digital image analysis is another motivation for using vision based approach. Additionally, in nature most of birds and insects use vision for collision avoidance and navigation. Following are the different vision based algorithms developed to address the problem of autonomous obstacle avoidance by UAVs.

2.2.1 Neural Network Approach

A vision based Grossberg Neural Network (GNN) [9] scheme is used for local collision avoidance [10]. The GNNs are nonlinear competitive neural networks. They are able to explain the working of human vision, and have been used in a variety of vision based applications, especially in pattern recognition [11]. GNNs can be used for UAV vision based navigation. A combination of Visibility Graphs and GNNs is used to achieve online collision avoidance.

A two-layer, dual-mode control architecture achieves a formation of UAVs as well as collision avoidance. The top layer generates a reference trajectory and the lower layer tracks this reference taking into account the dynamics of the vehicle. In an obstacle-free environment, “Safe Mode" operation is carried out, which develops and maintains a formation of UAVs. When obstacles are detected using an on-board sensor, the “Danger Mode" is activated, which finds the shortest path out of the danger zone. In the “Safe Mode", the reference trajectory is to be generated, so that the UAVs achieve and maintain the desired formation. The relative dynamics between the UAVs is used to develop an infinite time optimal scheme [12] of formation in a centralized way. In order to achieve this, the following cost function is to be minimized:

$$J = \int_{t}^{\infty} [(x_r - x_d)^T Q (x_r - x_d) + u_r^T R u_r] \, dt \tag{2.7}$$

Subject to

$$\dot{x}_r = A_r x_r + B_r u_r \tag{2.8}$$

x_r is the relative state (relative position and relative velocity between two UAVs) and x_d is the desired value of state. u_r is the relative control i.e. the resultant acceleration

between two UAVs. This formulation attempts to attain a formation as well as minimize the control effort for this. The reference trajectory is generated online at every step. The danger mode is activated when an obstacle is sensed. In this situation, the formation is allowed to break. The UAVs must avoid collisions with obstacles as well as with the other UAVs in the formation. The danger mode operation uses a combination of Visibility Graphs [13] and vision based Grossberg Neural Networks (GNN). A buffer zone is created around the obstacles. A visibility graph of the environment is formed by connecting all mutually visible vertices of the obstacle buffer zones together. In two dimensional environments the shortest distance paths are obtained by moving in straight lines and turning at the vertices of obstacles. Therefore, as part of the GNN, neurons are placed at the vertices of each obstacle's buffer zone. Figure 2.2 shows neuron placement, visibility graph and the re-planned trajectory.

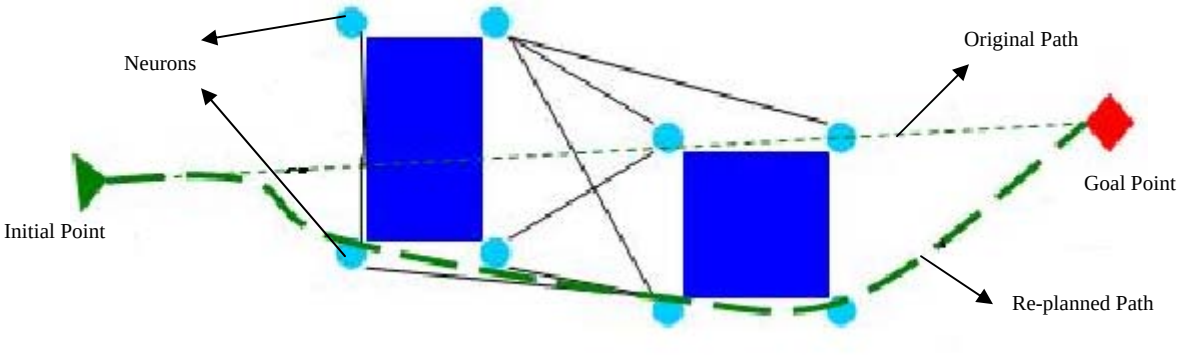

Figure 2.2: Danger Mode operation using visibility graph and GNNs

The activity of each neuron depends upon excitation received from other neurons as well as excitation from the goal point. The activity is calculated from a shunting equation:

$$\frac{dx_i}{dt} = -ax_i + (b - x_i)(E + \sum_{j=1}^{k} w_{ij} x_j) \tag{2.9}$$

where

$$E = E_1 + E_2 \tag{2.10}$$

$$E_1 = \frac{\alpha}{d_p} \tag{2.11}$$

$$E_2 = \begin{cases} 100, if\ the\ neuron\ is\ on\ destination \\ 0, otherwise \end{cases} \tag{2.12}$$

x_i is the activity of the i^{th} neuron, $w_{ij}x_j$ is excitation due to neighboring neuron, where $w_{ij} = \mu / d_{ij}$, d_{ij} is the distance between UAVs i and j, d_p is the perpendicular distance of the vertex from the line joining the UAV and the target. E is the excitation composed of two parts. E_1 is the excitation due to closeness of the vertex from the present path. α and μ are weights that must be tuned correctly so that the deviation from current path and closeness to goal are weighed correctly. The neurons nearest to the goal and nearest to the current path have the highest values of activity. Thus, by following the vertices with highest activities, the UAV is able to get out of the danger mode. The path followed will be the shortest distance path. Collision avoidance with other UAVs is achieved in the following way. When a potential collision is sensed, the UAV with lower index creates a buffer zone around the UAV with higher index and attempts to avoid it.

In the lower layer, tracking the reference generated by both the Safe Mode and the Danger Mode is done using a Model Predictive Controller (MPC) [14]. This method consists of finding the optimal control input sequence to minimize a cost function at every step. The cost function here is formulated such that the actual output tracks the reference output, along with control minimization. This method also takes into account practical vehicle constraints. The cost function to be minimized at the k^{th} step is given by following equation:

$$J_k = [y(t_k) - y_d(t_k)]^T Q_k [y(t_k) - y_d(t_k)] + \Delta U(t_k) R_k \Delta U(t_k) \tag{2.13}$$

y is the actual output, y_d is the desired output and ΔU is the control history. Q_k and R_k are weighting matrices to be chosen appropriately. Simulation results in [10] show that the UAVs developed a desired formation and successfully re-planned trajectories to avoid obstacles along the way. Cooperative collision avoidance among UAVs is also achieved. A possible extension of this method is the case of non-cooperative collision avoidance. This can be implemented with an estimator to find the hostile obstacles' velocity and position.

2.2.2 EKF Estimation Approach

In their research work, [15] focuses on vision-based navigation and guidance system design for UAVs to detect and avoid unforeseen obstacles while executing a waypoint tracking mission. Since the vision-based measurements are noisy and inaccurate, it is necessary to filter out the noise before applying the guidance. Additionally, visual measurements are also a nonlinear function of the relative state. Hence EKF is a very suitable choice to design the navigation filter. The UAV motion dynamics are modeled as following equations:

$$\dot{X}_v = \begin{bmatrix} \dot{x}_v \\ \dot{y}_v \\ \dot{z}_v \end{bmatrix} = \begin{bmatrix} u_v \\ v_v \\ w_v \end{bmatrix} = V_v \tag{2.14}$$

$$\dot{V}_v = \begin{bmatrix} \dot{u}_v \\ \dot{v}_v \\ \dot{w}_v \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = a \tag{2.15}$$

here $a_x = 0$, i.e. velocity is constant in X-direction. The vehicle is controlled by only lateral acceleration a_y and vertical acceleration a_z. A camera is mounted on the vehicle and its attitude is assumed to be known in the form of a rotation matrix from the local frame F_L to the camera frame F_C, which is denoted by L_{CL}. Let $X_{wp} = [x_{wp} \quad y_{wp} \quad z_{wp}]^T$ be a given waypoint location in F_L. Then the waypoint tracking problem is achieved if

$$y_v(t_f) = y_{wp}, z_v(t_f) = z_{wp} \tag{2.16}$$

where t_f is a time at which $x_v(t_f) = x_{wp}$ is satisfied. Let X_{obs} be obstacle's position in F_L and assume $\dot{X}_{obs} = 0$, i.e., stationary obstacles. Then the obstacle's relative motion dynamics with respect to the vehicle is written by

$$\dot{X} = \dot{X}_{obs} - \dot{X}_v = -V_v \tag{2.17}$$

where $X = X_{obs} - X_v = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is a relative position vector in F_L. For collision avoidance, the vehicle is required to keep a minimum distance d from every obstacle. That is, a collision-safety boundary becomes a spherical surface with a radius d and a center at X_{obs}. Therefore, a mission given to the vehicle is to satisfy (2.16) while always maintaining $\| X \| \geq d$ for all obstacles. However, the obstacle's location X_{obs} is unknown

to the vehicle, and so the relative position X is also unknown. Hence, for obstacle avoidance, the guidance system can only use its estimate which is obtained through a 2D vision sensor.

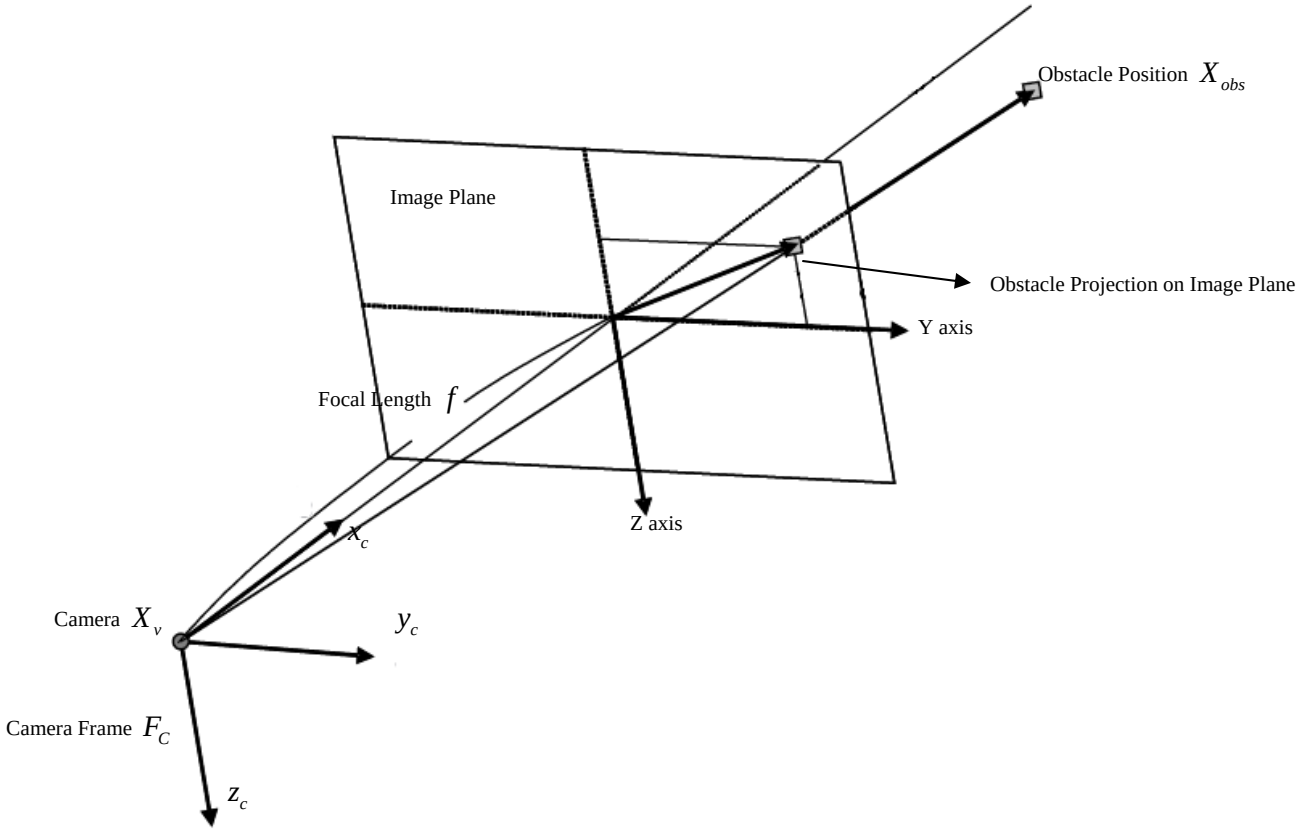

Figure 2.3: Pin-Hole Camera Model

The camera frame F_C is taken so that the X_C-axis aligns with the camera's optical axis. Let $X_C = L_{CL} X = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^T$ be the relative position vector expressed in F_C. Assuming the pin-hole camera model shown in Figure 2.3, the 2-D measurement of the obstacle position in an image plane at a k^{th} time step is given by

$$z_k = \frac{f}{x_{ck}} \begin{bmatrix} y_{ck} \\ z_{ck} \end{bmatrix} + \upsilon_k = h(X_{ck}) + \upsilon_k \tag{2.18}$$

where f is a focal length of the camera and υ_k is a zero mean Gaussian discrete white noise process with covariance matrix $R_k = \sigma^2 I$.

Since the measurement model (2.18) is nonlinear with respect to the relative state, an EKF is applied to estimate the relative position vector X of each obstacle. The EKF for

the process model (2.14), (2.15) and the measurement model (2.18) is formulated as follows.

- Update: The EKF update procedure is performed by using the residual between the actual measurement and the predicted measurement.

$$\hat{X}_k = \hat{X}_k^- + K_k\left(z_k - \hat{z}_k^-\right) \tag{2.19}$$

$$P_k = P_k^- - K_k H_k P_k^- \tag{2.20}$$

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{2.21}$$

where $\hat{X}_k$ is an updated estimate of X at a k^{th} time step and P_k is its error covariance matrix. K_k is a Kalman gain. $\hat{X}_k^-$ and P_k^- are a predicted estimate and its error covariance matrix. A predicted measurement is obtained by $\hat{z}_k^- = h(L_{CLk}\hat{X}_k^-)$ where L_{CLk} is a camera attitude at the k^{th} time step. And a measurement matrix H_k is calculated by

$$H_k = \left.\frac{\partial h(L_{CLk}X)}{\partial X}\right|_{X=\hat{X}_k^-} = \frac{1}{\hat{x}_{ck}^-}\begin{bmatrix} -\dfrac{y_{ck}^-}{\hat{x}_{ck}^-} & 1 & 0 \\ -\dfrac{z_{ck}^-}{\hat{x}_{ck}^-} & 0 & 1 \end{bmatrix} L_{CLk} = \frac{1}{\hat{x}_{ck}^-}\begin{bmatrix} -h(\hat{X}_{ck}^-) & I \end{bmatrix} L_{CLk} \tag{2.22}$$

- Prediction: The EKF prediction procedure propagates the updated estimate obtained at a current time step k to the next time step $k+1$ through the process model (2.23), (2.24).

$$\hat{X}_{k+1}^- = \hat{X}_k + V_{vk}\Delta t_k + \frac{1}{2}a_k\left(\Delta t_k\right)^2 \tag{2.23}$$

$$P_{k+1}^- = \Phi_k P_k \Phi_k^T + Q_k \tag{2.24}$$

where $\Delta t_k = t_{k+1} - t_k$ is a sampling time. Φ_k is a state transition matrix and which can be approximated by

$$\Phi_k \simeq I$$

for stationary obstacles when Δt_k is sufficiently small, Q_k is a covariance matrix of the process noise. The form $Q_k = \sigma_X^2 I \cdot \Delta t_k$ is used in the filter design.

Since there can be multiple obstacles in the vehicle's surroundings, the image processor may detect more than one obstacle in the same image frame. In order to update each estimate correctly, it is very important to create a right correspondence between the measurements and the estimates before applying the EKF procedure. The statistical z-test [16] is used for this purpose.

The problem of flying a UAV with vision based guidance can be seen as a control minimizing Proportional Navigation (PN) guidance problem in the absence of obstacles. PN Guidance is used in missile guidance, [17] and the problem of a UAV pursuing its goal may be interpreted as a similar problem. But when obstacles are to be avoided, multiple PN guidance problems need to be solved. A collision avoidance scheme based on PN guidance is presented in [18]. However, this scheme leads to a jump in the control effort every time a new target is pursued. Instead, a single Minimum Effort Guidance (MEG) approach minimizes the control effort along with avoiding collisions for multiple targets [19]. In other words, the control effort is minimized for the entire trajectory from the initial point to the goal point via the obstacle aiming point, leading to a lower overall control effort. A collision cone approach is used to detect potential collisions, for which an Extended Kalman Filter (EKF) [20] is used to estimate the relative distance from the UAV to the obstacle.

The PN guidance law is derived by solving the following optimization problem for control minimization for each aiming point obtained:

$$\min J_{oa} = \frac{1}{2}\int_{t_0}^{t_{go}} a^T(t)a(t)\,dt = \frac{1}{2}\int_{t_0}^{t_{go}} (a_y^2(t) + a_z^2(t))\,dt \tag{2.25}$$

Subject to vehicle dynamics with terminal conditions given in (2.16), the resulting optimal guidance law is

$$a_{oa}(t) = -3\left(\frac{1}{(\hat{t}_{go} - t_o)^2}\begin{bmatrix} 0 \\ y_v(t_0) - \hat{y}_{ap} \\ z_v(t_0) - \hat{z}_{ap} \end{bmatrix} + \frac{1}{\hat{t}_{go} - t_o}\begin{bmatrix} 0 \\ v(t_0) \\ w(t_0) \end{bmatrix}\right) \tag{2.26}$$

The optimal control obtained from the PN Guidance method is only piecewise continuous, with a jump between targets. MEG handles all the terminal conditions within one problem [21]. The optimal control is continuous and piecewise linear. This method, therefore, yields a lower cost. The optimization problem remains the same and all the terminal conditions are considered in the problem. The control law is found by cubic interpolation of the single condition case. The resulting optimal control law is

$$a(t_0) = a_{oa}(t) - \frac{3}{3(\hat{t}_{go} - t_o) + 4(t_f - \hat{t}_{go})} \left(\begin{bmatrix} 0 \\ v(t_0) \\ w(t_0) \end{bmatrix} + \frac{3}{(\hat{t}_{go} - t_o)} \begin{bmatrix} 0 \\ y_v(t_0) - \hat{y}_{ap} \\ z_v(t_0) - \hat{z}_{ap} \end{bmatrix} - \frac{2}{(t_f - \hat{t}_{go})} \begin{bmatrix} 0 \\ \hat{y}_{ap} - y_d \\ \hat{z}_{ap} - z_d \end{bmatrix} \right) \tag{2.27}$$

In [19], both PN Guidance and Minimum Effort Guidance are used to solve the same problem. The cost is found to be lower in MEG. Therefore, MEG is found to be a better method in terms of the control minimization achieved. However, in case of obstacle avoidance, vehicle safety has the foremost priority so minimum control effort is not a requirement. Also MEG maneuvers the UAV till it reaches the aiming point which is quite risky given the position of obstacle is not known with full certainty. So UAV velocity vector should be aligned along the aiming point as quick as possible in some part of time-to-go.

2.2.3 Optic Flow Based Approach

The obstacle detection through optic flow is an already widely studied problem [22]. The objective here is to explain changes in an image sequence as the result of a motion field, given an image sequence $I_t(x, y)$, the optical flow $\left(v_x, v_y\right)$ has to match the following optical flow equation:

$$\frac{\partial I_t}{\partial x} v_x + \frac{\partial I_t}{\partial y} v_y + \frac{\partial I_t}{\partial t} = 0 \tag{2.28}$$

No point wise resolution of the optical flow equation is possible since on each location and each time, this would consist in solving a single scalar equation from two scalar unknowns. This is known as the aperture problem. [23] and [24] survey a number of ways to compute optical flow of a changing image. The most appealing algorithms tend to use wavelets to interpolate the sparse flow data in between the borders.

The problem of collision avoidance has been addressed by the use of optic flow in various projects [25]. There is a relation between time to impact and optic flow. Optic Flow during motion towards an object is a direct measure for time to impact $t = \frac{d}{v_t}$. In practice this temporal distance is very useful to trigger a well-dosed reflex to avoid a collision. A high flow meaning high risk, this high optic flow can be used as trigger for a collision avoidance maneuver.

2.3 Summary of Literature Review

Much of the benefits of deploying UAVs can be derived from autonomous missions. Path planning with obstacle avoidance is an important problem which needs to be addressed to ensure safety of such vehicles in autonomous missions. An attempt has been made in this paper to present a brief overview of a few promising and evolving ideas such as model predictive control, vision based algorithms, minimum effort guidance etc. As mentioned earlier, there are several requirements that an algorithm must satisfy in order to solve the online obstacle avoidance problem completely. A few key issues that need to be addressed in a good collision avoidance algorithm include:

- Collision avoidance with fixed obstacles, as they appear during the flight without any or little priori information
- Solution of the problem taking the vehicle dynamics into account, including state and input constraints (many of the current algorithms are based on only kinematics)
- Development of fast algorithms, which can be implemented online with limited onboard processing capacity

- Capability to sense and avoid small obstacles (such as electric power lines, small birds etc)
- Robustness for issues such as limited information of the environment, partial loss of information etc

In addition to the above issues, there are many other issues for successful deployment of UAVs, such as requirement for light weight equipments, power efficiency (for high endurance), stealthness etc. Although an attempt has been made in this paper to give an overview of some of the recently proposed techniques which partially address some of these issues, promising algorithms satisfying many of these requirements simultaneously is yet to be developed. Additionally, some of the assumptions behind the proposed algorithms (such as non-maneuvering constant speed flying objects, appearance of one obstacle at a time, perfect information about the environment etc.) are not realistic and hence need to be relaxed. A lot of research is being carried out worldwide to design collision avoidance systems that address many of these important concerns.

Chapter 3
Obstacle Position Estimation Using EKF

In this section we describe the formulation of the obstacle position estimation problem. The motivation behind using vision based sensing is presented first. The assumptions made in this study are explained next. Then the problem geometry is described. Next section provides the summary of continues-discrete form of Extended Kalman Filter (EKF). Section 3.3 explains in detail how EKF is applied for the problem of obstacle position estimation with vision based measurements.

3.1 Vision Based Sensing

As stated earlier, certain limitations exist for any system or algorithm to be implemented onboard a small flying vehicle. First limitation is weight constrain i.e. any sensor used for obstacle detection must be light weight otherwise it might seriously compromise the UAV's flying and maneuvering capability. Second limitation is power consumption. Generally small UAVs are battery powered and they do not have much extra power available onboard. Power consumed by obstacle avoidance system will also limit the UAV flying time i.e. limiting the UAVs effectiveness for many missions. Another limitation, which concerns the military missions, is stealthness i.e. UAV should not be detectable while operative inside enemy aerospace. Thus sensor applied must be passive in nature i.e. while sensing the surrounding it should not send energy signal into the environment.

A small video camera addresses all these concerns. They are compact, light-weight, power efficient, economical as well as passive sensors. These advantages make a video camera very popular choice in such application. Increasing computational power of small processors and consequent improvement in quality of digital image processing is another motivation for applying vision based navigation. Additionally vision based navigation is a prevalent phenomenon in nature where almost all kinds of birds and insects exclusively use vision for detection and navigation.

3.1.1 Assumptions Made in this Study

Motivated by the increasing computer power, the improving quality of digital imaging and the increasing performance of digital image analysis, this algorithm assume that an image processor is available which is capable of detecting multiple obstacles and target, simultaneously from the images obtained from 2D vision sensor. It is assumed that obstacles are being point obstacles with known safety radius. The image processor can make corresponds between the present measurements of obstacles and their previous estimates. It is also capable of distinguishing between obstacles and target point (destination).

3.1.2 Problem Geometry

Figure 3.1 shows the problem geometry. For simplicity it is assumed that camera is fixed at UAV's center of mass and the UAV knows its own position with reasonable certainty with the help of GPS and/or INS. The relationship between the locus of obstacle's projection on image plan and its relative position can be easily shown from Figure 3.1 using symmetrical triangles as given by (3.1).

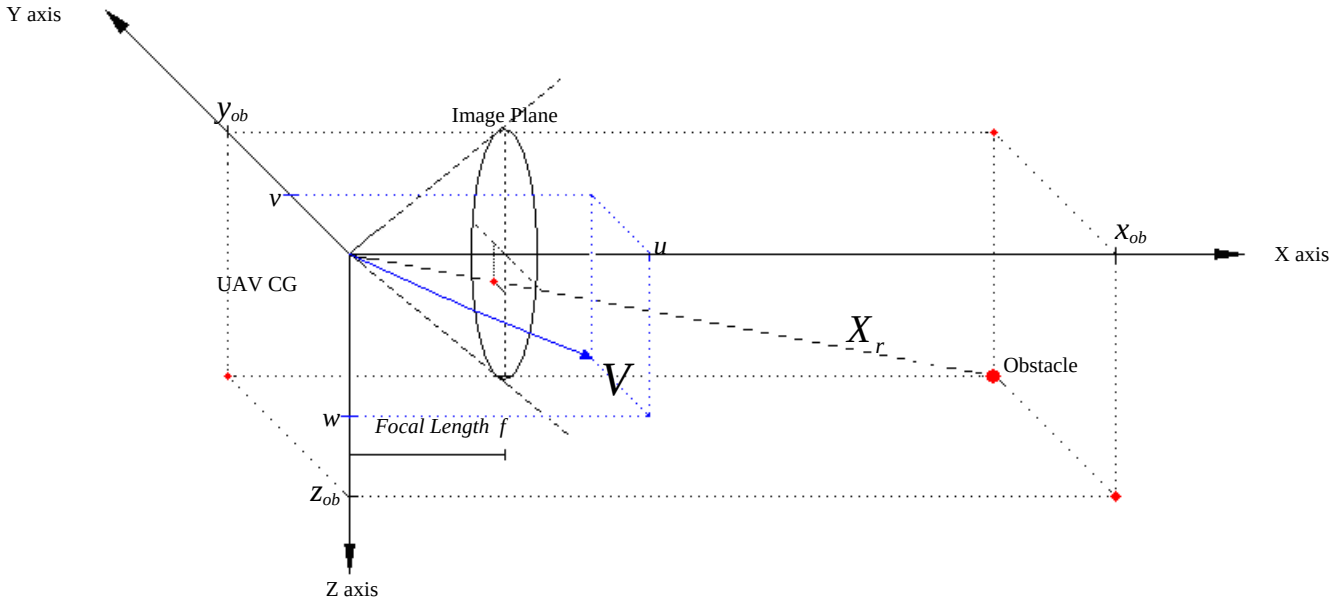

Figure 3.1: Video Sensor Model

$$Y_k = \begin{bmatrix} y_k^i \\ z_k^i \end{bmatrix} = \frac{f}{x_{ob_k}} \begin{bmatrix} y_{ob_k} \\ z_{ob_k} \end{bmatrix} \tag{3.1}$$

here f is focal length of the camera, $X_{r_k} = [x_{ob_k} \quad y_{ob_k} \quad z_{ob_k}]^T$ is relative position of the obstacle with respect to the UAV, $V = [u \quad v \quad w]^T$ is the UAV velocity vector and $Y_k = [y_k^i \quad z_k^i]^T$ is the locus of the obstacle projection on the image plan at time instant k. Objective here is to get the relative position of obstacle X_{r_k} based on measurement Y_k. Note here, equation (3.1) is a nonlinear function of relative state X_r. Additionally measurement noise is also present, to account for that (3.1) can be rewritten as following equation:

$$Y_k = \begin{bmatrix} y_k^i \\ z_k^i \end{bmatrix} = \frac{f}{x_{ob_k}} \begin{bmatrix} y_{ob_k} \\ z_{ob_k} \end{bmatrix} + v_k \tag{3.2}$$

here v_k is the measurement noise at time instant k. Due to nonlinearity and noise, it is necessary apply some filtering technique to get the best possible estimate of X_{r_k} from measurement Y_k. Extended Kalman Filter (EKF) is once such technique; next section provides the summary of the EKF in the context of our application.

3.2 Summary of Extended Kalman Filter

A large class of estimation problems involves nonlinear models. For several reasons, state estimation for nonlinear systems is considerably more difficult and admits a wider variety of solutions than the linear problem [20]. In a vast majority of nonlinear models (including ours), system states are continuously evolving and measurements are available only at discrete intervals of time. Therefore, continuous-discrete form of EKF best describes our system model. The nonlinear system dynamics and measurement model can be given by following equations:

$$\dot{X}(t) = f(X(t), U(t), t) + G(t)w(t) \tag{3.3}$$

$$Y_k = h(X_k) + v_k \tag{3.4}$$

Where, function f represents the system dynamics which is a nonlinear function of state X, the deterministic control input U, and time t. The process noise (or disturbance

input) $w(t)$ is white, zero mean Gaussian random process. The statistical properties of process noise w can be written as:

$$\begin{aligned} E\left[w(t)\right] &= 0 \\ E\left[w(t)w^T(\tau)\right] &= Q(t)\delta(t-\tau) \end{aligned} \tag{3.5}$$

The function h represents the measurement equation which is a nonlinear function of the state X. The measurement noise v is also white, zero mean Gaussian random process that is uncorrelated with process noise. The statistical properties of process noise v can be written as following equations:

$$\begin{aligned} E\left[v_k\right] &= 0 \\ E\left[v_k {v_j}^T\right] &= R_k\delta_{k-j} \\ E\left[w(t){v_k}^T\right] &= 0 \end{aligned} \tag{3.6}$$

The basic assumption in EKF is that the true state is sufficiently close to the estimated state. Therefore, the error dynamics can be represented fairly accurately by a linearized first-order Taylor series expansion. So we linearize the nonlinear system around the Kalman filter estimate of the state. The linearized matrices are given as following equations:

$$A(t) = \left.\frac{\partial f}{\partial X}\right|_{\hat{X}(t)} \tag{3.7}$$

$$C_k = \left.\frac{\partial h}{\partial X}\right|_{\hat{X}_k^-(t)} \tag{3.8}$$

The expected values of the initial state and its covariance are assumed known and given by following equations:

$$\hat{X}_0 = E\left(X_0\right) \tag{3.9}$$

$$P_0 = E\left[\left(X_0 - \hat{X}_0\right)\left(X_0 - \hat{X}_0\right)^T\right] \tag{3.10}$$

A weighting factor called Kalman filter gain computed using the covariance information, measurement covariance matrix and linearized measurement matrix given by (3.8). The Kalman filter gain is used to combine the propagated estimate with the new

measurement. This gain is defined in such a way that it minimizes the estimation error covariance after the update. The Kalman filter gain is given by following equation:

$$K_k = P_k^- C_k^T \left[C_k P_k C_k^T + R_k \right]^{-1} \tag{3.11}$$

New measurements are combined with the propagated state estimate to generate an updated estimated state. The state covariance is also updated to reflect the information gained through the measurement. The state and covariance update equations are:

$$\hat{X}_k^+ = \hat{X}_k^- + K_k \left[Y_k - h(\hat{X}_k^-) \right] \tag{3.12}$$

$$P_k^+ = (I - K_k C_k) P_k^- (I - K_k C_k)^T + K_k R_k K_k^T \tag{3.13}$$

Between measurements, EKF propagates the state estimate and its covariance using the known nonlinear dynamics by following equations:

$$\dot{\hat{X}}(t) = f(\hat{X}(t), U(t), t) \tag{3.14}$$

$$\dot{P}(t) = A(t)P(t) + P(t)A(t)^T + G(t)Q(t)G^T(t) \tag{3.15}$$

The EKF technique requires initialization of states and covariance, computation of Kalman filter gain, update of states and covariance followed by propagation of states and covariance. This process is followed in a cyclic manner except for the initialization part. Equation from (3.3) to (3.15) constitutes the EKF technique for continuous-time nonlinear systems with discrete measurements.

3.3 Obstacle Position Estimation using EKF

As stated earlier, the job of the EKF here is to get a good estimate of the obstacle position from noisy vision measurements. EKF is a very widely used technique for state estimation in nonlinear systems. This section provides details on how EKF is applied here.

3.3.1 System Dynamics and Measurement Equation

Initially following system dynamics are considered for UAV motion modeling, where X is UAV's position vector, V is its velocity vector and a is the control vector.

$$\dot{X} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = V \tag{3.16}$$

$$\dot{V} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = a \tag{3.17}$$

Furthermore, the velocity along X-axis is considered to be constant i.e. $a_x = 0$. The UAV is modeled as a point mass, so video sensor (camera) is assumed to be fixed at the UAV's center of gravity and its orientation with respect to an inertial frame of reference is known. The UAV system dynamics given by (3.16) & (3.17) is linear while the measurement equation (3.1) is quite nonlinear function of the state vector X_r. It is not a good idea to have a nonlinear measurement equation while applying EKF [20], since its partial differentiation can lead to quite complicated Jacobean. Particularly (3.1) can lead to singularity issue as UAV approaches close to the obstacle $x_{ob} \to 0$. So to work around these issues, it is better to shift from Cartesian system to Spherical coordinate system i.e. $X_r = \begin{bmatrix} r_{ob} & \theta_{ob} & \phi_{ob} \end{bmatrix}^T$. Now we measure the locus of obstacle projection on image plane in terms of angles θ_k^i and ϕ_k^i, instead of measuring in terms of y_k^i and z_k^i (Figure 3.2).

So measurement becomes a linear function of the relative obstacle position given by following:

$$Y_k = \begin{bmatrix} \theta_k^i \\ \phi_k^i \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{ob_k} & \theta_{ob_k} & \phi_{ob_k} \end{bmatrix}^T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X_{r_k} \tag{3.18}$$

Since state vector X_r has been changed from Cartesian to Spherical coordinate system, the system dynamics should also be changed accordingly.

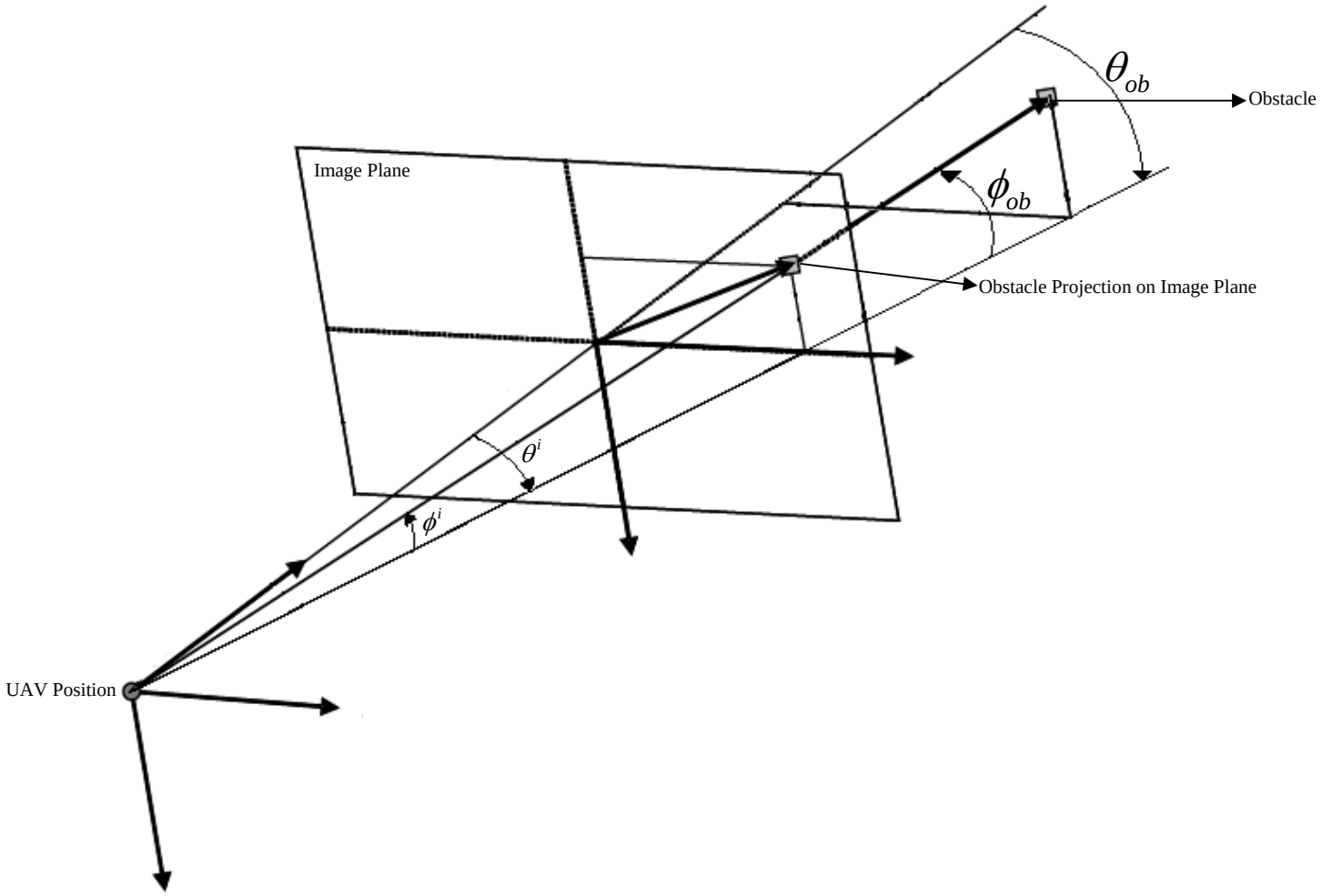

Figure 3.2: Measurement of Obstacle Projection on Image Plane in terms of θ and ϕ

The well known relationship between Cartesian and Spherical coordinates is given by following two sets of equations (3.19) and (3.20):

$$r^2 = x^2 + y^2 + z^2 \tag{3.19a}$$

$$\tan\theta = \frac{y}{x} \tag{3.19b}$$

$$\tan\phi = \frac{z}{\sqrt{x^2 + y^2}} \tag{3.19c}$$

and

$$x = r\cos\theta\cos\phi \tag{3.20a}$$

$$y = r\sin\theta\cos\phi \tag{3.20b}$$

$$z = r\sin\phi \tag{3.20c}$$

differentiating (3.19a)

$$2r\dot{r} = 2x\dot{x} + 2y\dot{y} + 2z\dot{z}$$

since $\dot{x} = u$, $\dot{y} = v$, and $\dot{z} = w$

$$r\dot{r} = xu + yv + zw$$

after substituting (3.20a), (3.20b) and (3.20c)

$$r\dot{r} = ur\cos\theta\cos\phi + vr\sin\theta\cos\phi + wr\sin\phi$$

$$\dot{r} = u\cos\theta\cos\phi + v\sin\theta\cos\phi + w\sin\phi \tag{3.21}$$

Similarly differentiating (3.19b)

$$\sec^2\theta\,\dot{\theta} = -\frac{y}{x^2}\dot{x} + \frac{1}{x}\dot{y}$$

$$\sec^2\theta\,\dot{\theta} = -\frac{y}{x^2}u + \frac{1}{x}v$$

after substituting (3.20a), (3.20b) and (3.20c)

$$\sec^2\theta\,\dot{\theta} = -\frac{r\sin\theta\cos\phi}{(r\cos\theta\cos\phi)^2}u + \frac{1}{r\cos\theta\cos\phi}v$$

$$\dot{\theta} = -\frac{\sin\theta}{r\cos\phi}u + \frac{\cos\theta}{r\cos\phi}v \tag{3.22}$$

Finally differentiating (3.19c)

$$\sec^2\phi\,\dot{\phi} = -\frac{xz}{(x^2+y^2)^{3/2}}\dot{x} - \frac{yz}{(x^2+y^2)^{3/2}}\dot{y} + \frac{1}{\sqrt{x^2+y^2}}\dot{z}$$

$$\sec^2\phi\,\dot{\phi} = -\frac{xz}{(x^2+y^2)^{3/2}}u - \frac{yz}{(x^2+y^2)^{3/2}}v + \frac{1}{\sqrt{x^2+y^2}}w$$

after substituting (3.20a), (3.20b) and (3.20c)

$$\dot{\phi} = -\frac{\cos\theta\sin\phi}{r}u - \frac{\sin\theta\sin\phi}{r}v + \frac{\cos\phi}{r}w \tag{3.23}$$

By combining (3.21), (3.22) & (3.23), the system dynamics become a nonlinear function of the state vector given by following:

$$\dot{X}_r = \begin{bmatrix} \dot{r}_{ob} \\ \dot{\theta}_{ob} \\ \dot{\phi}_{ob} \end{bmatrix} = \begin{bmatrix} u\cos\theta_{ob}\cos\phi_{ob} + v\sin\theta_{ob}\cos\phi_{ob} + w\sin\phi_{ob} \\ -\dfrac{\sin\theta_{ob}}{r_{ob}\cos\phi_{ob}}u + \dfrac{\cos\theta_{ob}}{r_{ob}\cos\phi_{ob}}v \\ -\dfrac{\cos\theta_{ob}\sin\phi_{ob}}{r_{ob}}u - \dfrac{\sin\theta_{ob}\sin\phi_{ob}}{r_{ob}}v + \dfrac{\cos\phi_{ob}}{r_{ob}}w \end{bmatrix} \tag{3.24}$$

3.3.2 Measurement Noise Model

Usually the measurement data from vision sensor is inaccurate and noisy. It is assumed that measurement noise is zero mean Gaussian process. Another assumption is that the magnitude of the measurement noise is a function of object range i.e. higher the distance between the sensor and the object, higher will be the measurement uncertainty. It is a reasonable assumption since it is common knowledge that closer you get to an object, higher will be the quality of the visual information obtained. Based on this philosophy, a function of range has been devised to calculate the measurement noise covariance. First we calculate the maximum amount of noise that can be added by following function.

$$r_k = r_0(1-\delta^{r_{ob_k}}) \tag{3.25}$$

here r_k is the maximum percentage noise at time instant k, r_0 is a constant which represents the highest possible amount of percentage measurement noise (as $r_{ob} \to \infty$), r_{ob_k} is the range of the object at time instant k, and δ is a parameter (close to 1) which defines how r_k changes with change in r_{ob_k}. Again based on common knowledge on visual sensing, it is assumed that r_k changes slowly for higher range value and drops quickly for lower values of range. Figure 3.3 shows variation in r_k with range from zero to 1000 meters for $r_0 = 20$ and $\delta = 0.99$.

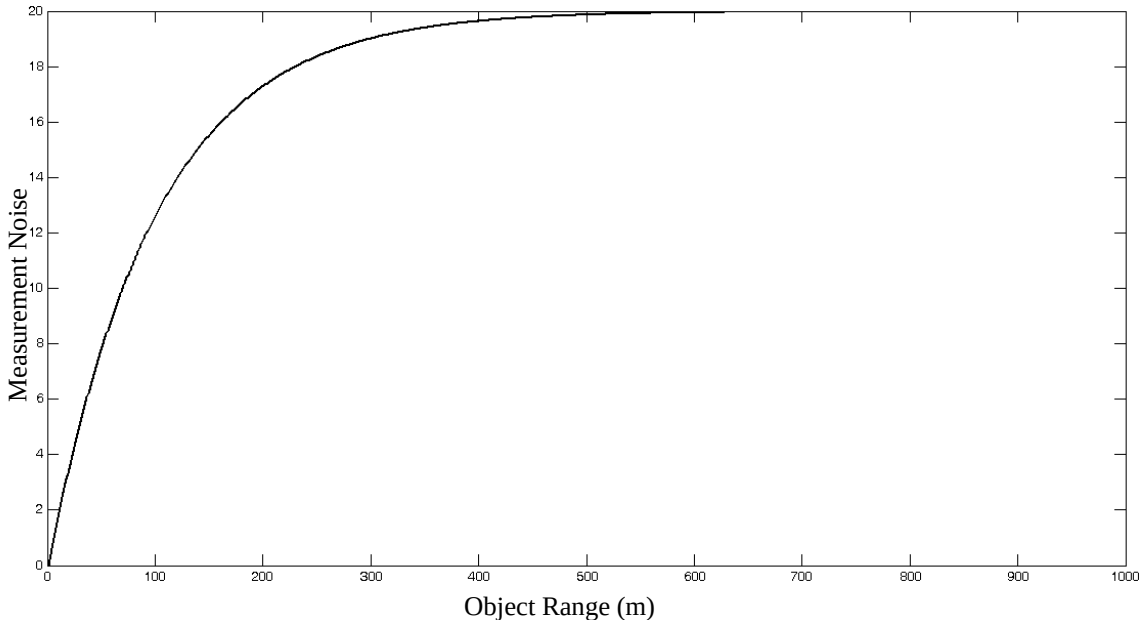

Figure 3.3: Measurement Noise as a function of Object Range

After calculating the maximum percentage noise at time instant k by (3.25), we calculate the measurement noise covariance by following equation:

$$R_k = \left(r_k \frac{W_V}{100} \times \frac{1}{3} \right)^2 \tag{3.26}$$

here W_V represents the angular width of the camera's field of view. Here it is assumed $120°$ on both horizontal and vertical axis. The whole expression inside the bracket represents the standard deviation of the measurement noise. Since noise is normally distributed, above equation insures that measurement noise will be bounded by r_k or 3σ for almost all of the cases (more than 99%).

3.3.3 Process Noise Model

Any real system whether linear or nonlinear has some amount of process noise. Process noise arise from parameters which are not modeled in system dynamics either because their very small effect or because of their mathematical intractability. For example, in our model, the effects of atmosphere such as random wind gusts, variation of gravity with altitude etc are ignored. However, they do have some effect over the system dynamics and hence it is necessary to accommodate them in our system model. The process noise is modeled by adding fictitious noise $w(t)$ in system dynamics. The process noise is

considered as Gaussian random input. The system dynamics given by (3.24) can be rewritten to account for process noise as following:

$$\dot{X}_r = \begin{bmatrix} u\cos\theta_{ob}\cos\phi_{ob} + v\sin\theta_{ob}\cos\phi_{ob} + w\sin\phi_{ob} \\ -\dfrac{\sin\theta_{ob}}{r_{ob}\cos\phi_{ob}}u + \dfrac{\cos\theta_{ob}}{r_{ob}\cos\phi_{ob}}v \\ -\dfrac{\cos\theta_{ob}\sin\phi_{ob}}{r_{ob}}u - \dfrac{\sin\theta_{ob}\sin\phi_{ob}}{r_{ob}}v + \dfrac{\cos\phi_{ob}}{r_{ob}}w \end{bmatrix} + G(t)w(t) \qquad (3.27)$$

here $G(t)$ is the process noise realization matrix and $w(t)$ is zero mean Gaussian noise with covariance given by (3.5). It is a reasonable assumption that process noise covariance $Q(t)$ can be considered constant for a given system over a short period of time, hence can be denoted as just Q. Additionally process noise in each state element is considered to be independent of each other hence process noise realization matrix will be unity i.e. $G(t) = I$.

3.3.4 Initialization of the EKF

Initialization of the EKF requires initial value of the state vector and initial error covariance matrix. However, knowing these two quantities at start is not possible in most of the real systems. That's why it requires some amount of priori information or some empirical knowledge of the system to make an educated guess.

As stated earlier, assuming that UAV is equipped with an image processor capable of identifying all the obstacles present in its view field, EKF is initialized based on the first processed image. The number of obstacles in the first image defines the length of state vector. For example, if image contains two obstacles and target (or waypoint), then the state vector will consist of 9 elements, where each 3 elements will represent the relative position of one of the three objects (whose position needs to be estimated). Similarly in case of one obstacle and target, state vector will contain only 6 elements, as shown in following equations (3.28) and (3.29) respectively:

$$\hat{X}_r(0) = \left[\hat{r}_{ob1}(0) \quad \hat{\theta}_{ob1}(0) \quad \hat{\phi}_{ob1}(0) \quad \hat{r}_{ob2}(0) \quad \hat{\theta}_{ob2}(0) \quad \hat{\phi}_{ob2}(0) \quad \hat{r}_{tr}(0) \quad \hat{\theta}_{tr}(0) \quad \hat{\phi}_{tr}(0)\right]^T \qquad (3.28)$$

$$\hat{X}_r(0) = \begin{bmatrix} \hat{r}_{ob}(0) & \hat{\theta}_{ob}(0) & \hat{\phi}_{ob}(0) & \hat{r}_{tr}(0) & \hat{\theta}_{tr}(0) & \hat{\phi}_{tr}(0) \end{bmatrix}^T \tag{3.29}$$

Since obstacles are assumed to be point obstacles, it is not possible for an obstacle to appear from behind another obstacle. So there is no need to augment the state vector to accommodate newly discovered obstacle while EKF is running. However, it is not a realistic assumption and needs to be relaxed in future studies.

- **Initialization of State Vector:** For simplicity of discussion, we will keep state vector 3 elements long i.e. we will estimate the position of one object only. State vector can be augmented in case of simultaneously estimating the multiple objects. The first image contain the θ and ϕ information (with measurement noise) of the object needs to be estimated. However, 2D vision sensing does not contain any information about the range or r. For initialization purpose, range of the object is assumed known with 50% uncertainty. Following equation shows initialization of state vector.

$$\hat{X}_r(0) = \begin{bmatrix} \hat{r}_{ob}(0) & \hat{\theta}_{ob}(0) & \hat{\phi}_{ob}(0) \end{bmatrix}^T \tag{3.30}$$

  here $\hat{r}_{ob}(0)$ is known as 50% error while $\hat{\theta}_{ob}(0)$ and $\hat{\phi}_{ob}(0)$ are based on first measurement taken and initial measurement noise given by (3.25). Initial estimation of an object requires some amount of priori information (in the form of their range information). The θ and ϕ estimates will be better for the closer objects (because of lower measurement noise) which is consistent with the fact that we will have better visual for the objects which are closer to us.

- **Initialization of Error Covariance Matrix:** Theoretically, the error covariance matrix is given by following:

$$P_0 = E\left[\left(X_r(0) - \hat{X}_r(0) \right) \left(X_r(0) - \hat{X}_r(0) \right)^T \right] \tag{3.10}$$

  here $X_r(0)$ is the actual initial value of the state vector while $\hat{X}_r(0)$ is the initial estimation of the state vector. However, $X_r(0)$ is not known so the initialization of error covariance matrix requires some idea of uncertainty in initially estimated state.

P_0 should be sufficiently high enough to contain maximum amount of error in the initial state estimation. Based on known initial uncertainty in range estimation and initial measurement noise, P_0 is initialized as the following diagonal matrix:

$$P_0 = diag\left(a_1 e^2 \quad a_2 \left(\frac{r_0 \times W_V}{100} \right)^2 \quad a_3 \left(\frac{r_0 \times W_V}{100} \right)^2 \right) \tag{3.31}$$

here e is the maximum range measurement uncertainty (in meters) given by the 50% of the distance between the start point and the destination, r_0 is maximum percentage measurement noise. a_1, a_2 and a_3 are scalar parameters, which needs to be tuned based on trial and error. W_V is camera's width of view (in radians). Initially it is assumed that estimation errors are independent of each other i.e. error in one state element does not affect the others and so on. Note that the dimensions of P matrix will depend on the length of the state vector. For example in case of a state vector with six elements, P_0 matrix will be a 6×6 diagonal square matrix.

3.3.5 Pre-Run of EKF

While applying EKF, it is highly recommended that filter run sufficiently ahead of time so that initial error can be stabilized before its actual application. Since our system is a closed loop system, if error in initial estimation is high, the corresponding control can completely destabilize the whole system. Hence we start EKF 10 seconds before applying any control i.e. UAV fly towards its initial velocity vector and EKF just estimate the relative position of the obstacles and target without applying any guidance. In other words, system behaves like an open loop system during the pre-run of EKF. After that, it reinitializes the $\hat{X}_r(0)$ as the average of all previous estimates as given by (3.32) and P_0 again as (3.31).

$$\hat{X}_r(0) = \frac{1}{N} \sum_{i=1}^{N} \hat{X}_r(i) \tag{3.32}$$

here N is the number of estimations made by EKF during pre-run. After reinitializing the state vector and error covariance matrix, the system transforms into a closed loop

system and starts applying the control based on estimated relative position of the obstacle/s and target.

3.3.6 Propagation of State and Error Covariance

The operation of EKF can be divided into two phases, Propagation and Updation. In propagation phase, EKF predicts the next state based on previous state and known system dynamics i.e. propagation of state from posterior estimate at time instant k to a priori estimate at $k+1$ or mathematically $\hat{X}_{r_k}^{+} \rightarrow \hat{X}_{r_{k+1}}^{-}$. Based on the system dynamics derived in (3.24), the state propagation equation of our system is given by following:

$$\dot{\hat{X}}_r = \begin{bmatrix} u\cos\hat{\theta}_{ob}\cos\hat{\phi}_{ob} + v\sin\hat{\theta}_{ob}\cos\hat{\phi}_{ob} + w\sin\hat{\phi}_{ob} \\ -\frac{\sin\hat{\theta}_{ob}}{\hat{r}_{ob}\cos\hat{\phi}_{ob}}u + \frac{\cos\hat{\theta}_{ob}}{\hat{r}_{ob}\cos\hat{\phi}_{ob}}v \\ -\frac{\cos\hat{\theta}_{ob}\sin\hat{\phi}_{ob}}{\hat{r}_{ob}}u - \frac{\sin\hat{\theta}_{ob}\sin\hat{\phi}_{ob}}{\hat{r}_{ob}}v + \frac{\cos\hat{\phi}_{ob}}{\hat{r}_{ob}}w \end{bmatrix} \tag{3.33}$$

here $V = [u \quad v \quad w]^T$ is present relative velocity vector between the UAV and the object being estimated. Since in this study only stationary obstacles are considered, vector $V = -V_{UAV}$ i.e. negative of UAV velocity vector V_{UAV} and $\hat{X}_r = [\hat{r}_{ob} \quad \hat{\theta}_{ob} \quad \hat{\phi}_{ob}]^T$ is previous estimate of the relative position of the object being estimated.

After propagating the state vector, we propagate the error covariance matrix or the P matrix. The P matrix propagation equation is given in section 3.2 EKF summary.

$$\dot{P}(t) = A(t)P(t) + P(t)A^T(t) + G(t)Q(t)G^T(t) \tag{3.15}$$

Since $G(t) = I$ (from section 3.3.3) and $Q(t)$ is constant (can be denoted as just Q), above equation can be modified as following equation:

$$\dot{P}(t) = A(t)P(t) + P(t)A^T(t) + Q \tag{3.34}$$

here $A(t) = \left.\frac{\partial f}{\partial X_r}\right|_{\hat{X}_r(t)}$ i.e. partial differential of the nonlinear state equation with respect to the state vector evaluated at a priori estimate of the state vector. The expression for the

matrix $A(t)$ is derived next in this section. After partially differentiating $\dot{X}_r$ (System dynamics given by (3.24)) with respect to state vector X_r, the matrix A is given by following expression.

$$A = \begin{bmatrix} 0 & \cos\phi(-u\sin\theta + v\cos\theta) & -u\cos\theta\sin\phi - v\sin\theta\sin\phi + w\cos\phi \\ \dfrac{u\sin\theta - v\cos\theta}{r^2\cos\phi} & \dfrac{u\cos\theta + v\sin\theta}{-r\cos\phi} & \dfrac{\sin\phi(-u\sin\theta + v\cos\theta)}{r\cos^2\phi} \\ \dfrac{u\cos\theta\sin\phi + v\sin\theta\sin\phi - w\cos\phi}{r^2} & \dfrac{\sin\phi(u\sin\theta - v\cos\theta)}{r} & \dfrac{-u\cos\theta\cos\phi - v\sin\theta\cos\phi - w\sin\phi}{r} \end{bmatrix} \tag{3.35}$$

3.3.7 Updation of State and Error Covariance

In Updation phase of operation, EKF updates or corrects the previously predicted state and error covariance based on the newly arrived measurements. After getting the measurements according to the following equation

$$Y_k = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X_{r_k} + v_k \tag{3.36}$$

here v_k is the zero mean Gaussian noise with covariance given by (3.14). Next step before Updation is to compute the Kalman Gain by following:

$$K_k = P_k^- C_k^T \left[C_k P_k^- C_k^T + R_k \right]^{-1} \tag{3.37}$$

here

$$C_k = \left[\frac{\partial h}{\partial Xr} \right] \Bigg|_{\hat{X}_{r_k}^-} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.38}$$

and R_k is measurement noise covariance given by measurement noise model described in section 3.3.2. After calculating the Kalman gain, next is update the state estimate and error covariance matrix based on following equations:

$$\hat{X}_{r_k}^+ = \hat{X}_{r_k}^- + K_k \left[Y_k - h(\hat{X}_{r_k}^-) \right] \tag{3.39}$$

$$P_k^+ = (I - K_k C_k) P_k^- (I - K_k C_k)^T + K_k R_k K_k^T \tag{3.40}$$

As seen above, the Kalman gain is a very important parameter in the EKF Updation phase. Kalman gain defines where to pay more attention while updating the state estimate between measurements and system dynamics. From above equations it is clear that higher

the Kalman gain, higher will be the effect of the measurements over the state Updation and vice versa.

3.3.8 Smoothing of Estimate

Due to measurement uncertainties, sometimes due to momentarily high noise position estimates tend to fluctuate i.e. newly estimated object position differs too much from previous estimates. Since it is a closed loop system, these fluctuations can produce large associative control accelerations and which can severely destabilize the whole system.

To avoid that, it is better to smooth the new estimate with respect to the previous estimates i.e. instead of using the current estimate only for guidance purpose, first take the average of current estimate with few previous estimates and then apply the guidance according to the averaged or "smoothed" estimate. Deciding how many previous estimates used for smoothing operation should be done carefully. It is a tradeoff between trusting a newly arrived estimate and trusting previous estimates. If we use too many previous estimates, the effect of any new estimate will almost zero. Similarly, too few previous estimates will not affect a large estimation fluctuation at all. Since we are using a range dependent measurement noise model, the measurements are improving as UAV is getting closer to the object. Thus more emphasis should be given to the newly arrived estimates. Based on this philosophy, after some trial runs it is found out that algorithm works best while smoothing operation is performed with ten previous estimates. The smoothing operation performed by following equation. Note that smoothing is only required after the pre-run of EKF is over and system is working as a closed loop system.

$$\hat{\bar{X}}_{r_k} = \frac{1}{n} \sum_{i=k-n-1}^{k} \hat{X}_{r_i} \tag{3.41}$$

here n is the number of previous estimates used for smoothing operation ($n = 10$ in our case), $\hat{\bar{X}}_{r_k}$ is smoothed estimate and $\hat{X}_{r_k}$ original estimate at time instant k.

3.3.9 Tuning of EKF

After developing the whole EKF, its tuning is the final step. Tuning of EKF requires proper selection of parameters Q, P_0 and R_k. As stated earlier, EKF is fragile in nature i.e. it works well only for a narrow band of Q, P_0 and R parameters. Hence tuning of EKF should be done carefully.

Since we are using a range dependent measurement noise model, parameter R_k is fixed by measurement noise model given by (3.26). In case of one obstacle and target estimation, R_k is given by following formula:

$$R_k = diag\left(\left(r_{ob_k}\frac{W_V}{100}\times\frac{1}{3}\right)^2 \quad \left(r_{ob_k}\frac{W_V}{100}\times\frac{1}{3}\right)^2 \quad \left(r_{tr_k}\frac{W_V}{100}\times\frac{1}{3}\right)^2 \quad \left(r_{tr_k}\frac{W_V}{100}\times\frac{1}{3}\right)^2\right) \tag{3.42}$$

here W_V is the angular width of the camera's view and assumed to be $\frac{2}{3}\pi$ radians (120°) for both vertical and horizontal axis. r_{ob_k} and r_{tr_k} are the maximum percentage noise in measurement of angles for obstacle and target respectively. They are calculated by (3.25) given in section 3.3.2 based on range dependent measurement noise model.

$P(0)$ is selected by some prior information about error in initial estimation of state. This knowledge can be empirical or can be an educated guess. It is given by following equation (3.43) for simultaneous estimation of an obstacle with target:

$$P_0 = diag\left(a_1 e_{ob}^2 \quad a_2\left(\frac{r_{ob_0}\times W_V}{100}\right)^2 \quad a_3\left(\frac{r_{ob_0}\times W_V}{100}\right)^2 \quad a_4 e_{tr}^2 \quad a_5\left(\frac{r_{tr_0}\times W_V}{100}\right)^2 \quad a_6\left(\frac{r_{tr_0}\times W_V}{100}\right)^2\right) \tag{3.43}$$

here e_{ob} and e_{tr} are the maximum range estimation uncertainty for obstacle and target respectively (in meters given by the 50% of the distance between the start point and the destination) and assumed known. a_1, a_2, a_3, a_4, a_5 and a_6 are scalar parameters, which are tuned to following values with trial and error method.

$$\begin{aligned} a_1 &= 1 \quad a_4 = 1 \\ a_2 &= 2 \quad a_5 = 2 \\ a_3 &= 2 \quad a_6 = 2 \end{aligned} \tag{3.44}$$

The process noise covariance Q set to:

$$Q = diag\begin{pmatrix} 0.2 & 0.025 & 0.025 & 0.2 & 0.025 & 0.025 \end{pmatrix} \tag{3.45}$$

here the diagonal elements of Q matrix are selected through trial and error method. First and fourth diagonal elements of Q matrix represent the process noise covariance (in meter) for the range elements of the state vector shown in equation (3.29). Similarly second, third, fifth and last diagonal elements represent the process noise covariance for angle elements (in radians) of state vector.

Chapter 4

Guidance and Navigation

Once the obstacle position is estimated, the objective reduces to applying the guidance to navigate the UAV around it. Now first task is to finding out whether obstacle is critical i.e. if collision with obstacle is imminent. For that Collision Cone approach is applied. This technique detects collisions and computes an alternate aiming point (if necessary). Then the geometry of the resulting guidance problem is analyzed. These steps aid in forming the guidance objective. Finally two nonlinear guidance laws used to achieve the guidance objective, Nonlinear Geometric Guidance (NGG) and Differential Geometric Guidance (DGG) are explained.

4.1 The Collision Detection and Aiming Point Computation

The UAV must detect an imminent collision and avoid it safely. The “collision cone" [3] is an effective tool for:

(i) Detecting collision

(ii) Finding an alternate direction of motion that will avoid the collision

In this approach, a collision cone is constructed and analyzed for every obstacle. The collision cone approach is used to find a safe aiming point X_{ap} and the time-to-go to the aiming point t_{go}. A suitable guidance law should then be used to steer the UAV to an aiming point X_{ap} in time t_{go}. The construction of the collision cone is shown in Figure 4.1. A spherical safety boundary of radius d is constructed around the obstacle. An obstacle is considered to be *critical* if the UAV is expected to violate the safety boundary in future. X_r is the relative distance between the UAV and the obstacle and V is the total velocity. Since the collision cone approach operates in two dimensions, the plane containing X_r and V is considered for constructing the cone. The safety boundary thus reduces to a circle β in this plane. A collision cone is constructed by dropping tangents from the UAV to the circle β. If the velocity vector V lies within the collision cone, the

UAV will violate β in due course and result in collision. Thus the obstacle is said to be critical. From Figure 4.1, V can be expressed in terms of the tangents r_1 and r_2 as follows:

$$V = ar_1 + br_2 \tag{4.1}$$

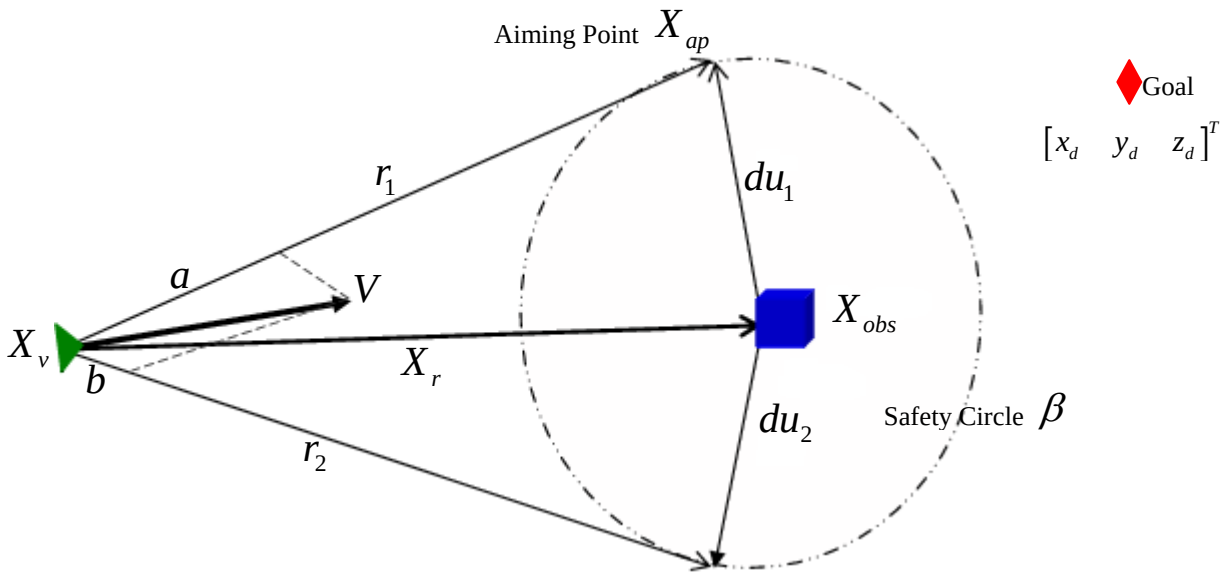

Figure 4.1: Construction and analysis of the collision cone

The collision criterion is stated as:

If $a > 0$ AND $b > 0$, the obstacle under consideration is said to be critical

The aiming point is then found from the collision cone. First, the tangents r_1 and r_2 are found as follows:

$$\begin{aligned} r_1 &= X_r + du_1 \\ r_2 &= X_r + du_2 \end{aligned} \tag{4.2}$$

where u_1 and u_2 are the unit vectors perpendicular to the tangents. The aiming point is determined in the following way:

$$\begin{aligned} if\ a > b, X_{ap} &= X_v + r_1 \\ f\ b > a, X_{ap} &= X_v + r_2 \end{aligned} \tag{4.3}$$

Since the velocity in x direction is assumed constant, the time-to-go t_{go} is found as follows:

$$t_{go} = \frac{1}{u}[(X_{ap})_x - (X_v)_x] \tag{4.4}$$

The expressions for a, b and X_{ap} are derived in Appendix A. Two practical issues are expected to arise in the implementation of the collision cone approach. We list these issues and describe the solution methodology for each.

- **Obstacle safety bound violation after aiming point is reached:** An obstacle may be considered critical only so long as the vehicle is behind it (i.e., the x-coordinate of the vehicle is less than the x-coordinate of the obstacle). Once the UAV is past an aiming point, it immediately looks to maneuver towards the next aiming point. This may result in a brief violation of the first obstacle's safety bound if the direction of the new aiming point lies in the opposite side of the safety sphere. Such a scenario is illustrated in Figure 4.2. In order to remedy this, a *sphere-tracking algorithm* is activated when $X_r < d$. The sphere tracking algorithm computes a new aiming point, called the *virtual aiming point* which is a point on the surface of the safety sphere. This is found by radially extending the original relative distance line X_r until it meets the surface of the safety sphere. The UAV then aims for the virtual aiming point until $X_r > d$. The mathematical details of the sphere tracking algorithm may be found in Appendix B.
- **Intersecting obstacle safety boundaries:** Figure 4.3 illustrates the problem of safety boundary intersection. It is apparent from the vehicle's orientation that obstacle-1 is critical, and that the UAV must aim towards X_{p2} (X_{p1} and X_{p2} are the two choices for the aiming point calculated from the collision cone approach). However X_{p2} is clearly illegal as it lies within the safety circle of obstacle-2. The algorithm must therefore be able to identify this issue and maneuver the UAV to X_{p1}. We solve this by determining the center *P*2 of the intersecting area of the two circles [26].

$$P_2 = P_0 + m\left(\frac{P_1 - P_0}{l}\right) \tag{4.5}$$

where $l = \| P_1 - P_0 \|$ and $m = \dfrac{l^2 + r_0^2 - r_1^2}{2l}$. We then choose the aiming point that is farther away from P_2, i.e.,

$$\begin{aligned} &if \ \| X_{p1} - P_2 \| \geq \| X_{p2} - P_2 \|, then \ \ X_{ap} = X_{p1} \\ &if \ \| X_{p1} - P_2 \| \leq \| X_{p2} - P_2 \|, then \ \ X_{ap} = X_{p2} \end{aligned} \tag{4.6}$$

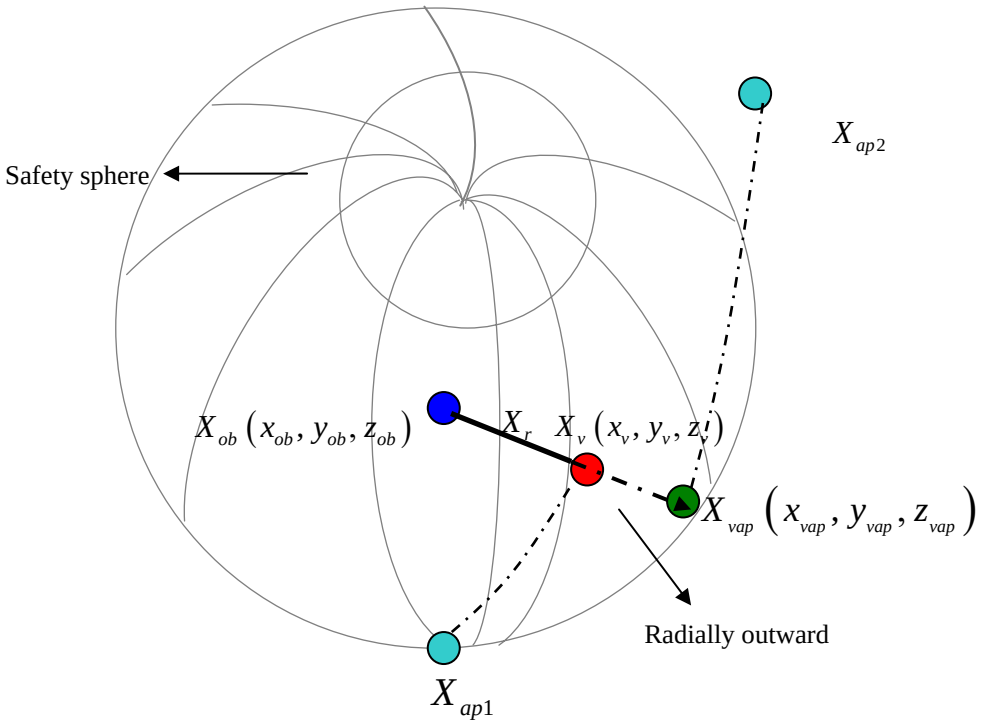

Figure 4.2: Safety boundary violation after aiming point is reached

The collision cone approach constitutes an effective tool for detecting impending collisions and finding an aiming point. The collision avoidance problem then becomes one of guiding the UAV from $X_v(t_0) = X_{in}$ to $X_v(t_0 + t_{go}) = X_{ap}$. Note that when no obstacles are critical $X_{ap} = X_d$. The collision avoidance problem therefore becomes similar to a sequential target interception problem.

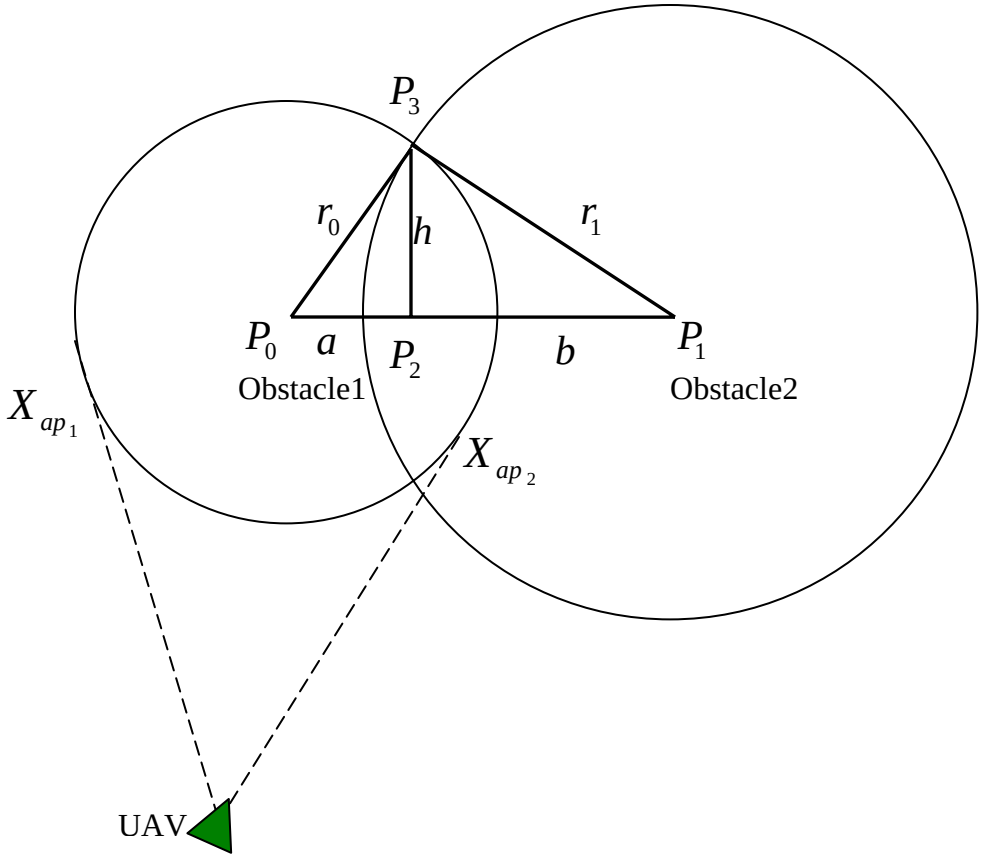

Figure 4.3: Intersection of safety boundaries of two obstacles

4.2 Geometry of the Guidance Problem

The guidance objective is to align the velocity vector to the aiming line so that collision is avoided. The aiming line is the line joining the aiming point X_{ap} to the center of gravity of the vehicle, in the case of stationary obstacles. The problem geometry is shown in Figure 4.4. u, v and w are the velocity components along the *x*, *y* and *z* directions respectively, and $a_y = \dot{v}$, $a_z = \dot{w}$ are the accelerations applied in the *y* and *z* directions respectively. Note that the velocity u is assumed to be a constant.

The angle θ between the total velocity vector and the aiming line is to be eliminated. In order to formulate a guidance which computes the controls a_y and a_z that would eliminate θ, we consider the 3D problem as a combination of two separate 2D problems in the XY and XZ planes. $(X_{ap})_{XY}$ and $(X_{ap})_{XZ}$ are the projections of the aiming line on

to the XY and XZ planes respectively, while V_{XY} and V_{XZ} are projections of the velocity vector on to the XY and XZ planes respectively. The orthogonal projection V_P of a vector V onto a plane with basis vectors X_1 and X_2 is [27]

$$V_p = c_1 X_1 + c_2 X_2 \tag{4.7}$$

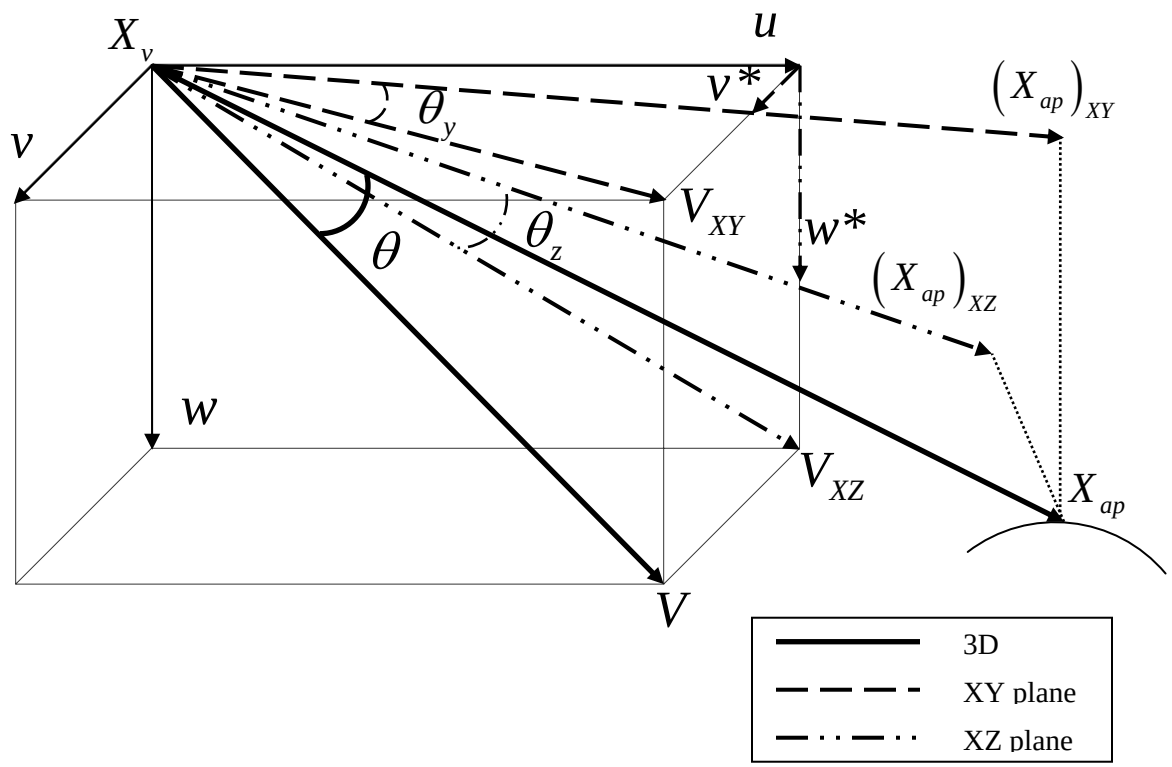

Figure 4.4: Geometry of the guidance problem in 3D

c_1 and c_2 are found as follows:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = [(X^T X)^{-1} X^T] V \tag{4.8}$$

where $X = \begin{bmatrix} X_1 & X_2 \end{bmatrix}^T$. The basis vectors in the problem under consideration are the unit vectors along the X, Y and Z axes i.e.

$$X = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, Y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, Z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.9}$$

θ_y is the angle between $(X_{ap})_{XY}$ and V_{XY} and θ_z is the angle between $(X_{ap})_{XZ}$ and V_{XZ}. The guidance objective is restated as: compute the controls a_y and a_z to eliminate θ_y and θ_z respectively in the two independent XY and XZ planes.

4.3 2D Decomposition of the 3D Guidance Problem

Figure 4.5 shows the geometry of the guidance problem in the XY plane. Due to symmetry, the geometry of the problem in XZ plane is the same.

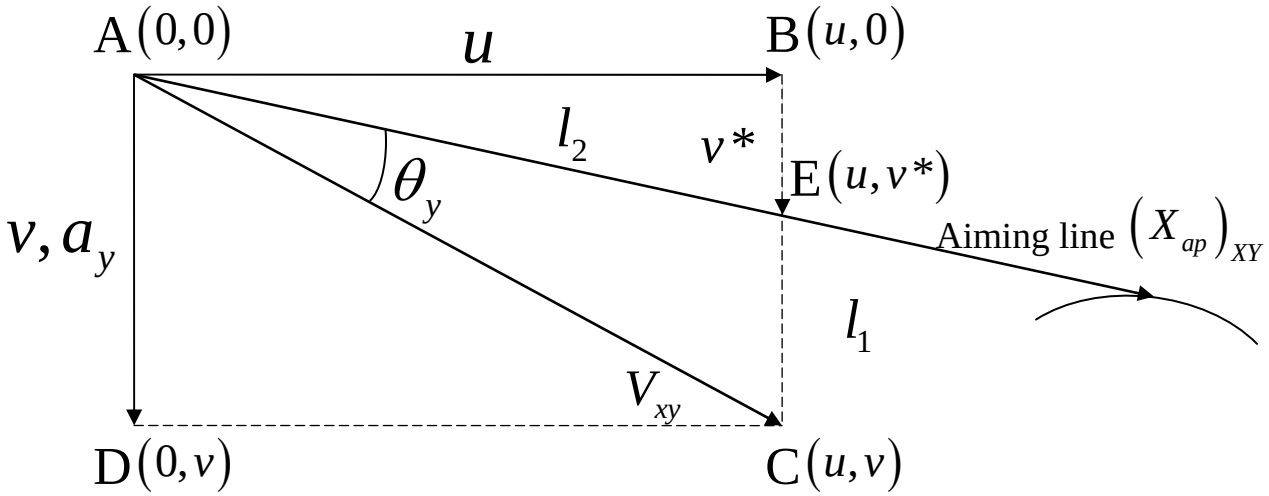

Figure 4.5: Geometry of the guidance problem in 2D

When the velocity vector V_{XY} is perfectly aligned with the aiming line X_{ap}, the *y*-component of the velocity vector changes to a new value, v^* (w^* in the XZ plane). Note that in keeping with the kinematics of the UAV, the *x*-velocity u remains constant. v^* is found by the intersection of the lines l_1 and l_2. The points $B(u,0)$, $E(u,v^*)$ and $C(u,v)$ lie on the line l_1 i.e.,

$$l_1 : x = u \tag{4.10}$$

The points $A(0,0)$, $E(u,v^*)$ and $X_{ap}(x_{ap},y_{ap})$ lie on the line l_2. The equation of l_2 using the two-point form of equation of the line is:

$$l_2 : \frac{y}{y_{ap}} = \frac{x}{x_{ap}} \tag{4.11}$$

The point E is the intersection of the lines l_1 and l_2. We substitute (4.10) into (4.11) in order to find v^*, the y-coordinate of E

$$v^* = \left(\frac{y_{ap}}{x_{ap}} \right) u \tag{4.12}$$

Since u is considered to be a constant and X_{ap} is a constant point, $\dot{v}^*$ at each instant.

4.4 Guidance Strategy

This section describes two guidance strategies used to navigate the UAV in order to avoid collision. The two new guidance strategies are Differential Geometric Guidance (DGG) and Nonlinear Geometric Guidance (NGG) proposed in [4].

4.4.1 Nonlinear Geometric Guidance

The Nonlinear Geometric Guidance (NGG) law is as follows:

$$\begin{bmatrix} a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \hat{k}_v \sin\theta_y \\ \hat{k}_w \sin\theta_z \end{bmatrix} \tag{4.13}$$

The control is thus a nonlinear function of the aiming angle θ. An advantage that immediately presents itself is that the range of the sine function is $[-1,1]$ whereas the range of θ is $[-\infty,\infty]$. This indicates that the acceleration in NGG is always bounded, provided $\hat{k}_v$ is bounded.

The gains $\hat{k}_v$ and $\hat{k}_w$ can be selected as constants with some tuning. However, it is not a good idea to have constant gains, because when t_{go} is low, high control needs to be generated and vice-versa. Hence control gains are set inversely proportional to t_{go} as given by following equation:

$$\hat{k}_v = \frac{a}{\alpha t_{go}}$$

$$\hat{k}_w = \frac{b}{\beta t_{go}}$$

here a, b, α and β are tuning parameters which are set to following values through trial and error.

$$a = 4 \quad \alpha = 0.6$$
$$b = 4 \quad \beta = 0.6$$

4.4.2 Nonlinear Differential Geometric Guidance

In order to achieve alignment of the velocity vector with the aiming line, the control a_y should be designed so that the y-velocity v approaches v^* in time t_{go}. The nonlinear Differential Geometric Guidance (DGG) is based on Dynamic Inversion (DI) [28, 29], a control strategy used for output tracking of nonlinear systems. The principle of DI is to drive a stabilizing error dynamics (chosen by the designer) to zero. The main advantage of DI is that it essentially guarantees global asymptotic stability with respect to the tracking error.

We now describe the DI guidance design. Let the error be

$$e_v = v - v^* \tag{4.14}$$

Imposing the first order error dynamics

$$\dot{e}_v + k_v e_v = 0 \tag{4.15}$$

i.e.

$$(\dot{v} - \dot{v}^*) + k_v (v - v^*) = 0 \tag{4.16}$$

With quasi-steady approximation (i.e. assuming u is a constant at every instant of time), v^* is a constant. In addition $\dot{v} = a_y$, from the system dynamics. The DI based guidance law is hence derived to be:

$$a_y = -k_v (v - v^*) \tag{4.17}$$

We call the expression of a_y in (4.17) the "Dynamic Inversion Guidance" law, or the "Nonlinear Differential Geometric Guidance" (DGG) law, since the guidance strategy is derived based on the derivative of the error. The constant k_v is designed such as the settling time (i.e. the time taken to align the velocity vector with the aiming line) is inversely proportional to the t_{go}, i.e. $k_v = \frac{1}{\tau_v}$ where τ_v is the desired time constant of the

error dynamics. Note that τ_v can be selected by choosing an appropriate settling time T_{Sv} , since for linear systems theory, $\tau_v = \frac{T_{Sv}}{4}$.

Furthermore one can choose T_{Sv} as $T_{Sv} = \alpha t_{go}$, $0 < \alpha < 1$. Such a guidance strategy ensures that a larger control is generated for an obstacle that is nearer (i.e. the t_{go} is smaller). The guidance strategy in (4.17) is proportional to the error in the y-velocity, and thus produces a large control input at the beginning which effects quick settlement along the aiming line. Two factors influence the control a_y:

- t_{go}: The gain k_v is designed to be inversely proportional to t_{go} so that larger control is demanded for obstacles that are closer.
- α : The choice of α determines the speed of settling of the control. If the value of α is close to 1, the settling is slow and consequently, the peak in control is low. However if fast settling is desired, a high value of α will effect this. However faster settling results in a higher peak in control.

If both the t_{go} and α are small, the peak in control may become unfeasibly large and result in control saturation. The value of α must therefore be chosen judiciously based on the requirement of speed of alignment. Since reactive collision avoidance requires quick maneuvering, a value of α between 0.3 and 0.8 are suitable. Slower maneuvers with $\alpha > 0.8$ may be suitable for flying to the destination.

In an analogous manner, the expression for a_z in XZ plane can be derived as

$$a_z = -k_w(w - w^*) \tag{4.18}$$

$$k_w = \frac{1}{\tau_w} \tag{4.19}$$

and

$$T_{Sw} = 4\tau_w \tag{4.20}$$

where

$$T_{Sw} = \beta t_{go} \tag{4.21}$$

We examine the stability of the DI-based guidance strategy with respect to the error. The error is defined as:

$$\begin{bmatrix} e_v \\ e_w \end{bmatrix} = \begin{bmatrix} v - v^* \\ w - w^* \end{bmatrix} \tag{4.22}$$

The error dynamics are described as follows:

$$\begin{bmatrix} \dot{e}_v \\ \dot{e}_w \end{bmatrix} = \begin{bmatrix} -k_v & 0 \\ 0 & -k_w \end{bmatrix} \begin{bmatrix} e_v \\ e_w \end{bmatrix} \tag{4.23}$$

Equation (4.23) constitutes a linear time-varying system. Consider a linear time-varying system of the form $\dot{x} = A(t)x$. The system is determined to be asymptotically stable if the symmetric matrix $A(t) + A^T(t)$ has all eigen values lying the left half of the complex plane [30]. In the system in (4.23), the eigen values of $A(t) + A^T(t)$ are $-2k_v$ and $-2k_w$. Since k_v and k_w are inversely proportional to the t_{go}, they are strictly positive. Therefore both the eigen values lie strictly on the left-half of the complex plane, and the system in (4.23) is asymptotically stable.

4.4.3 Correlation of DGG and NGG

The DGG law is equivalent to the NGG law, if control gains k_v and k_w are set as given by following formula.

$$k_v = \hat{k}_v \left(\frac{u}{\sqrt{u^2 + v^2}\sqrt{u^2 + (v^*)^2}} \right) \tag{5.6}$$

$$k_w = \hat{k}_w \left(\frac{u}{\sqrt{u^2 + w^2}\sqrt{u^2 + (w^*)^2}} \right) \tag{5.7}$$

here $V = \begin{bmatrix} u & v & w \end{bmatrix}^T$ is the UAV velocity vector. v^* and w^* are desired Y velocity and Z velocity respectively.

With above gain settings for DGG, the controls generated by it will be exactly same as controls generated by NGG law. Thus both guidance strategies are directly correlated. The NGG will therefore possess all the advantages of DGG discussed in previous section.

Chapter 5

Results

The EKF based estimation technique developed in this study is tested in a number of numerical experiments for two scenarios i.e. Single Obstacle with Target Estimation and Two Obstacles with Target Estimation in 3D separately for each of the guidance strategy. Additionally to check the consistency of the EKF, the Sigma-bound test was performed. The results of Sigma-bound test are also presented in this Chapter.

5.1 Success Criterions

The success of the algorithm was tested on three criterions:

- Violation of the safety sphere
- Divergence from the safety sphere
- UAV's Target Miss Distance

Important thing to note here is that while our primary objective is to avoid the obstacle, at the same time UAV should not diverge too much from its path in the process. If the estimate of the obstacle position is good then UAV's closest approach with the obstacle should be roughly equal to the radius of the safety sphere, since obstacles appear almost at the direct path between start point and destination. Both, too much violation of safety sphere and too much divergence from it, indicate that obstacle position estimates were not good enough.

The final success criterion represents how close UAV gets to the target. To calculate the UAV's target miss-distance, a second order curve was fitted among closest approach point, one before it and one after it. Following equations show how target miss-distance was calculated.

$$\begin{aligned} R(t_f - \Delta t) &= a + b\,(t_f - \Delta t) + c\,(t_f - \Delta t)^2 \\ R(t_f) &= a + b\,(t_f) + c\,(t_f)^2 \\ R(t_f + \Delta t) &= a + b\,(t_f + \Delta t) + c\,(t_f + \Delta t)^2 \end{aligned} \tag{5.1}$$

here $R(t)$ is the distance of the UAV from the target at time t. a, b and c are coefficients of a second order function and calculated by following equations.

$$\begin{bmatrix} R(t_f - \Delta t) \\ R(t_f) \\ R(t_f + \Delta t) \end{bmatrix} = \begin{bmatrix} 1 & (t_f - \Delta t) & (t_f - \Delta t)^2 \\ 1 & (t_f) & (t_f)^2 \\ 1 & (t_f + \Delta t) & (t_f + \Delta t)^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{5.2}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 & (t_f - \Delta t) & (t_f - \Delta t)^2 \\ 1 & (t_f) & (t_f)^2 \\ 1 & (t_f + \Delta t) & (t_f + \Delta t)^2 \end{bmatrix}^{-1} \begin{bmatrix} R(t_f - \Delta t) \\ R(t_f) \\ R(t_f + \Delta t) \end{bmatrix} \tag{5.3}$$

After computing a, b and c, the time of UAV's closest approach $t_{\min}$ calculated by (5.4) and then target miss-distance is calculated using (5.5).

$$t_{\min} = -\frac{b}{2c} \tag{5.4}$$

$$R_{\min}(t_{\min}) = a + b\,(t_{\min}) + c\,(t_{\min})^2 \tag{5.5}$$

Based on these success criterions, different segments of success are created defined by the band of the closest approach of the UAV with obstacles and target. These segments of success are named as S-1, S-2, S-3 and S-4 where each increment represents slightly relaxed success conditions i.e. width of the tolerable closest approach band is increased so S-1 represents the strictest conditions while S-4 represents most relaxed case. These conditions are described in Table I.

TABLE I: Different Success Bands

Success band	**Tolerable Safety Sphere Violation (as % of the Safety Sphere Radius)**	**Tolerable Divergence from Safety Sphere (as % of the Safety Sphere Radius)**	**UAV's Target Miss Distance (m)**
S-1	10%	10%	< 10
S-2	20%	20%	< 10
S-3	30%	30%	< 10
S-4	40%	40%	< 10

5.2 Single Obstacle with Target Estimation

The experiments with a single stationary obstacle involve a finite space with one point obstacles with a known safety sphere radius. The position of the obstacle is randomly chosen in each simulation run making sure it obstructs the path of the UAV. The initial velocity of the UAV is also chosen randomly uniformly distributed between following limit (in meters per second).

$$\begin{aligned} 5 \le u \le 20 \\ -5 \le v \le 5 \\ -5 \le w \le 5 \end{aligned} \tag{5.6}$$

For first 10 seconds of the simulation, no control is applied and EKF estimates the obstacle and target positions in an open loop system (pre-run). After the pre-run, EKF reinitializes the state vector and error covariance and then system applies one of the two navigation law in order to find a collision free path. Figure 5.1 shows an example scenario of 3D space where simulations are being conducted.

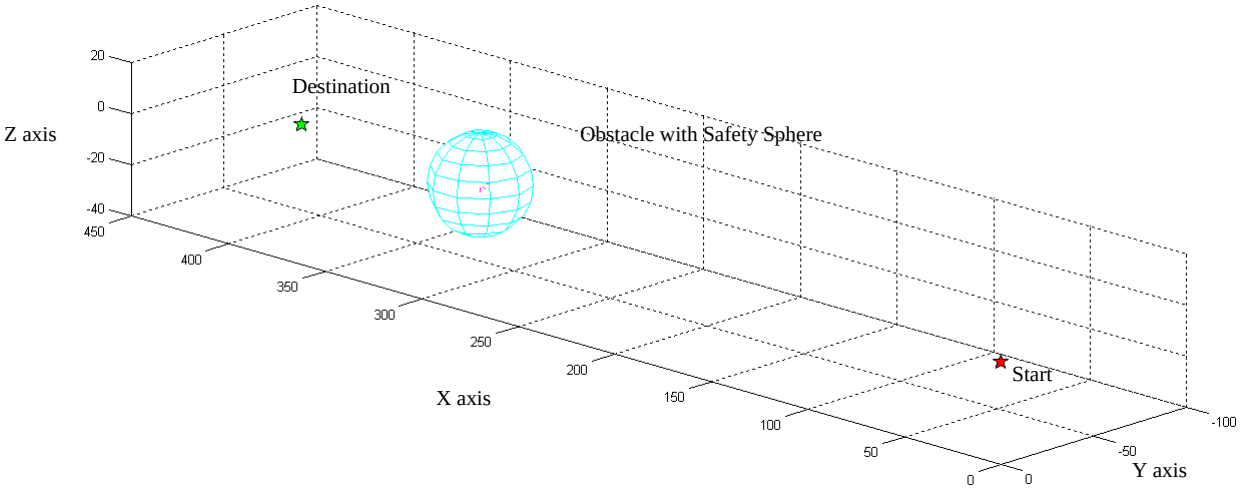

Figure 5.1 Finite 3D Space with One Obstacle

5.1.1 DGG Navigation Strategy

In this part of the experiments, DGG navigation law was applied in order to find a collision free path. A total of 1000 simulation runs performed in order to test the effectiveness of the DGG navigation law while estimating the obstacle and target position with EKF. Based on the success criterion described in section 5.1, following Table II shows the percentage of successes.

TABLE II: Success Percentage with DGG Navigation

Success band	% Run Satisfying the Success Conditions
S-1	51.3
S-2	73.8
S-3	84.1
S-4	90.5

The Figure 5.2 shows the UAVs closest approach with obstacle as the Percentage of the safety sphere radius. In Figure 5.2 different color lines represent the different success bands. The Figure 5.3 shows the final distance of UAV from the destination. In this Figures, Black dots represent the successful case while Red dots represent the failures.

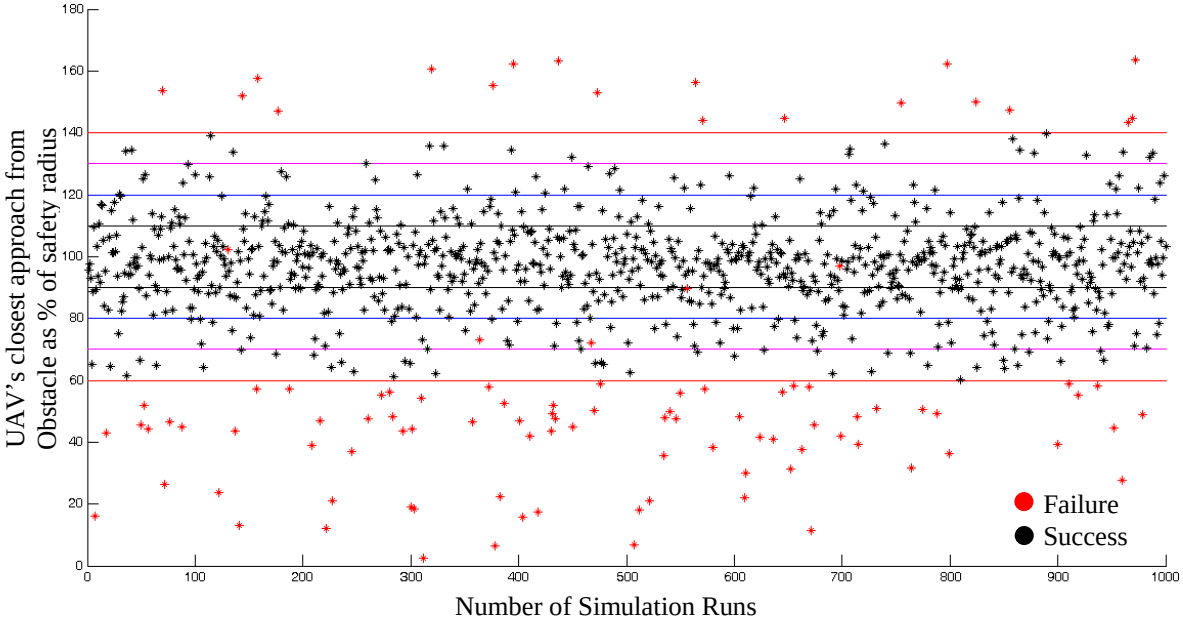

Figure 5.2: UAV's Closest Approach to Obstacle with DGG Navigation

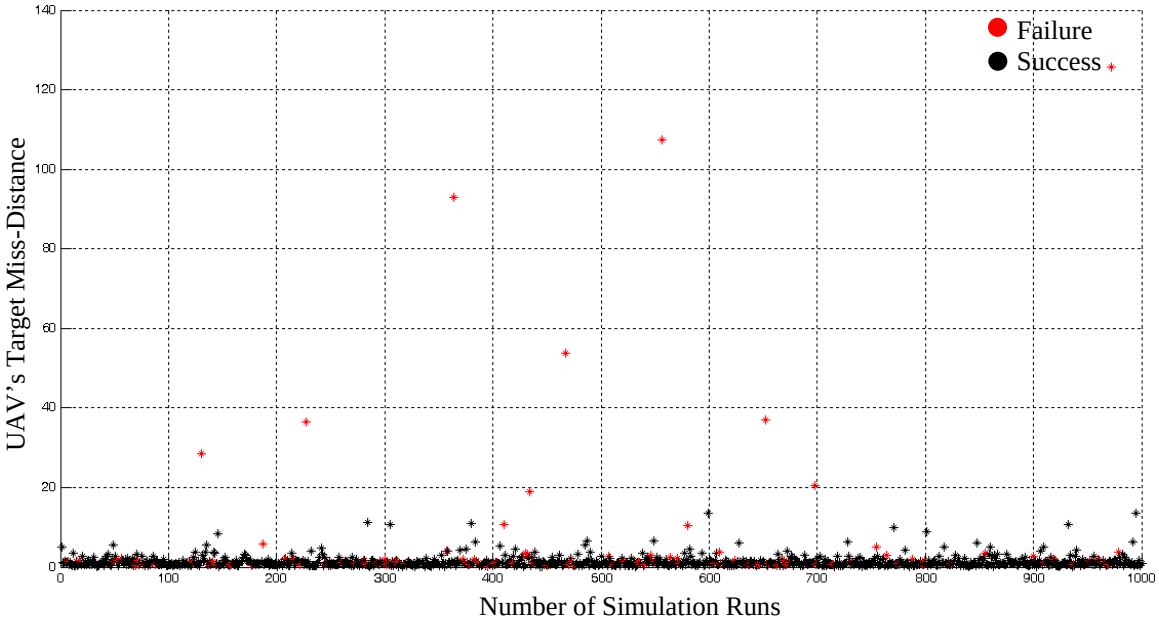

Figure 5.3: UAV's Target Miss-Distance with DGG Navigation

5.1.2 NGG Navigation Strategy

Here NGG navigation law was applied in order to find a collision free path. Again a total of 1000 simulation runs performed in order to test the effectiveness of the NGG navigation law while estimating the obstacle and target position with EKF. Based on the success criterion described in section 5.1, following Table III shows the Percentage of successes.

TABLE III: Success Percentage with NGG Navigation

Success band	% Run Satisfying the Success Conditions
S-1	50.1
S-2	72.5
S-3	84.1
S-4	91.2

The Figure 5.4 shows the UAVs closest approach with obstacle as the Percentage of the safety sphere radius. The Figure 5.5 shows the final distance of UAV from the destination.

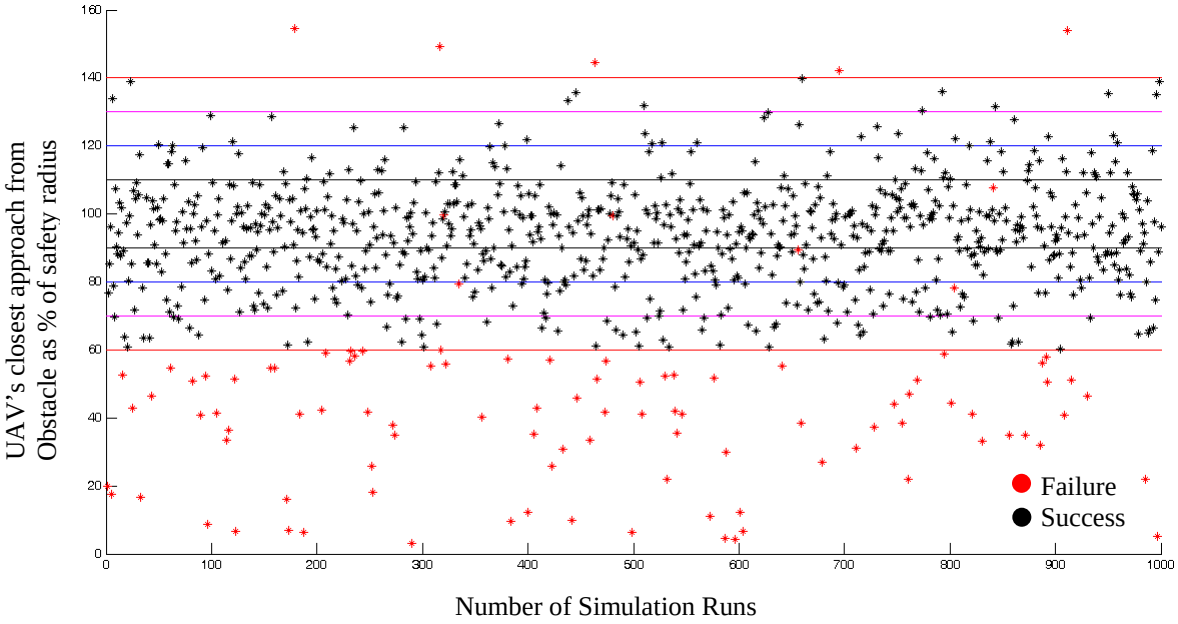

Figure 5.4: UAV's Closest Approach to Obstacle with NGG Navigation

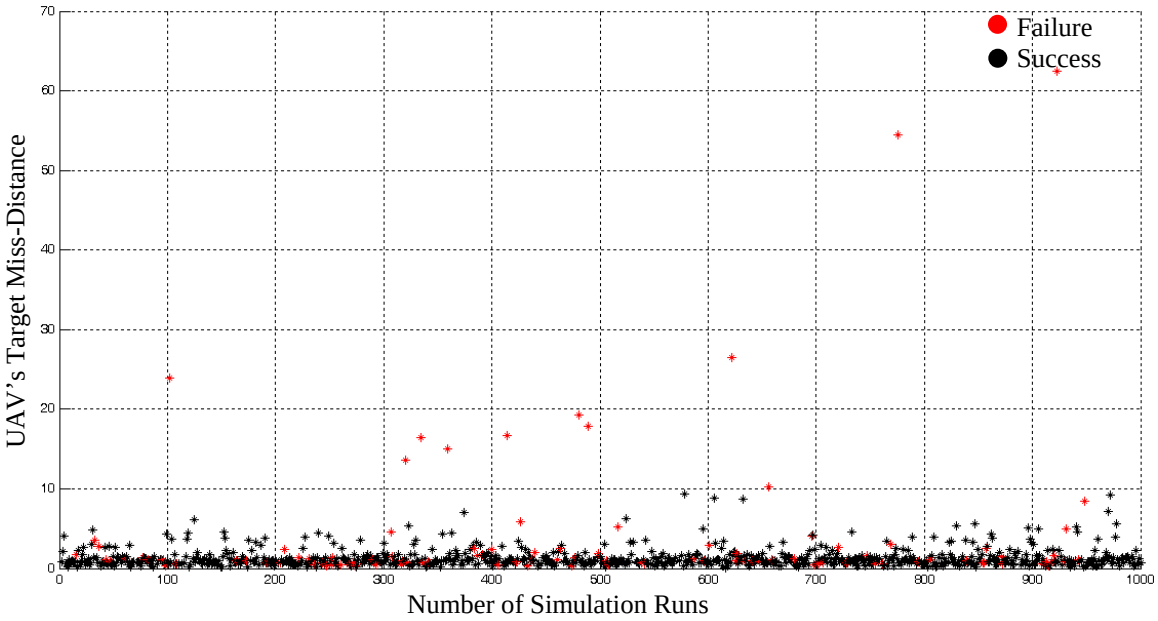

Figure 5.5: UAV's Target Miss Distance with NGG Navigation

5.2 Two Obstacles with Target Estimation

The experiments with two stationary obstacles involve a finite space with two point obstacles with known safety sphere radiuses. The positions of the obstacles are randomly chosen in each simulation run while making sure they obstruct the path of the UAV

between the start point and the destination. The initial velocity of the UAV is also chosen randomly uniformly distributed between following limit (in meters per second).

$$\begin{aligned} 5 \le u \le 20 \\ -5 \le v \le 5 \\ -5 \le w \le 5 \end{aligned} \tag{5.7}$$

For first 10 seconds of the simulation, no control is applied and EKF estimates the obstacle and target positions in an open loop system (pre-run). After the pre-run, EKF reinitializes the state vector and error covariance and then system applies one of the two navigation law in order to find a collision free path. Figure 5.6 shows an example scenario of 3D space where simulations are being conducted.

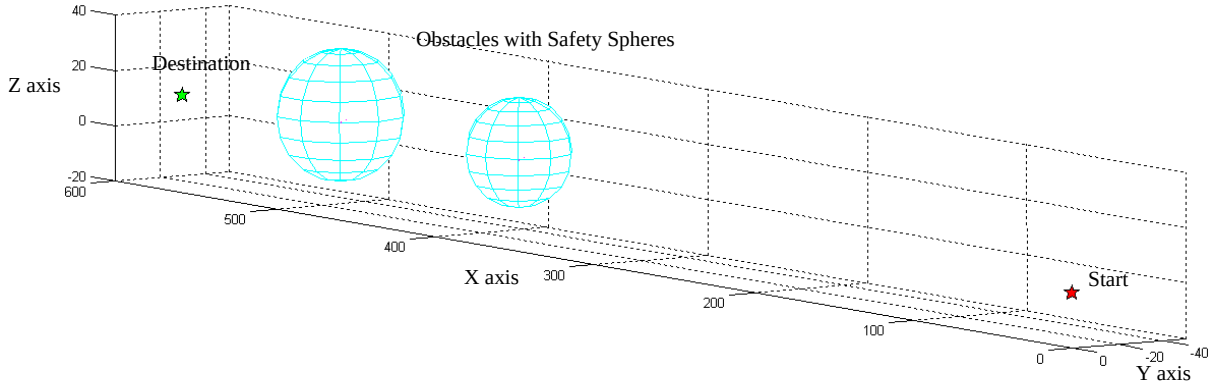

Figure 5.6: Finite 3D Space with Two Obstacles

5.2.1 DGG Navigation Strategy

In this part of the experiments, DGG navigation law was applied in order to find a collision free path. A total of 1000 simulation runs performed in order to test the effectiveness of the DGG navigation law while estimating two obstacles and target positions with EKF. Based on the success criterion described in section 5.1, following Table IV shows the Percentage of successes.

TABLE IV: Success Percentage with DGG Navigation

Success band	**% Run Satisfying the Success Conditions for Obstacle 1**	**% Run Satisfying the Success Conditions for Obstacle 2**	**% Run Satisfying the Success Conditions for Both Obstacles Simultaneously**
S-1	48.6	51.9	31.4
S-2	68.8	75	54.6

S-3	82.5	88.5	74.3
S-4	88.5	95.6	84.3

The Figure 5.7 shows the UAVs closest approach with obstacle 1 as the Percentage of the safety sphere radius. Similarly Figure 5.8 shows the UAVs closest approach with obstacle 2 as the Percentage of the safety sphere radius. The Figure 5.9 shows the final distance of UAV from the destination.

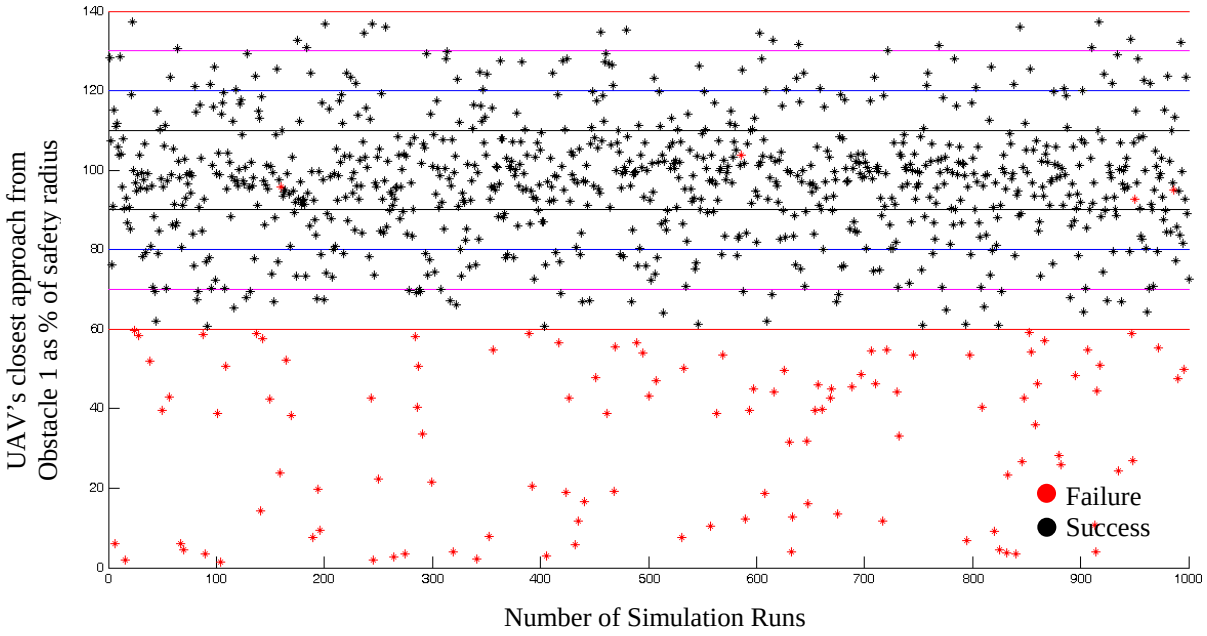

Figure 5.7: UAV's Closest Approach to Obstacle 1 with DGG Navigation

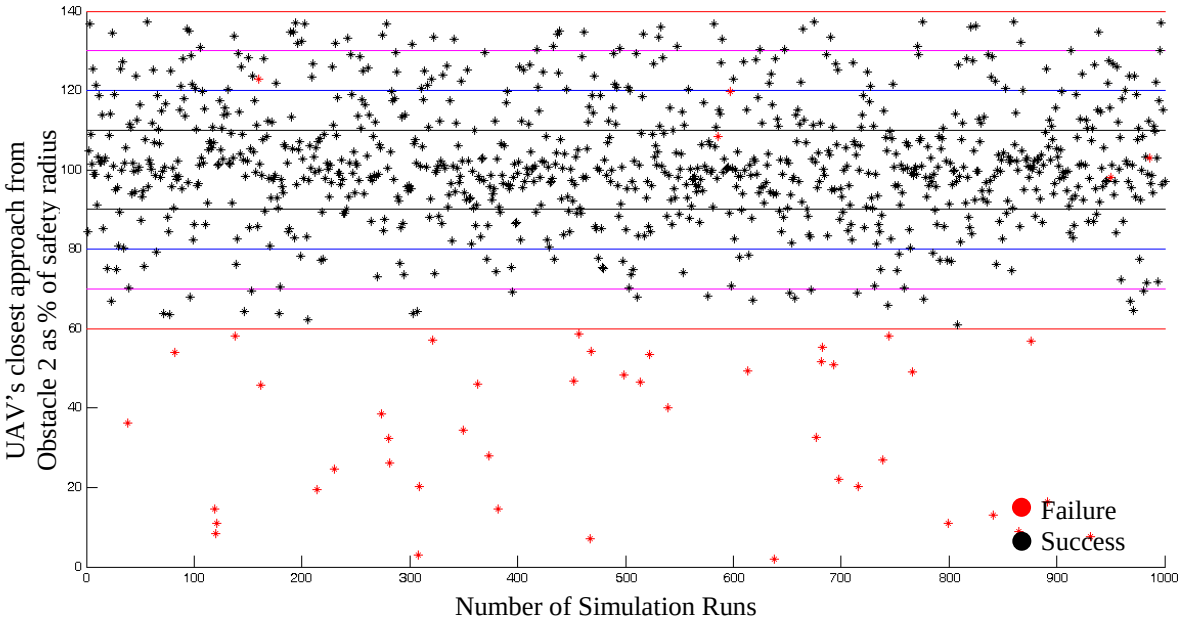

Figure 5.8: UAV's Closest Approach to Obstacle 2 with DGG Navigation

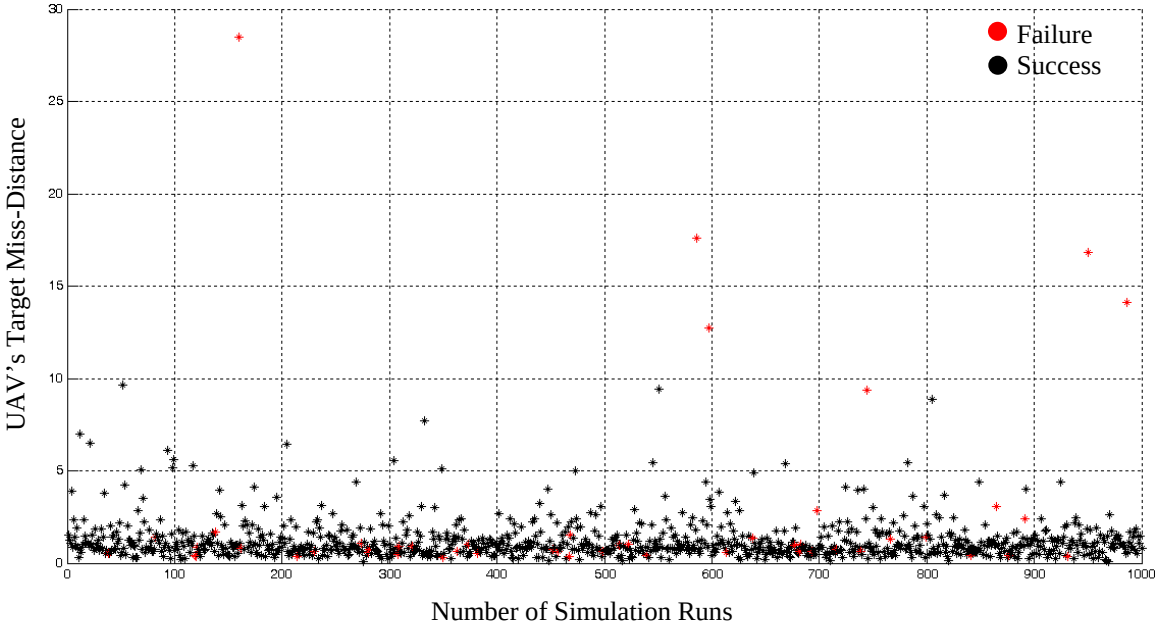

Figure 5.9: UAV's Target Miss-Distance with DGG Navigation

5.2.2 NGG Navigation Strategy

In this part of the experiments, NGG navigation law was applied in order to find a collision free path. A total of 1000 simulation runs performed in order to test the effectiveness of the NGG navigation law while estimating two obstacles and target

positions with EKF. Based on the success criterion described in section 5.1, following Table V shows the Percentage of successes.

TABLE V: Success Percentage with NGG Navigation

Success band	% Run Satisfying the Success Conditions for Obstacle 1	% Run Satisfying the Success Conditions for Obstacle 2	% Run Satisfying the Success Conditions for Both Obstacles Simultaneously
S-1	46.8	55.3	27.6
S-2	79.3	70	57.5
S-3	80.4	90.5	73.9
S-4	87.8	96.4	84.7

The Figure 5.10 shows the UAVs closest approach with obstacle 1 as the Percentage of the safety sphere radius. Similarly Figure 5.11 shows the UAVs closest approach with obstacle 2 as the Percentage of the safety sphere radius. The Figure 5.12 shows the final distance of UAV from the destination.

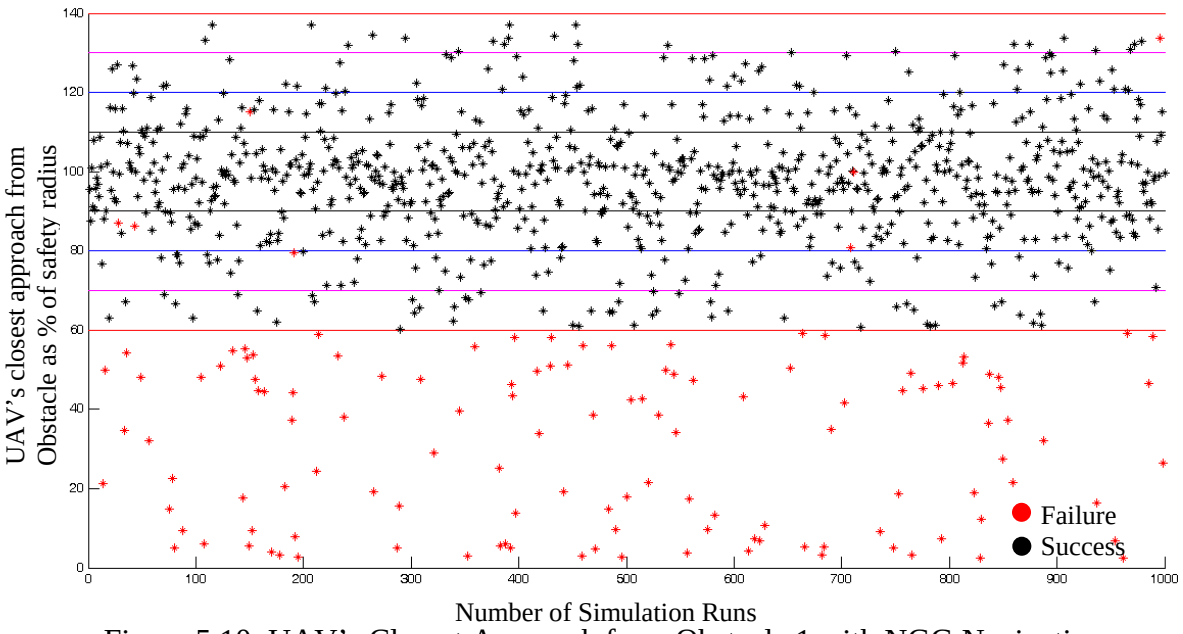

Figure 5.10: UAV's Closest Approach from Obstacle 1 with NGG Navigation

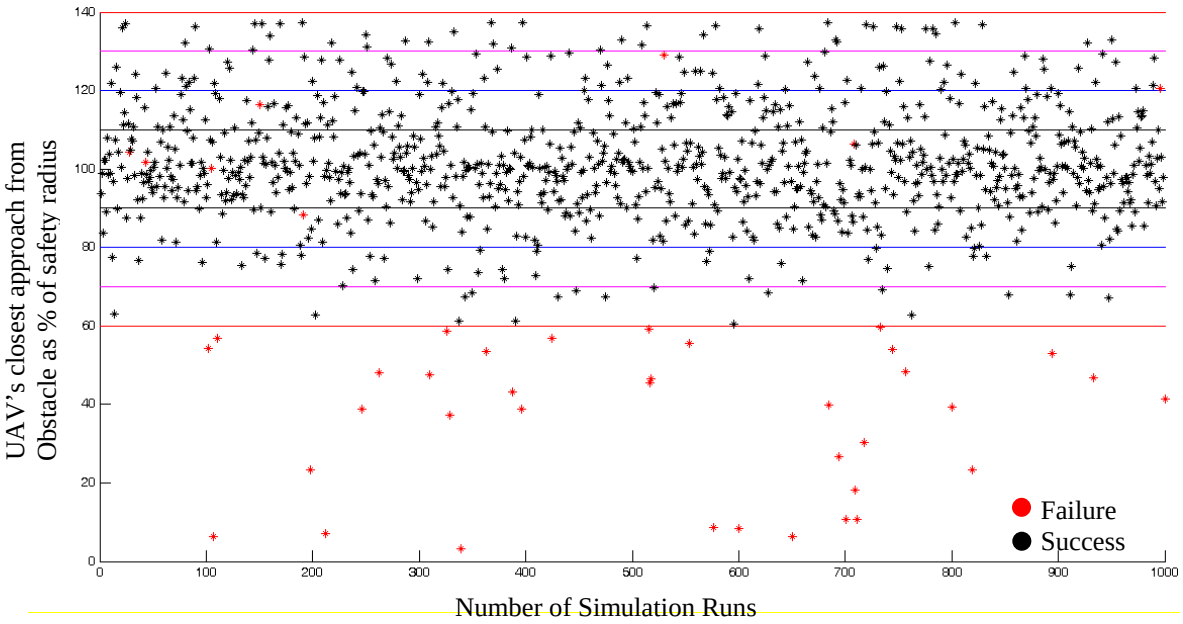

Figure 5.11: UAV's Closest Approach from Obstacle 2 with NGG Navigation

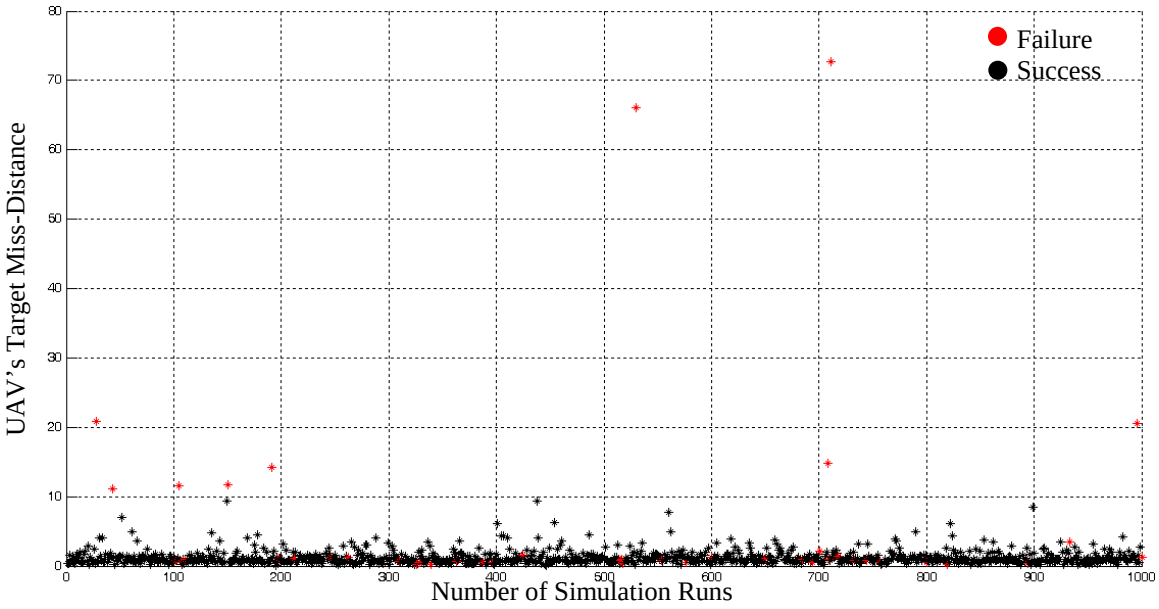

Figure 5.12: Distances between Destination and UAV's Final Location with NGG Navigation

5.3 Sigma Bound Test

It is important to perform the consistency check while using EKF. Sigma bound test is one such test which checks whether EKF is behaving close to what is theoretically

expected. During the simulation runs of system, Sigma-bound test was also performed in order to check if error in state estimates lies within the standard deviation given by the error covariance matrix P. Figure 5.13 shows the Sigma-bound test for obstacle estimation in one of the simulation run.

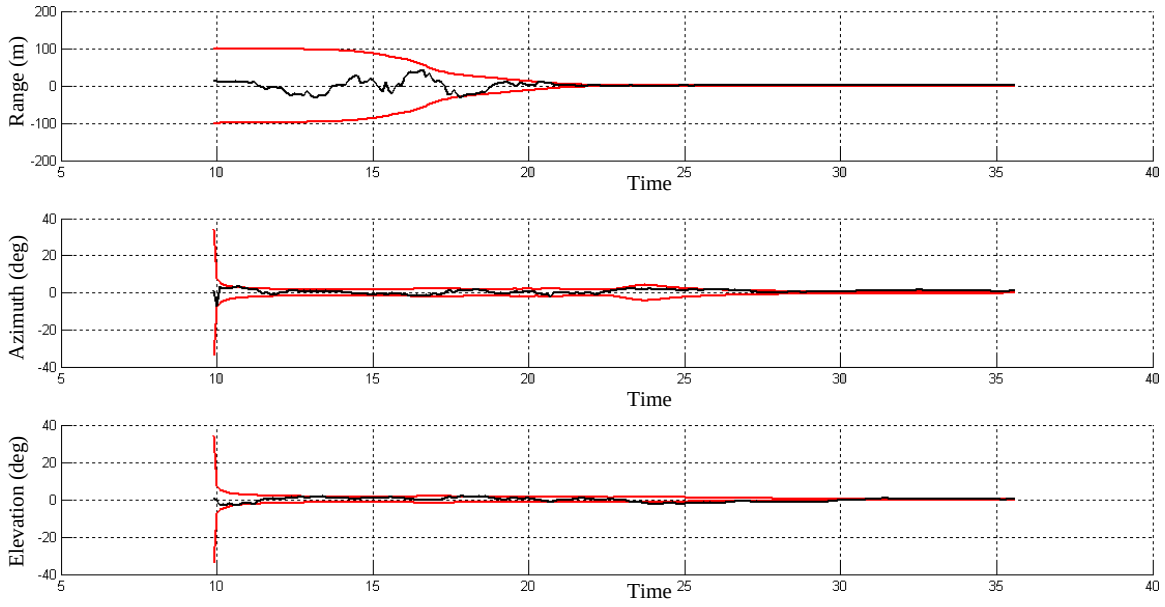

Figure 5.13: One Sigma Bound Test

With each simulation run, Sigma bound test was performed i.e. estimation error in state element is compared with the square root of the corresponding diagonal element of the P matrix. At the same time estimation errors are also compared with two times and three times of the error standard deviation. Following Tables VI to IX show the results of Sigma-bound test in terms of percentage of estimation errors bounded by 1-Sigma, 2-Sigma and 3-Sigma bounds for each set of the simulation. Results are given is average terms i.e. after running 1000 simulations, the Sigma test results are averaged for all simulations.

Table VI: Single Obstacle with Target Estimation while Applying DGG Navigation

State Vector Elements	Average % of the Estimation Error Bounded by 1-Sigma	Average % of the Estimation Error Bounded by 2-Sigma	Average % of the Estimation Error Bounded by 3-Sigma
r_{ob}	66.05	94.24	98.97
θ_{ob}	64.18	95.67	99.35
ϕ_{ob}	63.92	96.57	98.21
r_{tr}	77.73	91.44	94.83
θ_{tr}	68.59	94.46	98.68
ϕ_{tr}	69	94.49	98.76

Table VII: Single Obstacle with Target Estimation while Applying NGG Navigation

State Vector Elements	Average % of the Estimation Error Bounded by 1-Sigma	Average % of the Estimation Error Bounded by 2-Sigma	Average % of the Estimation Error Bounded by 3-Sigma
r_{ob}	65.95	91.85	97.05
θ_{ob}	62.65	96. 34	97.73
ϕ_{ob}	63.02	96.93	99.25
r_{tr}	77.06	91.89	94.5
θ_{tr}	69.22	94.26	98.65
ϕ_{tr}	68.14	94.28	98.79

Table VIII: Two Obstacles with Target Estimation while Applying DGG Navigation

State Vector Elements	Average % of the Estimation Error Bounded by 1-Sigma	Average % of the Estimation Error Bounded by 2-Sigma	Average % of the Estimation Error Bounded by 3-Sigma
r_{ob1}	65.98	93.06	99.46
θ_{ob1}	66.56	89.93	96.69
ϕ_{ob1}	65.32	90.63	96.31
r_{ob2}	70.28	96.43	99.18
θ_{ob2}	64.62	89.87	95.33
ϕ_{ob2}	63.74	89.72	95.37
r_{tr}	78.24	93.45	96.60
θ_{tr}	68.90	95.03	99.34
ϕ_{tr}	67.92	94.64	99.31

Table IX: Two Obstacles with Target Estimation while Applying NGG Navigation

State Vector Elements	Average % of the Estimation Error Bounded by 1-Sigma	Average % of the Estimation Error Bounded by 2-Sigma	Average % of the Estimation Error Bounded by 3-Sigma
r_{ob1}	67.98	86.710	93.93
θ_{ob1}	64.36	96.56	99.58
ϕ_{ob1}	62.56	84.21	91.49
r_{ob2}	62.41	89.63	95.81
θ_{ob2}	61.38	94.83	99.14
ϕ_{ob2}	68.58	95.69	99.49
r_{tr}	78.32	93.39	96.81
θ_{tr}	66.38	94.81	99.08
ϕ_{tr}	69.16	94.82	98.85

The Sigma bound tests whether the error covariance matrix is representing the estimation errors as theoretically expected. The diagonal elements of the P matrix correspond to the estimation error variance of the individual state vector elements. Theoretically, the estimation errors should be normally distributed with covariance given by the P matrix. If estimation errors are normally distributed, they should be according to the following empirical law of normal distribution:

- Percentage of estimation errors bounded by 1-Sigma: 68.26
- Percentage of estimation errors bounded by 2-Sigma: 95.44
- Percentage of estimation errors bounded by 3-Sigma: 99.74

It is clear from above tables that EKF developed in this algorithm is following these bonds with a satisfactory degree. This indicates that EKF is performing as theoretically expected with good convergence.

Chapter 6

Conclusions

A solution to the mission-critical problem of reactive collision avoidance of UAVs is presented in this paper. Two newly developed guidance strategies which are highly suited to solve the problem, i.e. NGG and DGG are validated in this paper with vision based sensing. The effectiveness of the two guidance laws in the presence of the visual information is demonstrated in the simulation results to problem scenarios with stationary obstacles in 3D.

The vision based sensing is one of most focused research area in UAV automation because of its many significant advantages. Additionally in nature, most of the birds and insects apply vision based sensing as most effective tool for navigation and detection. The significance of guidance strategies implemented here for collision avoidance lies in their emphasis on rapid alignment, in contrast with existing guidance laws. Since collision avoidance is a potentially mission-critical component, specifications such as low control effort, minimum fuel consumption etc. are not as crucial as the safety of the vehicle.

A possible extension to the ideas presented in this paper is the problem of collision avoidance with moving and maneuvering obstacles such as other UAVs (e.g., urban warfare situation). Note that the use of the vision based sensing is not limited to only collision avoidance. Other potential applications would include guidance of missiles and guided munitions since the obstacles can in fact be the real target. So instead of avoiding them, objective could be to intercept them.

Additionally, an effort has been made into developing an effective vision based collision avoidance technique; some of the assumptions made in this study (such as stationary point obstacles, perfect information about UAV’s position and velocity,

availability of assumed image processor, kinematic model) are not realistic and hence needs to be relaxed in future studies.

Acknowledgements

This work is supported by AFRL/AOARD, USA under the contract number FA-23860814102, which is being operated at Indian Institute of Science, Bangalore with the project code SID/PC 99148/08-9018.

Appendix A

Derivation of the Expressions for Criteria of Criticality and Aiming Point in the Collision Cone Approach

From Figure 4.1 we note that r_i and u_i are perpendicular, giving the following results:

$$r_i \cdot u_i = (X_r + du_i) \cdot u_i = 0 \qquad i = 1,2 \tag{A1}$$

Since $u_i \cdot u_i = 1$ we have:

$$X_r \cdot u_i + d = 0 \tag{A2}$$

The vector u_i is defined in a 2D space spanned by X_r and V and can therefore be expressed as

$$u_i = \alpha_i X_r + \beta_i V \tag{A3}$$

where α_i and β_i are scalar coefficients. Using this in (A2),

$$X_r \cdot (\alpha_i X_r + \beta_i V) + d = \alpha_i \| X_r \|^2 + \beta_i (X_r \cdot V) + d = 0 \tag{A4}$$

$$\alpha_i = -\frac{\beta_i (X_r \cdot V) + d}{\| X_r \|^2} \tag{A5}$$

Further, $u_i \cdot u_i = 1$, therefore from (A3),

$$(\alpha_i X_r + \beta_i V) \cdot (\alpha_i X_r + \beta_i V) = \alpha_i^2 \| X_r \|^2 + 2\alpha_i \beta_i (X_r \cdot V) + \beta_i^2 \| V \|^2 = 1 \tag{A6}$$

Substituting (A4) in (A6):

$$\frac{(\beta_i (X_r \cdot V) + d)^2}{\| X_r \|^2} - 2\frac{(\beta_i (X_r \cdot V) + d)\beta_i (X_r \cdot V)}{\| X_r \|^2} + \beta_i^2 \| V \|^2 = 1 \tag{A7}$$

Solving for β_i,

$$\beta_i = \pm \sqrt{\frac{\| X_r \|^2 - d^2}{\| X_r \|^2 \| V \|^2 - (X_r \cdot V)^2}} = \pm c \tag{A8}$$

Let $\beta_1 = +c$ and $\beta_2 = -c$, from (A3) and (A4)

$$u_1 = -\frac{1}{\| X_r^2 \|} (c(X_r \cdot V) + d) X_r + cV \tag{A9}$$

$$u_2 = \frac{1}{\| X_r^2 \|} (c(X_r \cdot V) - d) X_r - cV \tag{A10}$$

Then, the vectors r_i may be expressed as

$$r_1 = X_r + du_1 = X_r - \frac{d}{\| X_r^2 \|}(c(X_r \cdot V) + d)X_r + cdV \tag{A11}$$

$$r_2 = X_r + du_2 = X_r + \frac{d}{\| X_r^2 \|}(c(X_r \cdot V) - d)X_r - cdV \tag{A12}$$

After adding (A11) and (A12), we have

$$r_1 + r_2 = 2\frac{\| X_r \|^2 - d^2}{X_r^2} X_r \tag{A13}$$

$$X_r = \frac{1}{2}\frac{X_r^2}{\| X_r \|^2 - d^2}(r_1 + r_2) \tag{A14}$$

Substituting (A14) in (A11):

$$r_1 = \frac{1}{2}\left(1 - \frac{cd(X_r \cdot V)}{\| X_r \|^2 - d^2}\right)(r_1 + r_2) + cdV \tag{A15}$$

Therefore, the expression for V is derived to be:

$$V = \frac{1}{cd}\left(r_1 - \frac{1}{2}\left(1 - \frac{cd(X_r \cdot V)}{\| X_r \|^2 - d^2}\right)(r_1 + r_2)\right) \tag{A16}$$

$$V = \frac{1}{2}\left(\frac{X_r \cdot V}{\| X_r \|^2 - d^2} + \frac{1}{cd}\right)r_1 + \frac{1}{2}\left(\frac{X_r \cdot V}{\| X_r \|^2 - d^2} - \frac{1}{cd}\right)r_2 \tag{A17}$$

Comparing (A17) and (4.1), we have the following expressions

$$a = \frac{1}{2}\left(\frac{X_r \cdot V}{\| X_r \|^2 - d^2} + \frac{1}{cd}\right) \tag{A18}$$

$$b = \frac{1}{2}\left(\frac{X_r \cdot V}{\| X_r \|^2 - d^2} - \frac{1}{cd}\right) \tag{A19}$$

(A18) and (A19) give the expressions for a and b, used to determine the criticality of an obstacle. The unit vectors u_1 and u_2 in (A9) and (A10) are used for aiming point computation.

Appendix B

Derivation of the Virtual Aiming Point in the Sphere Tracking Algorithm for Tracking Safety Sphere

The objective of the sphere tracking algorithm is to compute a virtual aiming point. The virtual aiming point is a point on the surface of the safety sphere and is the intersection of X_r and the safety sphere (see Figure 4.2). The equation of the line X_r in the two-point form is:

$$\frac{x - x_{ob}}{x_v - x_{ob}} = \frac{y - y_{ob}}{y_v - y_{ob}} = \frac{z - z_{ob}}{z_v - z_{ob}} = k \tag{B1}$$

where k is the constant of proportionality. Further, the equation of the safety sphere is:

$$(x - x_{ob})^2 + (y - y_{ob})^2 + (z - z_{ob})^2 = d^2 \tag{B2}$$

We wish to find the intersection of the relative distance line with the safety sphere. Therefore, substituting (B1) in (B2) yields the following equation:

$$k^2(x_v - x_{ob})^2 + k^2(y_v - y_{ob})^2 + k^2(z_v - z_{ob})^2 = d^2 \tag{B3}$$

$$k = \pm \frac{d}{\sqrt{(x_v - x_{ob})^2 + (y_v - y_{ob})^2 + (z_v - z_{ob})}} \tag{B4}$$

The virtual aiming point $X_{vap}\,(x_{vap}, y_{vap}, z_{vap})$ is computed from (B1) by substituting for k:

$$x_{vap} = k(x_v - x_{ob}) + x_{ob} \tag{B5}$$

$$y_{vap} = k(y_v - y_{ob}) + y_{ob} \tag{B6}$$

$$z_{vap} = k(z_v - z_{ob}) + z_{ob} \tag{B6}$$

There are two solution sets for X_{vap} depending on the sign of k. Note that in the above equations for the x, y and z coordinates, the value of k may either be uniquely positive or uniquely negative (from (B4)). The solution chosen is the point closer to the vehicle's position X_v .

Appendix C

Vision Based Obstacle Avoidance Algorithm

The step-by-step algorithm implemented in the numerical simulations is given below. For simplicity algorithm presented here is for position estimation of only one object. It can be easily augmented for simultaneous position estimation of multiple objects by augmenting the state vector and EKF accordingly.

Step 1: After getting the first processed image from the image processor, initialize the state vector $\hat{X}_r(0)$ according to number of objects found.

$$\hat{X}_r(0) = \begin{bmatrix} \hat{r}_{ob}(0) & \hat{\theta}_{ob}(0) & \hat{\phi}_{ob}(0) \end{bmatrix}^T$$

here $\hat{r}_{ob}(0)$ is given with 50% uncertainty assuming some prior information, while $\hat{\theta}_{ob}(0)$ and $\hat{\phi}_{ob}(0)$ are given by first measurements.

Step 2: Initialize the Error Covariance Matrix P_0:

$$P_0 = diag\left(a_1 e^2 \quad a_2 \left(\frac{r_0 \times W_V}{100} \right)^2 \quad a_3 \left(\frac{r_0 \times W_V}{100} \right)^2 \right)$$

here e is the maximum range measurement uncertainty (in meters) given by the 50% of the distance between the start point and the destination, $r_0 = 20$ is maximum percentage measurement noise. $a_1 = 1$, $a_2 = 2$ and $a_3 = 2$ are tuning parameters. W_V is camera's width of view (in radians) assumed $\frac{2}{3}\pi$.

Step 3: Initialize the Process Noise Covariance Q:

$$Q = diag\begin{pmatrix} 0.2 & 0.025 & 0.025 \end{pmatrix}$$

Step 4: Run EKF in open loop system for 10 Seconds (Pre-run)

4.1: Propagate the State Vector $\hat{X}^{+}_{r_{k-1}} \rightarrow \hat{X}^{-}_{r_k}$:

$$\dot{\hat{X}}_r = \begin{bmatrix} u\cos\hat{\theta}_{ob}\cos\hat{\phi}_{ob} + v\sin\hat{\theta}_{ob}\cos\hat{\phi}_{ob} + w\sin\hat{\phi}_{ob} \\ -\dfrac{\sin\hat{\theta}_{ob}}{\hat{r}_{ob}\cos\hat{\phi}_{ob}}u + \dfrac{\cos\hat{\theta}_{ob}}{\hat{r}_{ob}\cos\hat{\phi}_{ob}}v \\ -\dfrac{\cos\hat{\theta}_{ob}\sin\hat{\phi}_{ob}}{\hat{r}_{ob}}u - \dfrac{\sin\hat{\theta}_{ob}\sin\hat{\phi}_{ob}}{\hat{r}_{ob}}v + \dfrac{\cos\hat{\phi}_{ob}}{\hat{r}_{ob}}w \end{bmatrix}$$

4.2: Propagate the Covariance Matrix $P^{+}_{k-1} \rightarrow P^{-}_{k}$:

$$\dot{P}(t) = A(t)P(t) + P(t)A^{T}(t) + Q$$

here $A(t)$ is calculated by using expression in equation (3.35).

4.3: Compute Measurement Error Covariance Matrix R_k:

$$R_k = diag\left(\left(r_{ob_k}\frac{W_V}{100}\times\frac{1}{3}\right)^2 \quad \left(r_{ob_k}\frac{W_V}{100}\times\frac{1}{3}\right)^2\right)$$

here r_{ob_k} is given by the equation (3.25) from range dependent measurement noise model.

4.4: Compute Kalman Filter Gain K_k:

$$K_k = P_k^{-}C_k^{T}\left[C_kP_kC_k^{T} + R_k\right]^{-1}$$

here $C_k = \left.\dfrac{\partial h}{\partial X}\right|_{\hat{X}_k^{-}(t)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

4.5: Take the Measurement Y_k:

$$Y_k = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}X_{r_k} + v_k$$

Note that here X_{r_k} is the actual value of the state vector and v_k is the measurement noise vector.

4.6: Update the State Vector $\hat{X}_{r_k}^{-} \rightarrow \hat{X}_{r_k}^{+}$:

$$\hat{X}_{r_k}^{+} = \hat{X}_{r_k}^{-} + K_k \left[Y_k - h(\hat{X}_{r_k}^{-}) \right]$$

4.7: Update the Covariance Matrix $P_k^{-} \rightarrow P_k^{+}$:

$$P_k^{+} = (I - K_k C_k) P_k^{-} (I - K_k C_k)^T + K_k R_k K_k^T$$

4.8: For first 10 seconds, at every grid point of time, repeat steps 4.1 to 4.7.

Step 5: Reinitialize the State Vector $\hat{X}_r(0)$

$$\hat{X}_r(0) = \frac{1}{N} \sum_{i=1}^{N} \hat{X}_r(i)$$

here N is the number of estimations made by EKF during pre-run (Step 4).

Step 6: Reinitialize the Error Covariance Matrix P_0:

$$P_0 = diag \left(a_1 e^2 \quad a_2 \left(\frac{r_0 \times W_V}{100} \right)^2 \quad a_3 \left(\frac{r_0 \times W_V}{100} \right)^2 \right)$$

Step 7: Run EKF with Guidance in Closed Loop System till the UAV crosses the Target

5.1: Repeat steps 4.1 to 4.7 i.e. Estimate the Obstacle or Target Position.

5.2: Perform the Smoothing Operation

$$\hat{\bar{X}}_{r_k} = \frac{1}{n} \sum_{i=k-n-1}^{k} \hat{X}_{r_i}$$

here $\hat{\bar{X}}_{r_k}$ is smoothed estimate, $\hat{X}_{r_k}$ original estimate at time instant k and $n = 10$.

5.3: If Estimated Object is Obstacle:

- Check for collisions (Apply Collision Cone approach given in Appendix A)
  - Compute a and b
  - If $a > 0$ AND $b > 0$, obstacle is critical
- Find X_{ap}
  - If $a > b$, $X_{ap} = X_v + r_1$
  - If $b > a$, $X_{ap} = X_v + r_2$

5.4: If Estimated Object is Target:

$$X_{ap} = \hat{\bar{X}}_{r_k}$$

5.5: Find $(X_{ap})_{XY}$, $(X_{ap})_{XZ}$, V_{XY} and V_{XZ}: Project X_{ap} and V on to XY and XZ planes

5.6: XY plane:

- Angle error $\theta_y = \cos^{-1}\left(\frac{V_{XY} \cdot (X_{ap})_{XY}}{\| V_{XY} \| \| (X_{ap})_{XY} \|}\right)$
- Desired velocity $v^* = \frac{[(X_{ap})_{XY}]_y}{[(X_{ap})_{XY}]_x} u$
- Sign convention:
  - If $v^* < v$, $\theta_y > 0$
  - If $v^* > v$, $\theta_y < 0$
- Compute control a_y
  - DGG: $a_y = k_v(v - v^*)$
  - NGG: $a_y = \hat{k}_v \sin\theta_y$ where $\hat{k}_v = k_v \frac{\sqrt{u^2 + v^2}\sqrt{u^2 + v*^2}}{u}$

5.7: XZ plane:

- Angle error $\theta_z = \cos^{-1}\left(\frac{V_{XZ} \cdot (X_{ap})_{XZ}}{\| V_{XZ} \| \| (X_{ap})_{XZ} \|}\right)$
- Desired velocity $w^* = \frac{[(X_{ap})_{XZ}]_z}{[(X_{ap})_{XZ}]_x} u$
- Sign convention:

- If $w^* < w$, $\theta_z > 0$
- If $w^* > w$, $\theta_z < 0$

- Compute control a_y
  - DGG: $a_z = k_w(w - w^*)$
  - NGG: $a_z = \hat{k}_w \sin\theta_z$ where $\hat{k}_w = k_w \dfrac{\sqrt{u^2 + w^2}\sqrt{u^2 + w^{*2}}}{u}$

5.8: State update:

$$\begin{bmatrix} \dot{X} \\ \dot{V} \end{bmatrix} = \begin{bmatrix} V \\ U \end{bmatrix} \text{ with } U = \begin{bmatrix} 0 \\ a_y \\ a_z \end{bmatrix}$$

5.9: If any point of time:

$$|X_r| \le d$$

That is if vehicle position violates obstacle safety boundary, activate Sphere Tracking algorithm given in Appendix B.

References

[1] DeGarmo M. & Nelson G. M. (2004). Prospective Unmanned Aerial Vehicle Operations in the Future National Airspace System. AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum, Chicago, Illinois, Sep. 20-22, 2004.

[2] C. De Wagter & J.A. Mulder (2005). Towards Vision-Based UAV Situation Awareness. AIAA Guidance, Navigation, and Control Conference, San Francisco, California, Aug 15-18, 2005.

[3] Chakravarthy A. & Ghose D. (1998). Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 28 (5), 562-574.

[4] Mujumdar A. and Padhi R. (2010). Nonlinear Geometric and Differential Geometric Guidance of UAVs for Reactive Collision Avoidance. Proceedings of the AIAA Guidance, Navigation, and Control Conference, 2010 Accepted.

[5] Camacho E.F. & Bordons C. (1999). *Model Predictive Control*. New York: Springer.

[6] Leung C., Huang S., Kwok N. & Dissanayake G. (2006). Planning under uncertainty using model predictive control for information gathering. *Robotics and Autonomous Systems*, *54*, 898910.

[7] Shim D. H., Chung H. & Sastry S. (2006). Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles. *IEEE Robotics and Automation Magazine*, *13* (3), 27-33.

[8] Shim D.H. & Sastry S. (2007). An Evasive Maneuvering Algorithm for UAVs in See-and-Avoid Situations. *Proceedings of the American Control Conference*, 3886-3891.

[9] Grossberg S. (1988). Nonlinear neural networks: Principles, mechanisms, architecture. *Neural Networks, 1*, 17-61.

[10] Wang X., Yadav V., & Balakrishnan S.N. (2007). Cooperative UAV Formation Flying With Obstacle/Collision Avoidance. *IEEE Transactions on Control Systems Technology, 15* (4), 672-679.

[11] Carpenter G.A. & Grossberg S. (1991). *Pattern Recognition By Self-Organizing Neural Networks*. The MIT Press.

[12] Bryson A. E. & Ho Y.C. (1975). *Applied Optimal Control: Optimization, Estimation, and Control*. New York: Taylor & Francis

[13] LaValle S.M. (2006). *Planning Algorithms*. Cambridge University Press.

[14] Camacho E.F. & Bordons C. (1999). *Model Predictive Control*. New York: Springer.

[15] Y. Watanabe, A.J. Calise, and E.N. Johnson (2007). Vision-based Obstacle Avoidance for UAVs. *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, August 2007.

[16] W.H.Hines, D.C.Montgomery, D.M.Goldsman and C.M.Borror (2003). *Probability and Statistics in Engineering*. John Wiley&Sons. 2003.

[17] Zarchan P. (1994). *Tactical and Strategic Missile Guidance*. AIAA.

[18] Han S.C. & Bang H. (2004). Proportional Navigation-Based Optimal Collision Avoidance for UAVs. Proceedings of *2nd International Conference on Autonomous Robots and Agents*, December 13-15, 2004, Palmerston North, New Zealand.

[19] Watanabe Y., Calise A.J & Johnson E.N. (2006). Minimum Effort Guidance for Vision-Based Collision Avoidance. *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, Keystone, Colorado, 21 - 24 August 2006.

[20] Crassidis J. L. & Junkins J. L. (2004). *Optimal Estimation of Dynamic Systems*. CRC Press.

[21] Ben-Asher J.Z. (1993). Minimum-Effort Interception of Multiple Targets. *Journal of Guidance, 16* (3), 600-602.

[22] Horn, B. K. and Schunck, B. G. (1981). Determining Optical Flow. Artificial Intelligence, Vol. 17, 1981, pp. 185–203.

[23] Bernard, C. (1997). Wavelets and Ill Posed Problems: Optic Flow and Scattered Data Interpolation, Ph.D. thesis, Ecole Polytechnique, 91128 Palaiseau Cedex, France, may 1997.

[24] Beauchemin, S. S. and Barron, J. L. (1995). The computation of optical flow. ACM Computing Surveys 27, Vol. 27, 1995, pp. 433–467.

[25] Barrows, G. L. (2000). Optic Flow micro-sensors and integration into mini UAVs. http://www.centeye.com/, 2000.

[26] Paul Borke (1997). Intersection of Two Circles. http://local.wasp.uwa.edu.au/~pbourke/geometry/2circle/, retrieved on 26*th* April 2010.

[27] Strang, G. (1998). *Linear Algebra and its Applications" (3nd ed.)*. Thomson Learning.

[28] Enns D., Bugajski D., Hendrick R., & Stein G.(1994). Dynamic Inversion: An Evolving Methodology for Flight Control Design. *International Journal of Control, 59* (1), 71-91.

[29] Kim B. & Calise A.J. (1997). Nonlinear Flight Control Using Neural Networks. *Journal of Guidance, Control and Dynamics, 20* (1), 26-33

[30] Slotine J.J.E. & Li W. (1991). *Applied Nonlinear Control*, Prentice Hall.

[31] Krozel J. & Peters M. (1997). Strategic Conflict Detection and Resolution for Free Flight. *Seagull technology, Inc.*

[32] Merz A.W. (1991). Maximum-Miss Aircraft Collision Avoidance. *Dynamics and Control, 1* (1), 25-34.

[33] [Park et al.(2008)] Park J.W.,Oh H.D. & Tahk M.J. (2008). UAV Collision Avoidance Based on Geometric Approach. *SICE Annual Conference*, 2122-2126.