DYNAMIC ESTIMATION OF OBSTACLE POSITION WITH VISION SENSING FOR REACTIVE COLLISION AVOIDANCE OF UAVs

Radhakant Padhi and Ankush Gupta
Department of Aerospace Engineering
Indian Institute of Science
Bangalore-560 012, India
Email : padhi@aero.iisc.ernet.in

Abstract

It is very crucial for unmanned aerial vehicles to have autonomous obstacle detection and avoidance capability for their survivability during flight. This paper proposes and validates the application of extended Kalman filter for online obstacle position estimation with a vision based sensor and the usefulness of this information with two recently developed guidance algorithms for collision avoidance. The vision sensor is assumed to continuously sense the environment in front of the vehicle during flight. In case any obstacle is detected, the information from this sensor is then utilized in the filter to estimate the obstacle position online. Simultaneously, the collision cone approach is applied to predict any potential collision in future and, in case of a potential threat, to steer away the vehicle in order to avoid the collision. This is done by first computing a suitable ‘aiming point’ towards which the velocity vector of the vehicle must be aligned as soon as possible and then by using either of two recently proposed guidance laws, namely nonlinear geometric guidance and differential geometric guidance (which are identically same with appropriate gain correlation, but otherwise are different) to achieve this objective. Exhaustive simulation studies show that this overall strategy is fairly successful.

Introduction

Potential applications of Unmanned Aerial Vehicles (UAVs) include reconnaissance, environmental monitoring, border patrol, search and rescue operations, disaster relief, traffic monitoring etc. Hence UAVs are expected to be ubiquitous in the near future for both civilian as well as military applications [1, 2, 3]. In many of these applications require the UAVs to fly at very low altitudes, and hence, close to artificial and natural structures (e.g. buildings, towers, trees, power lines etc). This situation poses a serious risk of a fatal collision resulting in mission failure as well as vehicle loss. Hence autonomous reactive collision avoidance is a basic requirement for successful operation of UAVs.

Collision avoidance can broadly be classified into global and local path planning algorithms, both of which need to be addressed in a successful mission. Where as global path planning (which is mainly done offline) broadly lays out a path that reaches the goal point, local collision avoidance algorithms, which are usually fast, reactive and carried out online, ensure safety of the vehicle from unexpected and unforeseen obstacles/collisions. Reactive collision avoidance is a problem of local path planning, where after sensing an obstacle that was not accounted before in global path planning, the vehicle must correct its flying path quickly to avoid the potential danger. An interested reader can see [16] for a comprehensive review of various collision avoidance and path planning algorithms proposed in the literature.

Since UAVs are usually small flying machines, there are several critical issues that need to be addressed for successful implementation of such an autonomous collision avoidance algorithm. First, any UAV should be as light as possible, as otherwise it compromises the endurance and maneuvering capability, thereby limiting the UAVs effectiveness for many missions. Because of this requirement, the payload of an UAV is severely restricted by size and weight, and hence, the onboard power supply...
system (usually a battery) is very limited in its resources. Due to this reason selection of the onboard processor is usually done keeping in mind the fact that it must be power efficient. This ultimately leads to the selection of energy efficient processors, which are usually poor in their computational efficiency. Moreover, since the vehicle must keep on flying to sustain itself in air (which is especially true for fixed wing UAVs), the computational time window is quite limited. Hence, any algorithm that needs to be executed in the onboard computer must be computationally efficient [4]. Second, again due to size and weight restrictions, obstacle sensing device must compact and be lightweight [1]. Third, military missions require stealthiness [5] i.e. such an UAV should not be detectable while operating inside enemy aerospace. Additionally, to minimize the overall cost, each component used in an UAV should be as economic as possible.

Both stealthness as well as power efficiency requirement leads to the conclusion that the sensors employed should be passive in nature. In view of these limitations, a vision based sensor (video camera) is a very suitable choice since it is compact, lightweight, economical, and passive. Increasing computational power of small processors and consequent improvement in digital image processing are other motivations for applying vision based sensing [6]. A major fundamental disadvantage of vision sensing, however, is the lack of depth perception, which in turn hinders its ability to perceive the world in three dimensions. This is because, unlike other active sensors (e.g. lidars) who rely on the reflected signal radiated from itself, a vision sensor relied only on the in-coming signal. In fact, getting a 3D perception of the world from the sequence of 2D images that the camera receives is a challenging problem for current computer vision systems [7]. Hence a suitable algorithm must be applied in order to estimate the depth of obstacles from 2D images. At the same time, algorithm must be computationally efficient to be implementable onboard UAV. In this paper, Extended Kalman Filter (EKF) based estimation technique is applied for this purpose. An important advantage of EKF is that it is recursive in nature hence it can be implemented online efficiently [8]. Moreover, it is a proven technique which has been applied successfully in a large number of complex practical problems. On the other hand, EKF is also "fragile" (i.e. it operates successfully only within a narrow band of tuning parameters), and hence good care must be taken for selecting its tuning parameters.

After estimating the obstacle position, the next logical step is to predict whether it is a critical one and, in case of any potential threat, to steer away the vehicle to avoid any potential collision. This requires a suitable collision prediction as well as an appropriate guidance logic. Among various guidance logics available in the literature [16], an interesting minimum effort guidance (MEG) law based on optimal control theory is proposed in [9] for reactive collision avoidance. However, reactive collision avoidance problems do not necessarily have minimum effort requirements. Additionally MEG distributes the control effort over the available time period and causes vehicle to maneuver until the aiming point, which can lead to safety ball intrusion, which can be quite risky given the fact that obstacle position is not known with absolute certainty. In this paper, two recently developed nonlinear guidance laws, named as Nonlinear Geometric Guidance (NGG) and Differential Geometric Guidance (DGG), are incorporated for guidance purpose. For the details about these guidance laws one can refer to [11]. These guidance laws first apply collision cone approach [12] to detect any potential collision and then compute an alternate aiming point in order to avoid it if necessary. The main feature of both guidance algorithms is that they align the velocity vector of the vehicle along the aiming point within a part of the available time-to-go i.e. these guidance laws produce higher control at the beginning itself. Therefore, there is no need to maneuver all the way until the aiming point is reached. These strategies ensure the quick reaction and safety of the vehicle. After avoiding the obstacles, the destination serves as final aiming point and hence the same guidance is applicable when UAV path is obstacle free. Hence, these guidance laws accomplish both obstacle avoidance and destination seeking. Note that this falls into the Level-4 of the autonomous mission control levels as discussed in a recent review paper [17].

**Problem Formulation of Obstacle Position Estimation**

**Modeling of UAV Motion and Vision Sensor**

In Fig.1, let \( F_i \) be an inertial reference frame. The origin of frame \( F_i \) is fixed by the UAV’s initial position. The axes of \( F_i \) are parallel to the that of UAV body frame \( F_0 \). Without the loss of generality, it is assumed that camera is fixed at UAV’s center of mass. The position of UAV is known with a reasonable certainty with the help of GPS and/or INS [13] in the frame \( F_i \). Let \( X = [x \ y \ z]^T \) is UAV position vector, \( V = [u \ v \ w]^T \) is UAV velocity vector, and \( a = [a_x \ a_y \ a_z]^T \) is UAV control (acceleration) input in reference frame \( F_i \). The UAV motion dynamics are modelled as \( \ddot{X} = V \) and \( \dot{V} = a \). The
velocity along X-axis is considered to be constant i.e. \( a_x = 0 \).

Let \( X_{\text{ob}} = \begin{bmatrix} x_{\text{ob}} \\ y_{\text{ob}} \\ z_{\text{ob}} \end{bmatrix}^T \) be the obstacle’s position in \( F_i \), then \( \dot{X}_{\text{ob}} = 0 \) i.e. obstacle is considered stationary.

From Fig. 1, the state vector \( X_r = \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}^T \) position of the obstacle in frame \( F_U \) can be written as Equation (1).

\[
X_r = X_{\text{ob}} - X
\]  

(1)

Since \( \dot{X}_{\text{ob}} = 0 \), the relative motion dynamics of obstacle will be \( \dot{X}_r = -\dot{X} \).

\[
\dot{X}_r = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{z}_r \end{bmatrix}^T = \begin{bmatrix} u_r \\ v_r \\ w_r \end{bmatrix}^T
\]  

(2)

Figure 2 shows the problem geometry. Here \( f \) is focal length of the camera and \( Y_k = \begin{bmatrix} y_k \\ z_k \end{bmatrix}^T \) is the locus of the obstacle projection on the image plane at time instant \( k \). The relationship between output \( Y_k \) and state vector \( X_r \), can be easily shown from Fig. 2 as Equation (3). Note that output equation is a nonlinear function of the state vector \( X_r \). Additionally measurement noise \( v_k \) is also present. Without loss of generality it is assumed here that the camera is placed near the nose of the aircraft (which is usually the case), the distance between the CG of the aircraft and camera CG shall be used during transformation.

\[
Y_k = \begin{bmatrix} y_k \\ z_k \end{bmatrix}^T = \frac{f}{X_r(k)} \begin{bmatrix} y_r(k) \\ z_r(k) \end{bmatrix} + v_k
\]  

(3)

However, Equation (3) can lead to singularity since state element \( x_r \) appears in denominator. It is very likely that at some point \( x_r \to 0 \) i.e. when UAV crosses the obstacle on X-axis. Moreover, in our experience estimation errors do not converge properly with nonlinear output equation while applying EKF. These issues an be avoided if \( X_r \) is defined in spherical coordinate system as Equation (4) instead.

\[
X_r(k) = \begin{bmatrix} r_r(k) \\ \theta_r(k) \\ \phi_r(k) \end{bmatrix}^T
\]  

(4)

Here \( r_r \) is range, \( \theta_r \) is azimuth and \( \phi_r \) represents elevation of obstacle. The locus of obstacle projection on image plane is measured in terms of angles \( \theta' \) and \( \phi' \) as shown in Fig. 3. So output equation becomes a linear function of the state vector. Another advantage of having a linear output equation is that \( C \) matrix becomes a constant. It saves significant computational load since \( C \) can be pre-stored and there is no need calculate \( C \) during each iteration.

\[
Y_k = \begin{bmatrix} \theta_k \\ \phi_k \end{bmatrix}^T = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} r_r(k) \\ \theta_r(k) \\ \phi_r(k) \end{bmatrix}^T
\]  

\[
Y_k = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} X_r(k)
\]  

(5)

Relative Obstacle Motion Dynamics in Spherical Coordinate Frame

The relationship between Cartesian and Spherical coordinates of obstacle in frame \( F_U \) can be given by the following sets of equations.

\[
r_r^2 = x_r^2 + y_r^2 + z_r^2
\]  

(6a)

\[
\tan \theta_r = \frac{y_r}{x_r}
\]  

(6b)

\[
\tan \phi_r = \frac{z_r}{\sqrt{x_r^2 + y_r^2}}
\]  

(6c)

\[
x_r = r_r \cos \theta_r \cos \phi_r
\]  

(7a)

\[
y_r = r_r \sin \theta_r \cos \phi_r
\]  

(7b)

\[
z_r = r_r \sin \phi_r
\]  

(7c)

Differentiating Equation (6a) and substituting Equations (2) and (7)

\[
\dot{r}_r = u_r \cos \theta_r \cos \phi_r + v_r \sin \theta_r \cos \phi_r + w_r \sin \phi_r
\]  

(8)

Similarly differentiating Equation (6b) and substituting Equations (2) and (7)

\[
\dot{\theta}_r = \frac{\sin \theta_r}{r_r \cos \phi_r} u_r + \frac{\cos \theta_r}{r_r \cos \phi_r} v_r
\]  

(9)
Finally differentiating Equation (6c) and substituting Equations (2) and (7)

\[ \dot{\phi}_r = -\frac{\cos \theta}{r} \sin \phi + \frac{\sin \theta}{r} \sin \phi + \frac{\cos \phi}{r} w_r \]  

(10)

Equations (8), (9) and (10) constitute the dynamics of the state vector defined in Equation (4).

\[
\begin{bmatrix}
\dot{r} \\
\dot{\theta} \\
\dot{\phi}
\end{bmatrix}
= 
\begin{bmatrix}
w \cos \theta, \cos \phi + v \sin \theta, \cos \phi + w \sin \phi \\
-\sin \theta, \cos \phi - v \cos \theta, \cos \phi - v \cos \phi \\
-\cos \theta \sin \phi, \sin \theta \sin \phi - v \cos \phi, \sin \theta \cos \phi + v \cos \phi
\end{bmatrix}
\]

(11)

\[ \dot{X}_r = f(X_r) \]  

(12)

Here \( f(\cdot) \) represents the nonlinear dynamics of state vector \( X_r \).

**Vision Based Position Estimation Using EKF**

This section presents the details of EKF implementation. The objective is to estimate the state vector \( X_r \) defined in Equation (4) based on output \( Y_k \) given by Equation (5).

**Dynamics of State Vector and Process Noise**

The system dynamics given by Equations (12) is rewritten to accommodate the process noise as Equation (13).

\[ \dot{X}_r = f(X_r) + G(t) w(t) \]  

(13)

Here \( G(t) \) is process noise influence matrix and \( w(t) \) is zero mean Gaussian noise with covariance given by Equation (14) [14]. The process noise in each state element is considered independent of each other hence process noise influence matrix will be unity i.e. \( G(t) = I \).

\[ E[w(t)] = 0 \]

\[ E[w(t)w^T(\tau)] = Q(t)\delta(t-\tau) \]  

(14)

**Measurement Noise Model**

It is assumed that measurement noise is zero mean Gaussian process having properties given by Equation (15) [14].

\[ E[v_k] = 0 \]

\[ E[v_k v_j^T] = R_k \delta_{k-j} \]  

(15)

Another assumption is that the magnitude of the measurement noise is a function of obstacle range i.e. higher the distance between the sensor and the obstacle, higher will be the measurement uncertainty. It is a reasonable assumption since closer you get to an obstacle, better is the quality of the visual information obtained. Based on this philosophy, a nonlinear function of range has been devised to calculate the measurement noise covariance.

\[ m_k = m_o \left(1 - \delta r_r(k)\right) \]  

(16)

here \( m_k \) is the percentage noise at time instant \( k \), \( m_o \) represents the initial percentage measurement noise \( r_r(k) \) is the range of the obstacle at time instant \( k \), and \( \delta \) is a tuning parameter, which defines how \( m_k \) changes with change in \( r_r(k) \). Moreover, based on nature of visual sensing, it is assumed that \( m_k \) changes slowly for higher range value and drops quickly for lower values of range. Fig 4 shows variation in \( m_k \) with range from zero to 500 meters for \( m_o = 20 \) and \( \delta = 0.99 \). The measurement noise covariance is given by Equation (17).

\[ R_k = \left(m_k \frac{w_v}{100} \times \frac{1}{3}\right)^2 \]  

(17)

here \( w_v \) is the angular width of 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 \( m_k \) or 3\( \sigma \) for almost all of the cases.
Initialization and Pre-Run of the EKF

- **Initialization of State Vector**: For initialization purpose, range of obstacle is assumed known with 50% uncertainty. Following equation shows initialization of state vector.

\[
\hat{X}_r(0) = \begin{bmatrix} \hat{r}_r(0) \\ \hat{\theta}_r(0) \end{bmatrix}
\]

where \( \hat{r}_r(0) \) is known as 50% error. The domain of initial obstacle range \( (r_r(0)) \) is between 300 to 500 meters while \( \hat{r}_r(0) \) is initialized as a randomly selected value within ±50% of the \( r_r(0) \). Other state elements \( \hat{\theta}_r(0) \) and \( \hat{\phi}_r(0) \) are initialized with the very first measurement from vision sensor.

- **Initialization of Error Covariance Matrix**: Based on known initial uncertainty in range estimation and initial measurement noise, \( P_o \) is initialized as the following diagonal matrix:

\[
P_o = \text{diag} \left( a_1 e^2_R, a_2 \left( \frac{m_o \times w_v}{100} \right)^2, a_3 \left( \frac{m_o \times w_v}{100} \right)^2 \right)
\]

here \( e_R \) is the maximum range measurement uncertainty (in meters), \( m_o \) is initial measurement noise \( a_1, a_2 \) and \( a_3 \) are scalar parameters, which needs to be tuned (Section - Tuning of EFK). \( w_v \) is camera’s width of view (in radians) defined earlier.

- **Pre-Run of EKF**: It is highly recommended that EKF runs sufficiently before its actual application so that initial error can be stabilized [14]. Hence we start EKF 10 seconds before applying any control in order to avoid it. It is assumed that obstacles are detected, by video sensor and image processor, sufficiently ahead of time. The UAV remains on its original global path during pre-run period. After that, it reinitializes \( P_o \) again as Equation (19) and \( \hat{X}_r(0) \) as the average of all previous estimates. Before performing the averaging operation, it is necessary to project all the estimates to the same time since all estimates are taken from different UAV positions during different time instants. So first all estimates made during pre-run are converted from Spherical to Cartesian system. Then these estimates are transferred to reference frame \( F_I \) from \( F_U \) by Equation (20) (Fig.1). Then we average all projected estimates in \( F_I \) as Equation (21). Finally we reinitialize the \( \hat{X}_r(0) \) by converting averaged estimate back from \( F_I \) to \( F_U \) and then from Cartesian to Spherical system.

\[
\hat{X}_{ob}(i) = X(i) + f^{\text{cart}}_{\text{sph}} \left( \hat{X}_r(i) \right)
\]

\[
\hat{X}_r(0) = f^{\text{sph}}_{\text{cart}} \left( \frac{1}{N} \sum_{i=1}^{N} \hat{X}_{ob}(i) - X(N) \right)
\]

here \( f^{\text{cart}}_{\text{sph}} \) represents the function which converts coordinates from spherical system to Cartesian system, similarly \( f^{\text{sph}}_{\text{cart}} \) converts Cartesian coordinates into the spherical coordinates. \( X(N) \) is the UAV position in \( F_I \) at the end of the pre-run. \( N \) is the number of estimations made by EKF during pre-run.

**Propagation of State and Error Covariance**

Based on the system dynamics derived in Section-Relative Obstacle Motion Dynamics in Spherical Coordinate Frame, the states are propagated as Equation (22).

\[
\hat{X}_r = f(\hat{X}_r)
\]

The propagation of error covariance matrix or the \( P \) matrix is given as Equation (23).

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

here \( A(t) = \frac{\partial f}{\partial X_r} \left| \hat{X}_r(t) \right. \).

**Updation of State and Error Covariance**

Once the measurements arrive, EKF updates the previously propagated state and error covariance based on the current measurements. First Kalman Gain is computed as Equation (24).

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

here \( C_k = \left. \frac{\partial h}{\partial X_r} \right| \hat{X}_r(0) \) \begin{bmatrix} 0 & 1 \end{bmatrix} \) (constant) and \( R_k \) is measurement noise covariance given by measurement
noise model described earlier. The state vector and error covariance matrix are updated according to Equations (25) and (26) respectively.

\[ \hat{X}_r^+(k) = \hat{X}_r^-(k) + K_k [Y_k - h(\hat{X}_r^-(k))] \]  
\[ P_k^+ = (I - K_k C_k) P_k^- (I - K_k C_k)^T + K_k R_k K_k^T \]  

**Smoothing of Estimate**

Sometimes due to momentarily high measurement noise, state estimate fluctuate. These fluctuations can produce large associative control accelerations since it is a closed loop system. This 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. The smoothing operation performed as following :

\[ \hat{X}_{ob}^+(i) = \frac{1}{n} \sum_{i=k-n}^{k} \left( f_{\text{sph}}^\text{cart}(\hat{X}_r^k(i)) + X(i) \right) \]  

here \( n \) is the number of previous estimates used for smoothing operation (\( n = 10 \) in our case), \( \hat{X}_{ob}^k(k) \) is smoothed estimate and \( \hat{X}_r^k(k) \) original estimate at time instant \( k \). The value of \( n \) is set at a low number because position estimations get better as UAV gets closer to the obstacle. Hence only recent estimates are considered for smoothing operation.

**Tuning of EKF**

After developing the whole EKF, its tuning is the final step. Tuning of EKF requires proper selection of parameters \( Q, P_o, \) and \( R_k \). As stated earlier, EKF is fragile in nature i.e. it works well only for a narrow band of \( Q, P_o, \) and \( R \) parameters [14]. 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 Equation (17). The \( R_k \) is given by following formula :

\[ R_k = \text{diag} \left( \frac{w_k}{500} \times \frac{1}{3}, \frac{w_k}{500} \times \frac{1}{3} \right) \]  
\[ P(0) \]  
\[ Q = \text{diag} \left( 0.2, 0.025, 0.025 \right) \]  

here the diagonal elements of \( Q \) matrix are selected through trial and error. First diagonal element of \( Q \) matrix represents the process noise covariance (in meter) for the range elements of the state vector. Similarly second and third diagonal elements represent the process noise covariance for angle elements (in radians) of state vector. The entire estimation algorithm can be found in a step by step form in Appendix.

**Guidance of UAV Using Vision Information**

Once the obstacle position is estimated, the objective reduces to applying the guidance to navigate the UAV around it.

**Collision Cone Philosophy**

The first task is to finding out whether obstacle is critical i.e. if collision with obstacle is imminent. For that, first we apply the Collision Cone approach [12]. The collision cone is an effective tool for :

- Detecting an incoming collision
- Finding an alternate direction of motion in order to avoid the collision

The construction of the collision cone is shown in Fig.5. A spherical safety boundary of radius \( d \) is con-
constructed around the obstacle. An obstacle is considered to be critical if the UAV is expected to violate the safety boundary in future. Since the collision cone approach operates in two dimensions, the plane containing $X_r$ and $V$ is considered for constructing the cone. The safety sphere thus reduces to a circle $\beta$ in this plane. A collision cone is constructed by dropping tangents from the UAV to the circle $\beta$. If the velocity vector $V$ lies within the collision cone, the UAV will violate $\beta$ in due course and result in collision. Thus the obstacle is said to be critical. The collision cone criterion can be stated as, if $a > 0$ AND $b > 0$, the obstacle under consideration is said to be critical. The aiming point is determined in the following way:

\[
\text{if } a > b, \quad X_{ap} = X + r_1 \\
\text{if } b > a, \quad X_{ap} = X + r_2
\]  

(31)

Note that if obstacle is not critical then destination serves as the aiming point. Further explanation on collision cone approach can be found in [11, 12]. After fixing the aiming point, Fig.6 shows the geometry of the resultant guidance problem. The objective is to align the VAV velocity vector $V$ in the direction of aiming point $X_{ap}$ i.e., eliminating the angle $\theta$. This 3D problem can be seen as a combination of two separate 2D problems in the XY and XZ planes. The guidance objective can be restated as to generate control accelerations $a_y$ and $a_z$ so that $v \rightarrow v^*$ and $w \rightarrow w^*$ respectively within available fraction of the $t_{go}$.

Nonlinear Geometric and Differential Geometric Guidance

The Nonlinear Geometric Guidance (NGG) [11] law is as follows:

\[
\begin{bmatrix}
  a_y \\
  a_z
\end{bmatrix} = \begin{bmatrix}
  \hat{k}_v \sin \theta_y \\
  \hat{k}_w \sin \theta_z
\end{bmatrix}
\]  

(32)

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

The nonlinear Differential Geometric Guidance (DGG) [11] is based on Dynamic Inversion (DI) [15], a control strategy used for output tracking of nonlinear systems. The main advantage of DI is that it essentially guarantees global asymptotic stability with respect to the tracking error. The DGG law is given as Equation (33).

\[
\begin{bmatrix}
  a_y \\
  a_z
\end{bmatrix} = \begin{bmatrix}
  -k_v (v - v^*) \\
  -k_w (w - w^*)
\end{bmatrix}
\]  

(33)

The constant $k_v$ and $k_w$ are 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 $t_{go}$. To make these guidance laws more realistic, a limit of $\pm 20 \text{m/s}^2$ is applied for both control accelerations.

The DGG is equivalent to the NGG, if its control gains $k_v$ and $k_w$ are set as given by Equation (34) and (35) [11]. With these gain settings for DGG, the controls generated by it will be exactly same as controls generated by NGG. Since both guidance strategies are directly correlated, the NGG also guarantees the global asymptotic stability. Mode details on these guidance laws can be found in [11].

\[
k_v = \hat{k}_v \left( \frac{u}{\sqrt{u^2 + v^2}} \right) \]  

(34)

\[
k_w = \hat{k}_w \left( \frac{u}{\sqrt{u^2 + w^2}} \right) \]  

(35)

Simulation Results

Test Environment

The simulations are conducted in two scenarios. Single Obstacle with Destination Estimation and Two Obstacles with Destination Estimation in 3D separately for each of the guidance strategy. The simulations involves a finite space with one or two point obstacles with pre-selected safety sphere radius. The position of the obstacle is randomly chosen in each simulation run while making sure it obstructs the path of the UAV. The origin and destination are chosen randomly with distance about 600m between them. Obstacles and Destination are located about 100m from each other. Fig.7 shows the UAV trajectory in two obstacles case with DGG guidance. Fig.8 shows the XY and XZ views of UAV trajectory with phases of the algorithm, while Fig.9 and 10 show the control output generated by DGG strategy in terms of $g$ (gravitational acceleration). The initial velocity of the UAV is also
chosen randomly between limits given by Equation (36) (in meters per second).

\[ 5 \leq u \leq 20 \]
\[ -5 \leq v \leq 5 \]
\[ -5 \leq w \leq 5 \]  \hspace{1cm} (36)

The Process noise is generated with covariance given by Equation (30) and added to the UAV’s position \( X \) during each iteration of the simulation run. Similarly to simulate the measurement noise, normally distributed random noise generated with covariance given by Equation (28) and magnitude given by Equation (16) and added to the real values of the relative obstacle position while taking the measurements. Note that since both DGG and NGG are directly correlated and equivalent to each other with proper gain settings, results are presented independently with no gain correlation.

**EKF Validation Check**

It is important to perform the consistency check while using EKF [8]. 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 \( P \). 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 Figs.11 and 12 show the Sigma-bound test for obstacle 1 and obstacle 2 position estimation for UAV flight shown in Fig.7 respectively.

**Success Criterion**

The success of the algorithm was tested on three criterions:

- Violation of the safety sphere
- Divergence from the safety sphere
- UAV’s destination 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 reasonably good then UAV’s closet 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. Based on these success criterions, different segments of success are created defined by the band of the closet approach of the UAV with obstacles and destination. These segments of success are named as S-1, S-2, S-3, S-4 and S-5 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-5 represents most relaxed case. These conditions are described in Table-1.

**Results : Single Obstacle with Destination Estimation**

A total of 1000 simulation runs performed in order to test the effectiveness of both DGG and NGG laws while estimating the obstacle and destination position with EKF. Based on the success criterion described earlier, following Table-2 shows the percentage of successes with the DGG and NGG guidance strategies.

Figures 13 and 14 show the UAVs closest approach with obstacle as the percentage of the safety sphere radius with DGG and NGG guidance respectively.

**Two Obstacles with Target Estimation**

A total of 1000 simulation runs performed in order to test the effectiveness of both DGG and NGG laws while estimating the obstacle and target position with EKF. Based on the success criterion described earlier, following

<table>
<thead>
<tr>
<th>Table-1 : Different Success Bands</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Success Band</strong></td>
</tr>
<tr>
<td>S-1</td>
</tr>
<tr>
<td>S-2</td>
</tr>
<tr>
<td>S-3</td>
</tr>
<tr>
<td>S-4</td>
</tr>
<tr>
<td>S-5</td>
</tr>
</tbody>
</table>
Tables-3 and 4 shows the percentage of successes with DGG and NGG guidance strategies respectively.

### Conclusions

This paper address the problem of reactive obstacle avoidance for UAVs with vision sensing. An EKF based technique is developed in order to estimate the obstacle position based on output from a vision sensor. Then two recently developed guidance strategies, NGG and DGG are incorporated to achieve the guidance objective. To test the effectiveness of this algorithm, a number of simulations are carried out in 3D scenario with stationary obstacles. The simulation results demonstrate that, this algorithm provides a good estimate of the obstacle position with reasonable certainty in the presence of synthetically generated process and measurement noise. Results also demonstrate the viability of these guidance laws in the presence of vision sensing.

### Acknowledgement

This work is supported by AOARD/AFRL, USA under the contract number FA-23860814102, which operated at the Society for Innovation and Development (SID), Indian Institute of Science, Bangalore with the project code SID/PC 99148.

### References


Appendix

Steps of Vision Based Obstacle Avoidance Algorithm

The step-by-step algorithm implemented in the numerical simulations is given in this appendix. For simplicity, algorithm presented is for position estimation of only one object. However, it can be easily scaled up 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 following.

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

Step 2: Initialize the Error Covariance Matrix $P_o$:

$$P_o = \text{diag} \left\{ a_1 \epsilon, a_2 \frac{m_o \times w}{100} \right\}$$
Step 3: Initialize the Process Noise Covariance \( Q \):
\[
Q = \text{diag} (0.2, 0.025, 0.025)
\]

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

4.1: Propagate the State Vector \( \hat{X}_r^+(k - 1) \rightarrow \hat{X}_r^-(k) \):
\[
\hat{X}_r = \begin{bmatrix}
  u \cos \hat{\theta} \cos \hat{\phi} + v \sin \hat{\theta} \cos \hat{\phi} + w \sin \hat{\phi} \\
  - \sin \hat{\theta} \cos \hat{\phi} \\
  - \frac{\cos \theta}{r} \sin \hat{\phi} + \frac{\sin \theta}{r} \cos \phi + \frac{w}{r} \sin \phi \\
  - \frac{\cos \theta}{r} \sin \phi - \frac{\sin \theta}{r} \cos \phi + \frac{w}{r} \cos \phi
\end{bmatrix} \]

4.2: Propagate the Covariance Matrix \( P_{k-1}^+ \rightarrow P_k^- \):
\[
\hat{P}(t) = A(t) \hat{P}(t) + \hat{P}(t) A^T(t) + Q
\]

4.3: Compute Measurement Error Covariance Matrix \( R_k \):
\[
R_k = \text{diag} \left( \frac{w}{m_k \times 100} \times \frac{2}{3}, \frac{w}{m_k \times 100} \times \frac{2}{3} \right)
\]

here \( m_k \) is given by the Equation (16) from range dependent measurement noise model.

4.4: Compute Kalman Filter Gain \( K_k \):
\[
K_k = P_k^- C_k^T \left( C_k P_k C_k^T + R_k \right)^{-1}
\]

here \( C_k = \frac{\partial h}{\partial \hat{x}} \bigg|_{\hat{x} = \hat{x}} \)
\[
= \begin{bmatrix}
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\]

4.5: Take the Measurement \( Y_k \):
\[
Y_k = \begin{bmatrix}
  0 & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix} \hat{X}_r + v_k
\]

Note that \( \hat{X}_k \) is the actual value of the state vector and \( v_k \) is the measurement noise.

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 \( \hat{X}_r(0) \)
\[
\hat{X}_r(0) = f_{\text{cart}} \left( X(i) + f_{\text{sph}} \left( \hat{X}_r(i) \right) \right)
\]

Step 6: Reinitialize the Error Covariance Matrix \( P_o \):
\[
P_o = \text{diag} \left( a_1^2, a_2^2, a_3^2 \right)
\]

Step 7: Run EKF with Guidance in Closed Loop System till the UAV reaches the Destination

7.1: Repeat steps 4.1 to 4.7 i.e. Estimate the Obstacle or Destination Position.

7.2 Perform the Smoothing Operation
\[
\hat{X}_{ob}^\wedge(i) = \frac{1}{n} \sum_{i=k-n}^{k} \left( f_{\text{cart}} \left( \hat{X}_r(i) \right) + X(i) \right)
\]

here \( \hat{X}_{ob}^\wedge \) is the smoothed estimate used to generate the control accelerations, \( \hat{X}_r^\wedge \) is original estimate at time instant \( k \) and \( n = 10 \).

7.3: If Estimated Object is Obstacle:
• Check for collisions (Apply Collision Cone approach [11])
  - 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 $b > a$, $X_{ap} = X_v + r_2$

7.4 : If Estimated Object is Destination :

$$X_{ap} = \frac{\hat{X}}{\hat{k}}$$

7.5 : Find $(X_{ap})_{XY} \cdot (X_{ap})_{XZ} \cdot V_{XY}$ and $V_{XZ}$ : Project $X_{ap}$ and $V$ on to $XY$ and $XZ$ planes

7.6 : $XY$ plane : (Same approach for $XZ$ plane)

• Angle error $\theta_y = \cos^{-1} \left( \frac{V_{XY} \cdot (X_{ap})_{XY}}{\| V_{XY} \| \| (X_{ap})_{XY} \|} \right)$

7.7 : State update :

$$\begin{bmatrix} \dot{X} \\ \dot{V} \\ \dot{a} \end{bmatrix} = \begin{bmatrix} V \\ a \end{bmatrix}$$

with $a = \begin{bmatrix} 0 & a_y & a_z \end{bmatrix}^T$

• Desired velocity $v^* = \frac{[ (X_{ap})_{XY} ]_y}{[ (X_{ap})_{XY} ]_x}$

• 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

$$\hat{k}_v = k_v \frac{\sqrt{u^2 + v^2}}{u}$$

7.8 : Measurement of Obstacle Projection on Image Plane in Terms of $\theta$ and $\phi$

![Fig.1 Inertial Reference Frame and UAV Body Frame](image1)

![Fig.2 Video Sensor Model](image2)

![Fig.3 Measurement of Obstacle Projection on Image Plane in Terms of $\theta$ and $\phi$](image3)

![Fig.4 Measurement Noise as a Function of Obstacle Range](image4)
Fig. 11 Obstacle 1 Estimation Error

Fig. 12 Obstacle 2 Estimation Error

Fig. 13 UAV’s Closest Approach to Obstacle with DGG Guidance

Fig. 14 UAV’s Closest Approach to Obstacle with NGG Guidance