Team Elite Ball Knowledge: Collision Avoidance for a Simulated Drone

AE403W Final Report

Parham Khodadi, Wyatt Welch, Ethan Dueck, Nicklous Ngo, and Luis Salas.

San Diego State University, Department of Aerospace Engineering

This paper presents the design, development, and testing of an autonomous collision avoidance system for a model multirotor by team Elite Ball Knowledge (EBK) at San Diego State University (SDSU). This project is developed as part of the Baseball Avoidance Multirotor (BAM) challenge from NASA Langley Research Center (LaRC). The objective of this project is to detect, predict, and avoid a baseball in real time while minimizing deviation from the drone's original trajectory. The system is divided into three main components: perception, prediction, and planning. The entire system was implemented entirely in MATLAB/Simulink. The final program implementation successfully avoided the baseball; however, it was not able to return to its original path autonomously on any tested trajectories.

Fig. 1 Team Logo

I. Nomenclature

ACA	=	Autonomous Collision Avoidance
BAM	=	Baseball Avoidance Multirotor
$SLAM$	=	Simultaneous Localization and Mapping
$LiDAR$	=	Light Detection and Ranging
RGB-D	=	Red-Green-Blue Depth Camera
IMU	=	Inertial Measurement Unit
EKF	=	Extended Kalman Filter
t_{cpa}	=	Time of Closest Point of Approach
Δt	=	Time step
$\mathbf{p}$	=	Position vector $[x\ y\ z]^T$
$\mathbf{v}$	=	Velocity vector $[v_x\ v_y\ v_z]^T$
$\mathbf{a}$	=	Acceleration vector
$\mathbf{g}$	=	Gravitational acceleration vector

$\mathbf{p}_{rel}$	=	Relative position vector (drone to baseball)
$\mathbf{v}_{rel}$	=	Relative velocity vector
d_{sep}	=	Separation distance between drone and baseball
Q	=	Process noise covariance matrix
R	=	Measurement noise covariance matrix
Q_{noise}	=	Process noise scaling factor
R_{noise}	=	Measurement noise scaling factor
q_{min}	=	Minimum process noise bound
r_{min}	=	Minimum measurement noise bound
σ_i^2	=	Variance of measurement in direction i
v_{mag}	=	Magnitude of velocity vector
v_{noise}	=	Velocity-dependent process noise term
C_d	=	Drag coefficient
$\mathbf{a}_{drag}$	=	Aerodynamic drag acceleration
$\mathbf{a}_{total}$	=	Total acceleration (drag + gravity)
$\mathbf{p}_k$	=	Position at time step k
$\mathbf{v}_k$	=	Velocity at time step k
$\mathbf{p}_{k+1}$	=	Position at next time step
$\mathbf{v}_{k+1}$	=	Velocity at next time step
$\mathbf{x}_{hat}$	=	Estimated state vector (EKF output)
$bball\_traj$	=	Predicted baseball trajectory array
$bball\_traj\_iner$	=	Measured baseball trajectory (inertial frame)
A	=	Amplitude of oscillation (general variable)

Contents

II. Introduction

Autonomous Collision Avoidance (ACA) is an important and actively researched area within aerospace technologies, particularly for unmanned aerial systems operating in several dynamic and uncertain environments. The NASA Baseball Avoidance Multirotor (BAM) challenge provides a unique framework for researchers exploring this problem by requiring a multirotor drone to detect and avoid a fast-moving projectile while maintaining stable flight and minimizing trajectory deviation.

The challenge introduces three core technical problems: perception, prediction, and planning. The perception involves detecting a baseball within a full 360-degree field of view and converting sensor data into usable position data. The prediction focuses on estimating the future trajectory of the incoming baseball with minimal error. Finally, the planning requires generating an avoidance maneuver that is dynamically feasible for a multirotor while meeting strict timing constraints.

To address these challenges, this project developed a modular system implemented in MATLAB/Simulink. Our early design efforts focused on integrating high-fidelity simulation tools such as AirSim, ROS2, and LiDAR-based perception. However, due to system compatibility issues and time constraints, the project scope shifted toward a more controlled and reliable Simulink-based approach. This allowed our team to focus on validating the prediction and planning algorithms using more idealized sensor inputs with added noise.

The final system was tested by running a batch of 100 randomly generated trajectories where the baseball could reach anywhere between 0 and 10 feet of the drone at its time of closest point of approach (t_{cpa}). Our program was able to successfully avoid the baseball most of the time. However, it was not able to return to its original desired trajectory. We were not able to fix this due to time constraints.

III. Literature Review

Autonomous collision avoidance is widely studied in both civilian and military aerospace applications. Previous research efforts have demonstrated the feasibility of integrating sensing, prediction, and control systems to detect and avoid potential collisions. Systems such as the Collision Avoidance System Testbed have shown how multiple subsystems can be combined together to achieve real-time avoidance, while other studies have explored more lightweight sensing approaches such as acoustic or vision-based detection.

From a methodological perspective, a variety of approaches have been developed for obstacle avoidance. Geometric and velocity-based methods provide relatively simple and computationally efficient solutions, while optimization-based techniques such as the Dynamic Window Approach and Optical Reciprocal Collision Avoidance offer more advanced strategies for navigating dynamic environments. Sensor technologies including LiDAR, radar, and camera systems are most commonly used for object and projectile detection, with LiDAR providing particularly strong performance due to its significant accuracy and ability to capture full-field spatial information. In addition, filtering techniques such as Extended Kalman Filters are widely used to improve state estimations by reducing the effects of noise and uncertainty.

However, despite these advancements, gaps still remain in the ability to perform real-time avoidance of fast-moving projectiles, especially for small unmanned aerial systems operating in three-dimensional environments. We address these gaps by further developing a complete and integrated collision avoidance pipeline.

IV. Design Alternatives and Trade Studies

1. PERCEPTION

Within research and industry, state-of-the-art autonomous drone perception systems commonly employ multi-sensor Simultaneous Localization and Mapping (SLAM) architectures. These systems integrate multiple sensing modalities, including depth cameras, Light Detection and Ranging (LiDAR), Inertial Measurement Units (IMU), and odometry, to develop a comprehensive understanding of the environment. However, to remain within the scope of this project and reduce system complexity, a single-sensor approach was selected for high-fidelity environmental data acquisition.

The primary sensing options evaluated were LiDAR sensors and RGB-D depth cameras. LiDAR systems generate three-dimensional point cloud data through laser pulse reflections, providing high accuracy, long detection range, and full-field spatial awareness. In contrast, depth cameras estimate object distance using structured light or infrared sensing, offering a more compact and computationally efficient solution but with limitations in range and sensitivity to environmental conditions.

A trade study was conducted to quantitatively evaluate candidate sensors using weighted performance metrics,

including accuracy and precision (40%), measurement rate (30%), and system complexity (30%), as shown in Fig. 2. The results indicate that both LiDAR and RGB-Cameras perform well. LiDAR has consistently better accuracy, precision, and data rates. The key difference is that LiDAR would be lot easier to implement into the current simulation architecture, making it the most suitable choice.

Among the LiDAR options considered, the ROCK Ultra LiDAR achieved the highest overall weighted score, driven by strong performance in both accuracy and measurement rate while maintaining moderate system complexity. Based on these results, the ROCK Ultra LiDAR was selected as the preferred sensing solution for this system.

Sensor System Trade Study							
Criteria	**Grade**	**Weight**	**XT32M/32/16 High-Precision 360° Mid-Range Lidar**	**ROCK Ultra**	**ROCK R3 Pro V2 LiDAR**	**CS30 RGB-D ToF Depth Camera for ROS, Interactive AI & SLAM**	**Intel® RealSense™ Depth Camera D435**
Accuracy and Precison	9 = Both very precise and accurate 0 = Not precise or accurate	40%	6	8	8	5	5
Measurement Rate	9 = Very quick measurement rate 0 = Slow measurement rate	30%	6	8	6	6	6
System Complexity	9 = Simple and easy integration 0 = Extremely complex and many failure points	30%	7	7	7	5	5
		100%	70.37%	85.19%	77.78%	59.26%	59.26%

Fig. 2 Sensor system trade study comparing LiDAR and RGB-D depth camera options using weighted metrics of accuracy and precision (40%), measurement rate (30%), and system complexity (30%). The ROCK Ultra LiDAR achieved the highest overall score.

Realism and the ability to implement into actual hardware are both long-term design objectives. Accordingly, commercially available sensor specifications were used to inform simulated sensor behavior. Although LiDAR was selected as the preferred sensing modality, challenges associated with integrating LiDAR data through ROS2 and AirSim necessitated refinement of the implementation approach. As a result, the final simulation utilized idealized sensor inputs with adjustable noise levels to emulate real-world measurement uncertainty. This approach preserves the relative performance characteristics identified in the trade study while enabling efficient development and validation within the simulation environment.

2. PREDICTION

When looking at UAV autonomous prediction, many various methods have been introduced. Various obstacle avoidance methods include geometric-based prediction, machine learning, and vector field analysis.[1] These are combined with various environment building methods such as image processing technology, map-based data collection, and sensor data collection. An example for geometric prediction is the 3D-SWAP algorithm, modeling dynamic objects as geometric shapes, defining safety zones around them for avoidance. [2]

Many of the prediction algorithms involve hybrid methods, such as the Dynamic Window Approach (DWA) and Optimal Reciprocal Collision Avoidance (ORCA)[3]. The DWA works by sampling possible object velocities, forming short trajectories, then scoring and choosing the best one based off of a cost function involving efficiency, obstacle distance and directional goal. A study combined this with ORCA, which uses a velocity obstacle theory combined with plane-constraints to find an optimal path with minimal velocity. Implementing both systems created an optimization which would determine available paths and find the optimal one quickly.

A notable algorithm used for quick-response obstacle tracking is a Kalman filter. The Kalman filter creates dynamic object state-estimations with implemented noisy sensor measurements. This requires initial states read by sensors or image processing such as LiDAR previously mentioned, however, it has the ability to make positional predictions under a controlled uncertainty. In a fast dynamical environment, this design seems optimal as a demonstration to build a prediction module modeling the results for this mission.

V. Methods

A. Final System Design

The final system is implemented entirely within MATLAB/Simulink and consists of perception, prediction, and planning subsystems.

1. PERCEPTION

The Simulink diagram shown in Fig. 3 illustrates the process used to apply measurement noise to the baseball's pure-state data taken from the baseball's actual simulation location. A band-limited white noise block is added to the position signal, introducing controlled noise levels (0%, 10%, and 50%) to simulate measurement uncertainty. Velocity and acceleration are unchanged as they aren't used to feed the prediction and planning systems. Noise and a 1-second delay are added to this data to simulate the effect a high-fidelity sensor may have. Increase in noise allows the system to replicate poorer sensor capabilities or conditions. Weather conditions like rain, snow, and fog can increase noise within the data acquired by sensors. Adjustability allows a simple way to mimic this effect. Using pure-state data has proven to be a simple yet elegant way to refine the downstream systems that rely on it. Using the pure state creates an extremely repeatable, predictable, and configurable perception technique that has proven crucial in enabling the speedy testing and debugging of the entire system. Long-term, adding high-fidelity sensors in a LiDAR-only or multi-sensor SLAM system will be ideal and generate the most realistic perception.

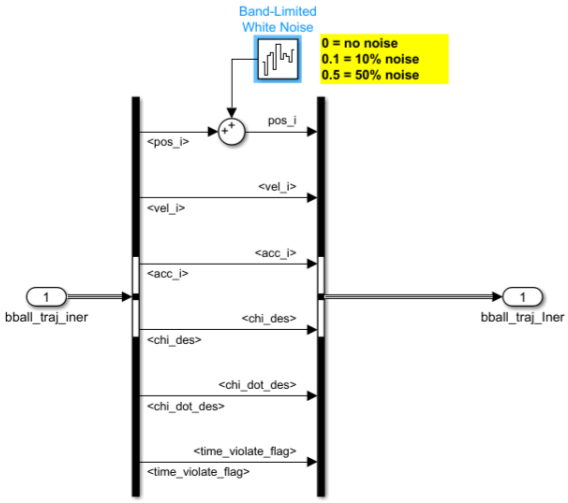
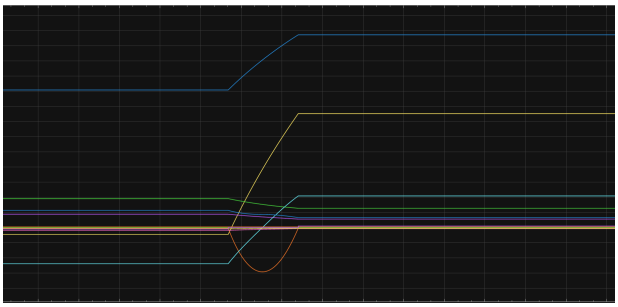
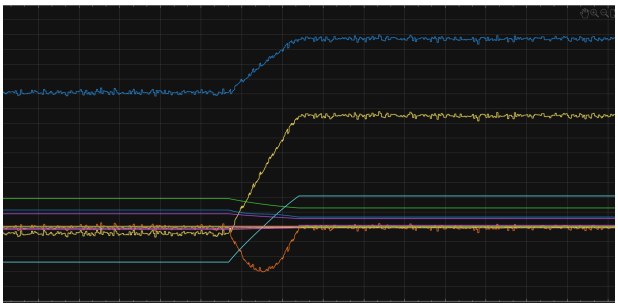


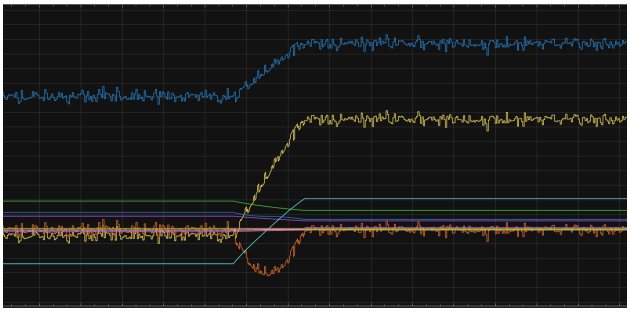
Fig. 3 Simulink implementation of measurement noise injection.



(a) 0% noise



(b) 10% noise



(c) 50% noise

Fig. 4 Comparison of system response under increasing measurement noise levels.

In addition to noise modeling, a logic-based detection block was implemented to determine whether the baseball is actively moving within the simulation. As shown in Fig. 5, the system evaluates the change in position between consecutive measurement frames. If the magnitude of the displacement exceeds a predefined threshold, the baseball is classified as moving.

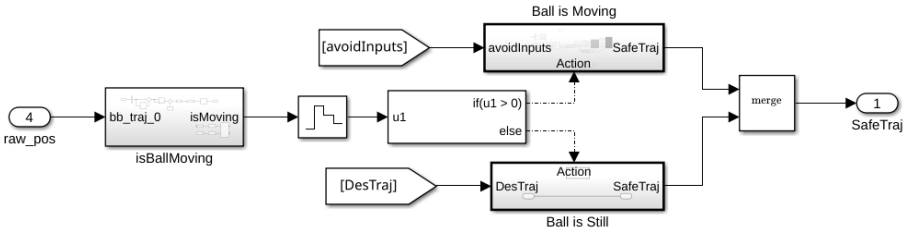


Fig. 5 Simulink logic used to determine whether the baseball is in motion. The system compares consecutive position measurements to detect movement and switches between nominal and avoidance trajectory generation accordingly.

When motion is detected, the avoidance subsystem is activated, and a safe trajectory is generated to prevent potential collision. Conversely, if the displacement between frames is negligible, the baseball is considered stationary, and the drone continues along its nominal desired trajectory. This conditional switching mechanism ensures that avoidance maneuvers are only triggered when necessary, reducing unnecessary trajectory deviations and improving overall system efficiency.

2. PREDICTION

The Prediction system in the BAM code functions with a three part program: Estimate the true ball position, estimate ball velocity, then predict the future ball trajectory. The first section of which is done via an Extended Kalman Filter (EKF). Where a normal Kalman Filter is a linear measurement state estimation algorithm, an EKF utilizes Jacobian matrices to estimate non linear approximations given data and noise estimation matrices. We utilize the estimated initial position of the ball using this algorithm, and with the proper calibration, we can essentially eliminate noise from the system. The State and Measurement Noise Covariances (Labeled Q and R), are calculated live using a separate function after being fed the current position and velocity measurements. The state process noise is calculated with a constant positional noise estimation, then a magnitude of the velocity for the velocity component, restricted to a maximum of 10%, in order to cap the EKF to realistic expectations, speeding up the convergence. The measurement noise has a similar cap, but the covariance matrix is only a 3x3 due to only position being observed. The equations used for this process are provided. Equations on the EKF itself are widly studied and will not be provided in this paper.

Process Noise (Q):

$$v_{mag} = \sqrt{v_x^2 + v_y^2 + v_z^2}, \quad v_{noise} = \max(Q_{noise} \cdot v_{mag}, q_{min}) \tag{1}$$

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & v_{noise} & 0 & 0 \\ 0 & 0 & 0 & 0 & v_{noise} & 0 \\ 0 & 0 & 0 & 0 & 0 & v_{noise} \end{bmatrix} \tag{2}$$

Measurement Noise (R):

$$\sigma_i^2 = \max((R_{noise} \cdot p_i)^2, r_{min}) \quad \text{for } i \in \{x, y, z\} \tag{3}$$

$$R = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \tag{4}$$

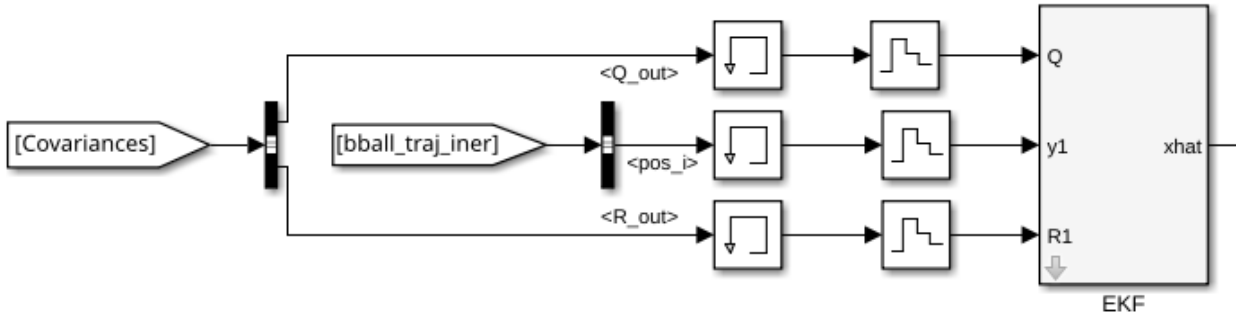


Fig. 6 EKF Block with incoming positional data (`bball_traj_iner`), and The Covariance Matrices (Covariances). The output being stabilized positional data (`xhat`)

The Estimation of the ball's velocity was done through the use of a hysteresis, where previous data points were saved and used for estimation of it's current state. This allowed us to calculate the current velocity with a high degree of certainty once the EKF had the time needed to settle. That, combined with a low Pass Filter as the algorithm was increasing in confidence, gave us very accurate velocity results when compared with truth data. The Calculation itself was a simple Polynomial Fitting algorithm, in which we had separate estimations for the X-Y axes, and the Z, which was a second degree Polynomial due to gravity. Finally, the prediction phase works via Standard Euler Equations and kinematics, predicting based on a timestep, with the negative acceleration due to drag also estimated to prevent large error. The drag coefficient was calculated through the use of our knowledge of our object. This would change based on the incoming object. All future predicted points were combined into an outgoing array, bb_traj.

Acceleration (Drag + Gravity):

$$\mathbf{a}_{drag} = -C_d \cdot \|\mathbf{v}\| \cdot \mathbf{v} \tag{5}$$

$$\mathbf{a}_{total} = \mathbf{g} + \mathbf{a}_{drag} \tag{6}$$

Forward Euler Update:

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \mathbf{a}_{total} \cdot \Delta t \tag{7}$$

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{v}_{k+1} \cdot \Delta t \tag{8}$$

3. PLANNING (ASSESSMENT

The Assessment section to the planner requires that the future trajectories of both the drone and the ball be analyzed in order to predict their potential for collision. We had already predicted the path of the ball by this point, but in order to get the future positions and velocities of the drone, we take advantage of the method in which the drone's path was programmed. By utilizing Bezier curves, which are parametric functions which compute smooth scalable curves using anchor points and intermediate control points, we can accurately predict the direction the drone is moving. This process would, of course, be vastly different for a live input drone, but because the purpose of BAM is to add autonomy to an already autonomous drone, this is an acceptable methodology. Once both future trajectories are found and sized equally, the relative trajectories are calculated. We calculate the norm of the relative position to find all future separations between the two.

$$d_{sep} = \|\mathbf{p}_{rel}\| = \sqrt{\sum \mathbf{p}_{rel}^2} \tag{9}$$

(10)

We can also more quickly determine if we should return to nominal pathing after avoidance by determining if the bodies are converging or diverging by taking the dot product between the position and velocity, which gives the angle between the position and velocity vectors. If the angle is less than zero, the objects are approaching each other, and vice versa for diverging.

$$\text{Converging if:} \quad \mathbf{p}_{rel} \cdot \mathbf{v}_{rel} < 0 \tag{11}$$

$$\text{Diverging if:} \quad \mathbf{p}_{rel} \cdot \mathbf{v}_{rel} \geq 0 \tag{12}$$

After calculating the future states of the ball, we follow a multi-factor flagging system between the time available for maneuvering after a delay, the distance needed to move, and the max lateral displacement given drone requirements before t_cpa (time to closest approach). If the bodies are converging, and the distance needed to move is greater than zero, and the distance available is less than the distance needed, then the avoidance flag is triggered. If the bodies are diverging, and the current separation is greater than the allowed distance, we consider the ball avoided, and we attempt to return to nominal.

4. PLANNING (AVOIDANCE

The final section of the planner takes the input flag, then calculates the best method for avoidance by generating a lateral avoidance offset and return to nominal pathway. First, the direction and magnitude of force needed is calculated. We find the unit heading of the drone, and use that to project the balls incoming position vector perpendicular to the path of the drone. We take a candidate direction and remove the component that lies along the drones unit heading. By doing so, we force the resulting direction to lie on the plane perpendicular to the drones flight path, which is one of the variable methods we have taken in avoiding the drone during many iterations of testing. If this is found to be to pointing towards the ball, we flip the sign.

1. Heading Unit Vector

$$\mathbf{e}_v = \text{normalize}(\mathbf{v}_{body}) \tag{13}$$

2. Perpendicular Candidate (Vector Rejection)

$$\mathbf{n}_{raw} = \mathbf{v}_{rel} - (\mathbf{v}_{rel} \cdot \mathbf{e}_v)\mathbf{e}_v \tag{14}$$

$$\mathbf{n}_{unit} = \frac{\mathbf{n}_{raw}}{|\mathbf{n}_{raw}|} \tag{15}$$

3. Direction Flipping (Avoidance)

$$\mathbf{n}_{final} = \begin{cases} -\mathbf{n}_{unit} & \text{if } \mathbf{n}_{unit} \cdot \mathbf{p}_{rel} > 0 \quad \mathbf{n}_{unit} \\ \text{otherwise} \end{cases} \tag{16}$$

To produce the Quintic Polynomials, we define some coefficients that are calibrated to our use case, then constrain the velocity, acceleration, and jerk based on the timing available for our avoidance. We also include state scaling (α) which represents normalized position, velocity, and acceleration that we previously calculated. This creates an extremely smooth transition in and out of the desired trajectory, similar to a low pass filter, but with a stronger basis in dynamics.

1. Polynomial Basis (Quintic)

$$\alpha(s) = 10s^3 - 15s^4 + 6s^5 \tag{17}$$

$$\dot{\alpha}(s) = 30s^2 - 60s^3 + 30s^4 \tag{18}$$

$$\ddot{\alpha}(s) = 60s - 180s^2 + 120s^3 \tag{19}$$

2. Dynamic Scaling

$$D(t) = D_{start} + \Delta D \cdot \alpha(s) \tag{20}$$

$$\dot{D}(t) = \frac{\Delta D}{T}\dot{\alpha}(s), \quad \ddot{D}(t) = \frac{\Delta D}{T^2}\ddot{\alpha}(s) \tag{21}$$

3. Time Constraint Selection

$$T_{req} = \max\left(T_{min}, \frac{1.875\Delta D}{v_{max}}, \sqrt{\frac{5.77\Delta D}{a_{max}}}, \sqrt[3]{\frac{60\Delta D}{j_{max}}}\right) \tag{22}$$

The final section of this algorithm is to swap cases based on the flag state and what case it is currently in. The states we can navigate between are Nominal, Blend Out, Hold Offset, and Blend Back. When the state is switched to trigger the flag, we transition from nominal to blend out. If it is found that the threat is persistent, we can increase the offset. If the threat has not worsened, but we are still within the flag state, we hold the offset until it gets worse or the flag clears. Once the flag clears, from either the blend out or the holding case, we blend back in, which eventually returns us to normal.

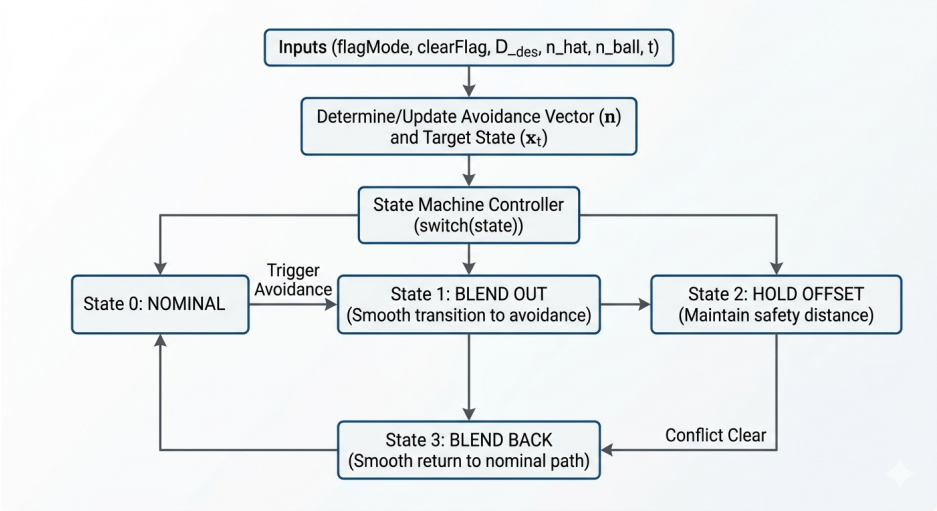


Fig. 7 Avoidance Pathway based on cases

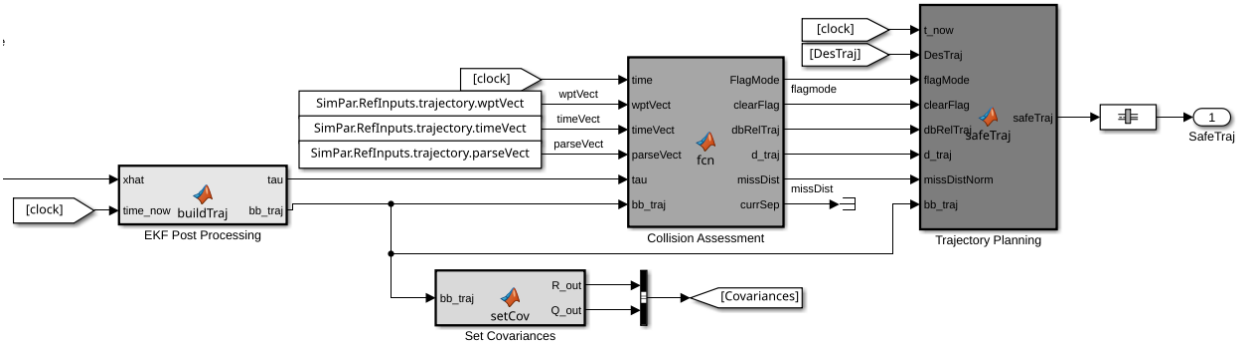


Fig. 8 Full bus post-EKF including Prediction, Covariance generation, Assessment, and Avoidance (Left to Right).

B. Test Procedures

Testing was conducted using a MATLAB batch script that simulated 100 matched own-ship and baseball trajectory pairs from the BAM Challenge Problem data sets. For each run, the script loaded the corresponding drone and baseball trajectories, updated the trajectory-dependent simulation inputs, initialized the EKF using the baseball launch position and velocity, and ran the full Simulink collision avoidance model without ROS2/Unreal visualization. After each simulation, the minimum drone-to-baseball separation distance and time of closest approach were computed. A run was classified as successful if the minimum separation distance exceeded the 3.0 m safety radius; otherwise, it was classified as a collision or near miss. The script also recorded simulation errors and generated aggregate plots showing success rate, closest approach distance, and representative trajectory outcomes.

VI. Data Analysis and Comparison to Literature

A. Kalman Filter Noise-Rejection Verification

To isolate the performance of the prediction module from the full avoidance system, the Kalman filter output was compared under increasing measurement-noise levels. Figure 9 shows the filtered ball-to-drone separation response for the same representative trajectory under multiple added-noise conditions. As expected, the zero-noise case produces a smooth separation history, while the higher-noise cases introduce visible measurement fluctuations. However, the

filtered trajectory preserves the same overall trend: the ball approaches the drone, reaches minimum separation near closest approach, and then moves away. This indicates that the Kalman filter is performing its intended role of reducing the effect of noisy measurements and maintaining a usable estimate for downstream prediction.

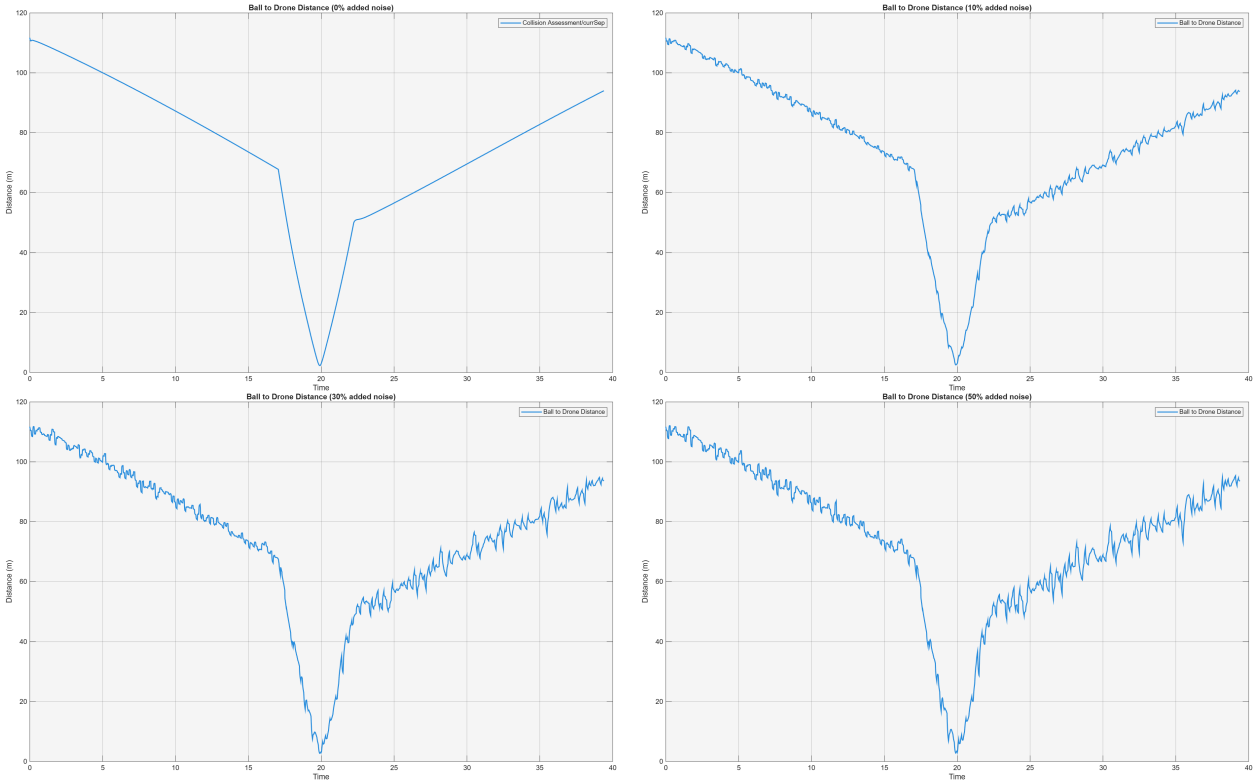


Fig. 9 Kalman filter response under increasing measurement-noise levels. The plots compare the ball-to-drone separation trend for representative trajectories with added sensor noise. Although higher noise levels introduce visible fluctuations, the filtered response preserves the overall approach, closest-approach, and separation behavior needed by the downstream prediction and planning modules.

These Kalman filter results also clarify why the trajectory-pair batch tests alone do not conclusively indicate whether the filter is working. The batch tests evaluate the full collision-avoidance pipeline, including prediction, collision assessment, avoidance planning, and drone control response. Therefore, failures or extreme deviations in the batch results cannot be attributed solely to the Kalman filter. In this project, the main avoidance issue appears to come from hypersensitive control behavior and the Planning module's response after avoidance is triggered, rather than from the filter's ability to smooth noisy ball-position measurements. Since the Kalman filter visibly reduces measurement noise and preserves the useful trajectory trend, future work can focus on improving the avoidance planner, control tuning, and return-to-nominal behavior while keeping the filtering approach as a validated part of the system.

B. Trajectory Pair Batch Test Results

To evaluate system performance under realistic sensing uncertainty, the final collision avoidance architecture was tested using simulated baseball trajectories with 10% added measurement noise. A total of 95 trajectory pairs were evaluated. Performance was assessed based on avoidance success rate and minimum separation distance between the baseball and drone. Representative trajectory visualizations were also selected to illustrate the closest successful avoidance case, the tightest dodge, and the case with the greatest achieved separation.

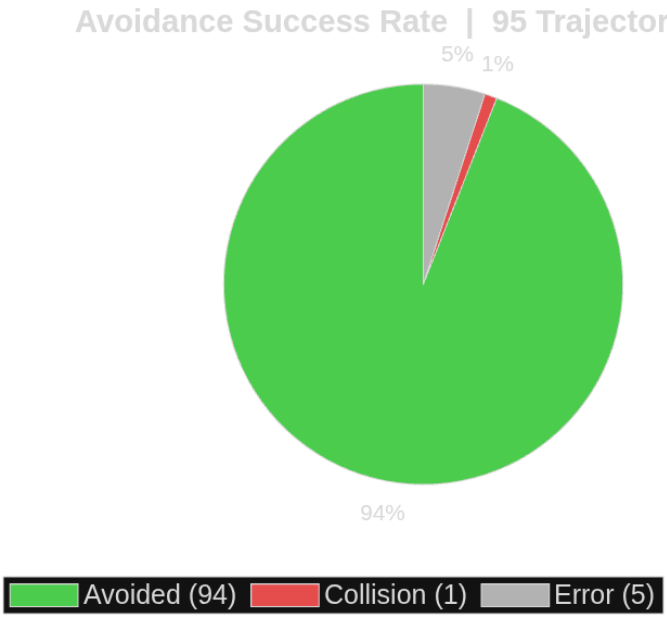
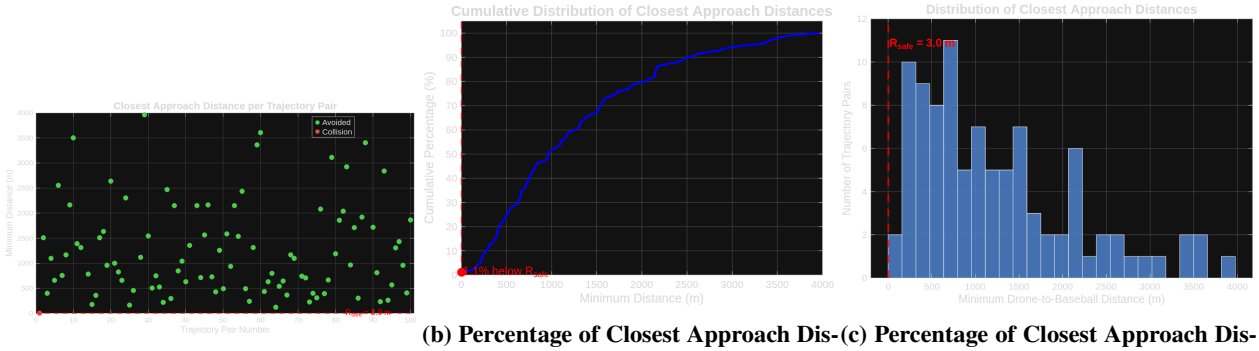


Fig. 10 Avoidance outcome distribution for 95 simulated trajectory pairs with 10% added measurement noise. Of the total runs, 94% resulted in successful avoidance, 1% resulted in collision, and 5% terminated in simulation or processing error.

Figure 10 shows the overall avoidance performance for the 10% noise case. The system successfully avoided the baseball in 94 out of 100 valid trajectory pairs, corresponding to a 94% success rate. One case resulted in a collision, while five runs produced errors and were excluded from successful avoidance. These results indicate that the prediction and planning framework remained highly effective under moderate measurement uncertainty, although some sensitivity to noisy inputs and edge-case failures was still present.

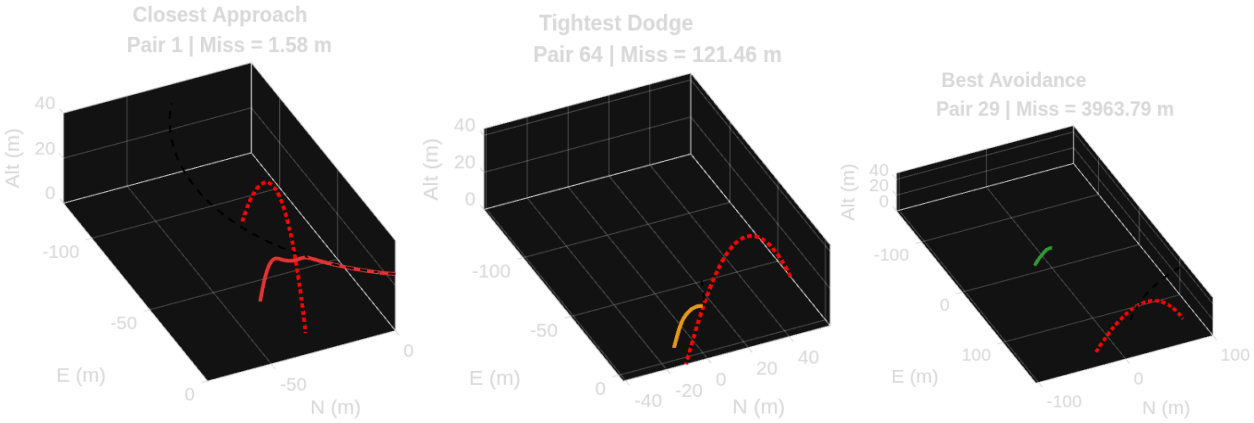


(a) Dot Plot of Closest Approach Distances

(b) Percentage of Closest Approach Distances From Minimum to Maximum

(c) Percentage of Closest Approach Distances From Minimum to Maximum

Fig. 11 Collective trajectory pair outcomes for the 10% noise test case. The graphs show various views of the 95 trajectory pairs, comparing case-by-case, minimum to maximum, and a distribution graph.



(a) Closest approach **(b) Tightest dodge** **(c) Best avoidance**

Fig. 12 Representative trajectory pair outcomes for the 10% noise test case. These plots illustrate a near-minimum safe separation event, a tight but successful avoidance maneuver, and a case with very large achieved miss distance.

Figure 12 presents representative three-dimensional trajectory comparisons for the 10% noise case. In Fig. 12a, Pair 1 represents the closest successful pass, with a miss distance of 1.58 m. This case demonstrates that the system was capable of generating avoidance behavior even when the available maneuver margin was small. In Fig. 12b, Pair 64 shows a successful but more aggressive avoidance trajectory, with a miss distance of 121.46 m. Finally, Fig. 12c shows Pair 29, which produced the largest separation distance of 3963.79 m, indicating that some avoidance responses were highly conservative relative to the incoming baseball trajectory.

These results suggest that the planner was generally successful at preventing collisions under moderate noise conditions, but the achieved miss distance varied significantly from case to case. The smaller miss-distance cases demonstrate the minimum performance capability of the system, while the larger miss-distance cases suggest that the planner may at times overreact, producing unnecessarily large deviations from the nominal path. This behavior is consistent with the broader observation that the system prioritized avoidance success over trajectory recovery or maneuver efficiency.

VII. Results

(a) Beginning of flight

(b) Ball spawns in simulation

(c) Drone detects ball and maneuvers

(d) Maneuver somewhat complete

(e) System freak out

(f) Crash

Fig. 13 Here is trajectory pair 29.

The images found in Figure 13 show a great example of how the ball manages to detect the baseball, assess collision, and attempt an avoidance maneuver. This freak out is the case with all "Avoided" cases shown in Figure 10, so technically, no avoidance was done in the intended manner.

VIII. Suggestions For Future Work

Future work for this project should focus on improving both the simulation framework and the transition toward more realistic sensing and hardware implementation. Within the current developed MATLAB/Simulink environment, further refinement of the avoidance planner is definitely needed, particularly in calibrating avoidance maneuvers based on the physical movement limitations of the drone. Our current implementation generates several avoidance trajectories that are effective in simulation, but additional tuning is required to ensure that these maneuvers remain feasible under realistic dynamic constraints. Constraints such as maximum acceleration, velocity limits, and actuator response times. In addition, improvements specifically to the trajectory prediction model are necessary to reduce exaggerated or overly aggressive predictions of the baseball's motion. By enhancing the prediction accuracy, this will lead to a more reliable collision assessment and will improve the triggering of system states, such as avoidance, nominal flight, and recovery. Expanding the number of simulation runs to include a larger portion of the available test cases will also provide a more comprehensive validation of system performance and help identify edge cases where the current approach may fail.

Beyond our current simulation environment, our future work should emphasize improving the system robustness under uncertainty and moving toward higher-fidelity sensing and simulation testing. Incorporating more advanced trajectory optimization methods that account for dynamic constraints and uncertainty, will improve the reliability of avoidance maneuvers. As well as, reintegrating high-fidelity sensor simulation through platforms such as Unreal Engine, Colosseum/AirSim, and ROS2 would significantly enhance the realism of the project by providing more representative sensor data. Investigating alternative perception approaches, such as a visual simultaneous localization and mapping (VSLAM) system that leverages depth cameras in combination with LiDAR for environmental mapping, could provide a more robust and scalable sensing solution.

Ultimately, a great direction for future work is making the transition from simulation to hardware implementation. This includes integrating physical sensors such as LiDAR and onboard cameras with compatible processing hardware and deploying the system on an actual multirotor platform. We believe Real-world flight testing would allow for further validation of the system under practical conditions, including environmental disturbances, sensor limitations, and real-time computational constraints. Advancing toward hardware implementation will be essential for demonstrating the true effectiveness and applicability of the collision avoidance system beyond controlled simulation environments.

IX. Conclusion

The program successfully demonstrated that it could detect and attempt to avoid the baseball on its path. However, it was not able to successfully return to its original desired path. Although there is clear future work ahead, our project successfully demonstrated the capabilities of our team in developing an end-to-end autonomous collision avoidance system within this simulated environment.

Future work must integrate the high-fidelity AirSim sensors into the system. This will allow for more sophisticated perception system through computer vision and machine learning. As mentioned before, future work must also figure out why the drone does not return to its original trajectory and fix it, then ensure the drone path deviation is also at a reasonable minimum.

This project met its primary goal of demonstrating a working collision avoidance system, however it represents an early-stage solution rather than a fully optimized/production-ready system.

Acknowledgments

The Elite Ball Knowledge Team would like to thank Professor Roni Goldshmid for her support and guidance throughout the semester. We would also like to thank NASA Langley Research Center (LaRC) Dynamics Systems and Control Branch (D-316) for making this challenge possible. We would like to thank Master's students Sean Gidley and Tyler Felgenhauer for their support. As well as, we would like to thank our GWAR advisor Jacon Hubbard for the constructive feedback throughout the semester.

References

[1] Shi, P., "A literature review on obstacle avoidance algorithms and attitude control of unmanned aerial vehicles," *Journal of Computing and Electronic Information Management*, Vol. 15, No. 3, 2024, pp. 16–19. URL https://drpress.org/ojs/index.php/jceim/article/view/28244.

[2] Chang, X., Wang, J., Li, K., Zhang, X., and Tang, Q., "Research on multi-UAV autonomous obstacle avoidance algorithm integrating improved dynamic window approach and ORCA," *Scientific Reports*, Vol. 15, 2025, p. 14646. https://doi.org/10.1038/s41598-025-99111-8, URL https://www.nature.com/articles/s41598-025-99111-8.

[3] Merei, A., Mcheick, H., Ghaddar, A., and Rebaine, D., "A survey on obstacle detection and avoidance methods for UAVs," *Drones*, Vol. 9, No. 3, 2025, p. 203. URL https://www.mdpi.com/2504-446X/9/3/203.