# A Vision-Based Localization & Path Planning Robot System for Autonomous Emergency Response

Huaze Liu
*Harvey Mudd College*
Claremont, USA
hualiu@g.hmc.edu

Jessica Liu
*Harvey Mudd College*
Claremont, USA
jesliu@g.hmc.edu

## I. Abstract & Introduction

### A. Motivation & Significance

In the aftermath of natural disasters such as earthquakes and wildfires, search and rescue teams must navigate unstable, hazardous environments. This not only endangers human responders but also delays critical interventions. To address these challenges, we present a dual-robot framework: an *explorer* robot that performs vision-based detection and localization, aiming for rapid information collection of victims; and a *rescuer* robot that computes and follows efficient, safe paths to each detected target. The explorer is designed to be lightweight and precise, using HSV-based object detection, PD control, and an State Estimation filter for robust tracking. Upon identification, it records GPS coordinates and transmits them to the rescuer, which then applies A* path planning to reach each victim. By automating detection, localization, and navigation, our approach reduces human risk and accelerates rescue operations while maintaining sub-meter targeting accuracy.

### B. Related Work

Over the past decade, there has been significant interest in rescue robots, leading to numerous innovative developments in areas such as SLAM, robotic operation, and detection techniques. In this section, we highlight several closely related works and discuss relevant comparisons.

In our project, we adopt a computer vision-based approach that uses a raw RGB camera as the primary input for our search algorithm. This approach is consistent with the vision-based strategies outlined by Shahria et al. in their comprehensive review of robotic applications and system components [1]. Their proposed framework for vision-based systems involves the following sequence: RGB image acquisition, followed by image segmentation or filtering, then action planning based on computational analysis and control, and finally, execution. This structured pipeline closely mirrors the framework we implemented throughout our project. In the search phase of rescue operations, effective navigation is critical—particularly in low-light or dark environments. Daftry et al. [2] proposed a robust vision-based system for autonomous night-time navigation and landing using micro aerial vehicles (MAVs). Their approach utilizes thermal-infrared cameras to reconstruct a 3D map of the target area, allowing for accurate localization even in complete darkness. In contrast, our system relies on a 2D RGB camera, which reduces hardware complexity and computational demands but depends on adequate lighting conditions. While this design supports lightweight and cost-efficient deployment, it limits applicability in real-world scenarios where lighting may be unpredictable or insufficient. However, our goal for the final project is to proposed a possible framework.

Many studies on rescue robotics explore various modes of operation. One commonly adopted system is air-ground collaboration, which combines aerial drones and ground robots to carry out coordinated tasks. For instance, a 2024 study titled *Air-Ground Collaborative Robots for Fire and Rescue Missions: Towards Mapping and Navigation Perspective* presents an innovative approach to air-ground rescue operations [3]. This is closely aligned with our project, which similarly employs at least two robots to perform distinct tasks—searching and rescuing. Aerial robots, such as drones, can perform initial searches more quickly and efficiently due to their broader field of view in the air. However, our operation involves two ground robots. We chose not to include aerial robots because, in real-world rescue scenarios, drones often face limitations. For example, they may struggle to enter smoke-filled buildings or navigate environments with numerous obstacles that block or restrict their movement. While ground robots are usually more flexible in searching.

The second major component of our proposed framework is path planning, with a particular emphasis on the algorithm selection. Both Fransen's study, "Efficient Path Planning for Automated Guided Vehicles Using A (Astar) Algorithm Incorporating Turning Costs in Search Heuristic"* [4], and Krishnaswamy's "A Comparison of Efficiency in Pathfinding Algorithms in Game Development" [5] highlight the A* algorithm as one of the most efficient and fastest methods for pathfinding. Given the critical importance of speed and efficiency in rescue operations, we have chosen to implement A* as the core algorithm in our final project.

## II. Methodology

Our framework aims to enable a mobile robot to autonomously search for, approach, and sequentially "collect" multiple target objects within an environment, utilizing vision-based detection and state estimation for robust control. With the information of the number of the target objects and their locations, the second rescue robot will plan an efficient path to sequentially approach each of them. For this project, we implement and evaluate this system within the Webots robotics simulator [6]. The task involves a single TurtleBot3 Burger robot equipped with a forward-facing RGB camera and a GPS sensor operating in a planar arena containing multiple identical yellow spheres (target objects). This setup allows us to focus on the core perception, estimation, control, multi-target sequencing logic, and path planning. This section details the system architecture, object detection pipeline, the state estimation filters employed, the control strategy, the logic for multi-ball collection, and path planning algorithm 1.

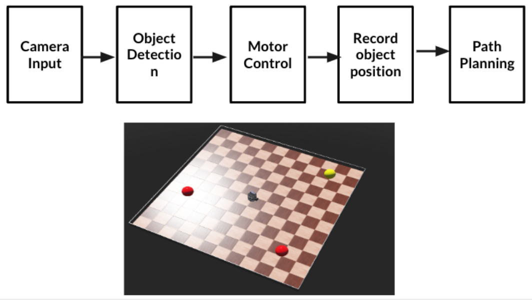


Fig. 1. System Overview

### A. System Architecture Overview

For our first robot, it operates in a continuous loop involving perception, state estimation, decision-making, and control (summarized in Figure 2. The camera captures images processed by the Object Detection module to identify yellow target balls using color thresholding. The primary measurement derived is the horizontal pixel error between the largest detected ball's centroid and the camera's image center.

This noisy measurement is fed into State Estimation Filters. In this project, we examine and test two commonly-used filters: Extended Kalman Filter (EKF) [7] and Unscented Kalman Filter (UKF) [8]. Both filters estimate the ball's tracking state (pixel error, error rate, bearing angle) based on the visual measurements and a motion model.

The Control & Avoidance Logic uses the filtered state to generate wheel velocity commands. It incorporates GPS data to manage multi-ball collection, preventing re-collection of already visited targets and implementing basic avoidance of cleared areas during approach maneuvers. A Termination Condition monitors the time since the last successful collection to halt the robot when no further targets are found.

For our second robot, the primary task is to receive GPS data from the first robot and compute the most efficient and shortest path using a path planning algorithm. In our project, we have chosen to use A* as the core path planning method.

### B. Object Detection Method

Target object (yellow ball) detection utilizes color thresholding in the HSV (Hue, Saturation, Value) color space, known

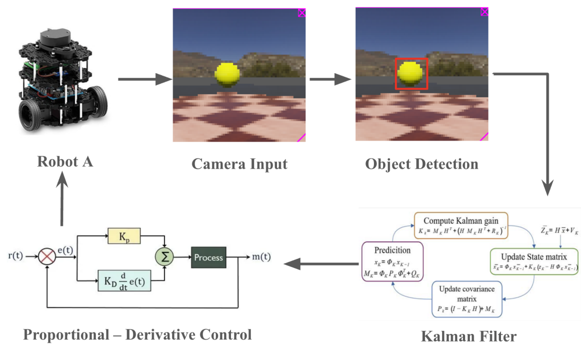

Robot A
Camera Input
Object Detection
r(t)
e(t)
$K_p$
$\Sigma$
$K_D \frac{d}{dt} e(t)$
Process
m(t)
Proportional – Derivative Control
Compute Kalman gain
$K_k = M_K H^T + (H M_K H^T + R_K)^{-1}$
$\bar{Z}_K = H\bar{x} + V_K$
Predicition
$x_K = \Phi_K x_{K-1}$
$M_K = \Phi_K P_K \Phi_K^T + Q_K$
Update State matrix
$\bar{x}_K = \Phi_K \bar{x}_{K-1} + K_K(z_K - H\Phi_K \bar{x}_{K-1})$
Update covariance matrix
$P_k = (I - K_K H) + M_K$
Kalman Filter

Fig. 2. Detection and Localization Framework

for its relative robustness to illumination changes compared to RGB [9]. The input BGR image from the camera is first converted to HSV.

We then apply a binary threshold based on experimentally determined ranges for yellow under the simulation's lighting conditions:

$$M(x,y) = \begin{cases} 1, & H(x,y) \in [20^\circ, 35^\circ], \\ & S(x,y) \geq 70,\ V(x,y) \geq 60, \\ 0, & \text{otherwise}. \end{cases} \tag{1}$$

To mitigate noise resulting from thresholding (e.g., small erroneous patches or holes in the detected ball mask), morphological opening followed by closing is applied using a $7 \times 7$ elliptical structuring element $B$:

$$M_{\text{open}} = (M \ominus B) \oplus B, \quad M_{\text{clean}} = (M_{\text{open}} \oplus B) \ominus B.$$

Finally, external contours are extracted from the cleaned binary mask $M_{\text{clean}}$ using `cv2.findContours`. Contours with an area $A$ less than $A_{\min} = 30\,\text{px}$ are discarded. The centroid $(c_x, c_y)$, calculated using image moments (`cv2.moments`) of the largest remaining contour, is used. The horizontal pixel error $e_{\text{meas}} = c_x - (\text{ImageWidth}/2)$ serves as the primary input measurement for the state estimation filters.

### C. State Estimation Filters & Control

To handle measurement noise and provide smoother estimates for control, we employ state estimation filters. The system implements both an EKF and a UKF to drive the control loop.

*1) State Representation:* The state vector for both filters is defined as:

$$\mathbf{x}_k = \begin{bmatrix} e_k \\ \dot{e}_k \\ \theta_k \end{bmatrix},$$

where $e_k$ is the horizontal pixel error from the image center to the ball's centroid, $\dot{e}_k$ is its discrete-time rate of change, and $\theta_k$ is the estimated bearing angle to the ball relative to the camera's optical axis.

*2) Motion Model (for the both filters):* We assume a simple constant-velocity model for the pixel error dynamics between steps, perturbed by noise. The state propagates according to the linear process model:

$$\mathbf{x}_{k+1} = \mathbf{F}\,\mathbf{x}_k + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}),$$

where $\Delta t$ is the simulation timestep, and the state transition matrix $\mathbf{F}$ and process noise covariance $\mathbf{Q}$ are:

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.001 \end{bmatrix}.$$

In our case, the process noise Q reflects assumed small uncertainties in the velocity and angle dynamics between timesteps.

*3) Measurement Model (for the both filters):* The measurement vector consists of the directly observed pixel error and the bearing angle calculated from it:

$$\mathbf{z}_k = \begin{bmatrix} e_{\text{meas}} \\ \theta_{\text{meas}} \end{bmatrix} \approx h(\mathbf{x}_k) + \mathbf{v}_k = \begin{bmatrix} e_k \\ \arctan(e_k/f) \end{bmatrix} + \mathbf{v}_k,$$

where $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R})$, $f$ is the camera's focal length (77.25 pixels), and $\mathbf{R}$ is the measurement noise covariance:

$$\mathbf{R} = \begin{bmatrix} 100 & 0 \\ 0 & 0.01 \end{bmatrix}.$$

In our case, the measurement noise R assumes higher variance (100) for the direct pixel error measurement due to detection noise and contour centroid instability, and lower variance (0.01) for the derived angle measurement.

*4) Extended Kalman Filter:* The EKF handles the non-linear arctan in the measurement function via first-order Taylor expansion (linearization). It uses the measurement Jacobian matrix $\mathbf{H}$ evaluated at the predicted state $\hat{\mathbf{x}}_{k|k-1}$ during the update step:

$$\mathbf{H}_k = \frac{\partial h}{\partial \mathbf{x}}\bigg|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k-1}} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The standard EKF prediction and update equations [7] are used, yielding the state estimate $\hat{\mathbf{x}}_k^{\text{EKF}} = [\hat{e}_k, \widehat{\dot{e}}_k, \hat{\theta}_k]^T$ which drives the robot's control.

*5) Unscented Kalman Filter (UKF):* Implemented for comparison using the `filterpy` library [10]. The UKF employs the unscented transform with MerweScaledSigmaPoints (parameters $\alpha = 0.1, \beta = 2.0, \kappa = 0$) to directly propagate state uncertainty through the non-linear measurement function $h(\mathbf{x}) = [x_1, \arctan(x_1/f)]^T$ without requiring explicit linearization or Jacobian matrices. It uses the same linear motion model function $f(\mathbf{x}, dt) = \mathbf{F}\mathbf{x}$ and the same $\mathbf{Q}$ and $\mathbf{R}$ matrices as the EKF for fair comparison. Its state estimate $\hat{\mathbf{x}}_k^{\text{UKF}}$ is logged.

*6) Control Law:* The robot's motion is governed by the EKF's filtered state estimate $\hat{\mathbf{x}}_k^{\text{EKF}}$ and UKF's filtered state estimate $\hat{\mathbf{x}}_k^{\text{UKF}}$.

- Turning: If a valid ball is detected but the filtered error magnitude $|\hat{e}_k|$ exceeds the threshold, a Proportional-Derivative (PD) control command is generated for turning:
$$u_k = K_P\,\hat{e}_k \;+\; K_D\,\widehat{\dot{e}}_k$$
. The output $u_k$ is clamped to the range $[-\text{SEARCH\_SPEED}, +\text{SEARCH\_SPEED}]$ and applied differentially to the wheels ($v_{\text{left}} = u_k$, $v_{\text{right}} = -u_k$).
- Approach: If $|\hat{e}_k| <$ ERROR_THRESHOLD and the ball's area $A <$ SAFE_AREA_THRESHOLD, the robot drives straight forward. This phase includes the cleared area avoidance check (Section II-D).
- Searching: If no valid ball is detected , the robot spins in place at $\pm$SEARCH_SPEED in a randomly chosen direction.

### *D. Multi-Target Sequencing, Avoidance, and Termination*

To enable the robot to collect multiple targets sequentially while avoiding redundant efforts, several logic components are integrated into the control loop:

*1) Collection Memory and Verification:* The system maintains a list of the world coordinates (obtained via GPS) where targets have previously been successfully collected. Upon reaching a potential target (i.e., centered in view and estimated to be sufficiently close based on visual cues like area), the robot performs a final proximity check. It compares its current GPS location against all previously recorded collection locations. If it finds itself within a small, predefined distance threshold of any known collected location, it presumes this is a target it has already processed. In this case, collection is aborted. If the proximity check confirms the target is likely new, the collection proceeds: the current GPS location is added to the memory of collected locations, and a timer tracking the time of the last successful collection is updated.

*2) Disengagement Maneuver:* Whether a collection attempt is successful (a new GPS location is stored) or aborted (due to proximity to an already collected location), the robot executes a standardized disengagement maneuver. This typically involves reversing direction for a short period and then rotating approximately 180 degrees, facilitating the search for subsequent, distinct targets.

*3) Cleared Area Avoidance:* To prevent the robot from inefficiently re-entering areas it has already cleared during its approach to other potential targets, a proactive avoidance mechanism is employed. While driving forward towards a centered target, the robot continuously monitors its GPS position relative to all previously collected locations. If it enters a larger, predefined "avoidance radius" surrounding any collected location, the forward approach is immediately halted. An evasive maneuver (e.g., stopping and re-initiating a search scan, or a specific turning pattern) is triggered to redirect the robot away from the cleared zone before it continues its operation.

*4) Task Termination Condition:* To ensure the robot eventually stops its mission without requiring prior knowledge of the total number of targets, a timeout heuristic based on collection activity is used. The system tracks the elapsed time since the last target was successfully collected and recorded. If this time exceeds a specified maximum duration, the robot concludes that no further reachable targets are likely to be found and terminates its operational loop.

### *E. Path Planning*

To compute the optimal path through the environment, we used the A* search algorithm, a heuristic-based pathfinding method that balances path cost and estimated distance to the goal. A* maintains a priority queue of nodes based on the evaluation function:

$$f(n) = g(h) + h(n)$$

where $g(n)$ is the the cost from the start node to node $n$, $h(n)$ is the heuristic estimate of the cost from node $n$ to the goal (we used Euclidean distance $h(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2}$) This algorithm expands based on the lowest $f(n)$ by updating the lowest cost $f(n)$ of neighbors and parents nodes until all targets are reach.

### *F. Algorithm Flow Summary*

In summary, the robot initializes by starting a random search spin. The main control loop continuously performs state prediction (EKF and UKF), acquires and processes camera images to detect the largest valid yellow ball contour, and performs state updates. If a ball is detected, the EKF-based PD controller turns the robot to center it. Once centered, the robot checks if its path is clear of previously collected areas; if the path is clear, it drives forward. If it enters an avoidance zone, it performs an evasive maneuver. If it reaches the target proximity, it performs a final GPS proximity check. If the ball is verified as new, it is collected (GPS logged, collection timer reset), and a post-collection maneuver is executed. If it's identified as already collected (either during approach or final check), an appropriate maneuver moves the robot away. If no valid ball is detected, or after a maneuver, the robot resumes its search spin. Once the first bot gathers all the GPS coordinates of the targets, the second bot will calculate the shortest path using A* path planning algorithm as described in section E. Refer to Algorithm 1 for a summary of the system framework.

## III. Experiments

In this section, we are going to detail the experimental setup, procedures, and evaluation metrics used to validate the proposed ball detection, multi-ball collection system, compare the performance of the EKF and UKF for state estimation, and path planning within this framework.

**Algorithm 1** Sequential Yellow Ball Tracking with EKF and UKF

```
0: Initialize EKF/UKF state $\mathbf{x}, \mathbf{P}$
0: Initialize collected_gps_list $\leftarrow \emptyset$
0: while robot.step() do
0:   ####Prediction ####
0:   $(\mathbf{x}, \mathbf{P}) \leftarrow$ predict$(\mathbf{x}, \mathbf{P}, \mathbf{F}, \mathbf{Q})$
0:   ####Detection ####
0:   $(\mathbf{z}$, area$) \leftarrow$ detectBall$(I)$
0:   if no detection then
0:     rotateSearch()
0:     continue
0:   end if
0:   ####Update ####
0:   $(\mathbf{x}, \mathbf{P}) \leftarrow$ update$(\mathbf{x}, \mathbf{P}, \mathbf{z}, \mathbf{H}, \mathbf{R})$
0:   $\hat{e} \leftarrow \mathbf{x}[0]$; $\widehat{\dot{e}} \leftarrow \mathbf{x}[1]$ {Extract filtered state}
0:   ####Control ####
0:   if $|\hat{e}| < E_{\max}$ and area $< A_{\max}$ then
0:     driveForward()
0:   else if $|\hat{e}| < E_{\max}$ and area $\geq A_{\max}$ then
0:     $\mathbf{p}_{\text{curr}} \leftarrow$ getGPS()
0:     Append $\mathbf{p}_{\text{curr}}$ to collected_gps_list
0:     performManeuver()
0:   else
0:     rotatePD$(\hat{e}, \widehat{\dot{e}})$
0:   end if
0: end while
0: ####Path Planning #### =0
```

### *A. Experimental Setup*

*1) Hardware and Software Platform:* All experiments were conducted within the Webots robotic simulator (Version R2025) The robot model used was the standard TurtleBot3 Burger PROTO, equipped with a forward-facing RGB Camera sensor and a GPS sensor node. The simulation environment consisted of a planar arena containing three yellow spheres (target objects) placed at known locations.

The robot controller was implemented in Python (Version 3.11) using the Webots `controller` API. Key libraries utilized include `OpenCV` for image processing and contour detection, `NumPy` for numerical operations and state/matrix representations, and `filterpy` [10] for the UKF implementation.

*2) Key Experimental Parameters:* The core parameters governing the robot's behavior and the filters' operation were configured as follows, summarized in Table I. These values were determined through initial tuning and remained consistent across the comparative experiments unless otherwise noted.

### *B. Experiments Conducted*

The primary goal was to evaluate the system's ability to sequentially collect multiple targets, compare the filtering performance of the EKF and UKF under identical conditions,

TABLE I
KEY EXPERIMENTAL PARAMETERS

| Category | Parameter | Value |
|---|---|---|
| *Control* | P Coefficient ($K_P$) | 0.05 |
| | D Coefficient ($K_D$) | 0.05 |
| | Error Threshold ($E_{\mathrm{thr}}$) | 10 pixels |
| | Search Speed ($S_{\mathrm{search}}$) | 5 rad/s |
| | Forward Speed ($S_{\mathrm{fwd}}$) | 5.0 |
| *Vision* | Min Area ($A_{\min}$) | 30 pixels |
| | Safe Area Threshold ($A_{\mathrm{safe}}$) | 900 pixels |
| | HSV Lower Bound | [20, 70, 60] |
| | HSV Upper Bound | [35, 255, 255] |
| | Morph Kernel Size | (7, 7) |
| *Collection/ Avoidance* | Min GPS Collect Dist ($D_{\mathrm{collect}}$) | 0.3 m |
| | Avoidance Radius ($R_{\mathrm{avoid}}$) | 0.6 m |
| *Termination* | Max Time w/o Collect ($T_{\mathrm{timeout}}$) | 60.0 s |
| *Filters* | Process Noise ($\mathbb{Q}$) | diag(0.01, 0.01, 0.001) |
| | Measurement Noise ($\mathbb{R}$) | diag(100, 0.01) |
| | UKF Alpha ($\alpha$) | 0.1 |
| | UKF Beta ($\beta$) | 2.0 |
| | UKF Kappa ($\kappa$) | 0 |
| *Simulation* | Timestep ($\Delta t$) | 32 ms |
| | Focal Length ($f$) | 77.25 pixels |

and evaluate the path planning capacity given the recorded GPS information. The main experiment involved:

- Placing 3 yellow balls at known, distinct locations within the arena.
- Running the robot controller which uses the EKF and UKF state estimate for active control (turning, approaching).
- Logging the raw measured pixel error, the EKF's filtered pixel error, and the UKF's filtered pixel error at each timestep where a valid target was detected.
- Recording the GPS locations upon successful ball collections.
- Allowing the run to continue until either all specified balls were collected or the termination condition (timeout since last collection) was met.
- Multiple trials (5 runs) were conducted with the same initial setup to assess consistency.
- Using the mean GPS coordinates obtained from previous trials, we applied the A* algorithm to compute the optimal path and its total length. This result was then compared with the straight-line paths between target locations to evaluate the effectiveness of our path planning approach.

### C. Baseline and Metrics

*1) Baseline Algorithms:* To evaluate the effectiveness of the implemented filters, the following baselines were considered:

- **Raw Measurement:** The unfiltered horizontal pixel error ($e_{\mathrm{meas}}$) calculated directly from the detected contour centroid serves as the baseline input signal, representing the performance without any state estimation filtering.
- **EKF Performance:** The EKF provides the baseline filtering performance, as it is the filter actively used for controlling the robot in these experiments.
- **UKF Performance:** The UKF's filtering output, generated using the same inputs and noise parameters as the EKF, serves as the primary comparison point to evaluate potential improvements offered by its different approach to handling non-linearity.
- **Path Planning Accuracy**: Compares the optimal path generated by the A* algorithm with all manually calculated possible paths and their total lengths.

*2) Validation Metrics:* Performance was evaluated using both qualitative and quantitative metrics:

- **Qualitative Analysis:**
  - Visual inspection of the masks of the yellow balls over the process of robot searching.
  - Visual inspection of time-series plots comparing the raw measured error against the EKF-filtered and UKF-filtered error signals. Key aspects observed were signal smoothness, noise reduction level, tracking responsiveness (lag) during maneuvers, and stability when the error was expected to be near zero (during final approach).
  - Observation of the robot's behavior in the simulation, noting the success or failure of target acquisition, approach smoothness, effectiveness of avoidance maneuvers, and successful task completion (collecting all balls before timeout).
- **Quantitative Analysis:**
  - Filtered Error Variance: Calculation of the overall variance ($\sigma^2$) or standard deviation ($\sigma$) for both the EKF and UKF filtered error time series. Lower values indicate greater smoothness / noise reduction. This was also considered during stable approach phases (e.g., last second before collection) to assess stability near the target state.
  - Object Localization Accuracy: The accuracy of the robot's final position relative to the target at the moment of collection. This is quantified by calculating the 2D Euclidean distance between the robot's recorded GPS coordinates upon collection ($\mathbf{p}_{\mathrm{curr}} = [x, y, z]$) and the known ground truth coordinates ($\mathbf{p}_{\mathrm{true}} = [x_t, y_t, z_t]$, obtained from the simulation world file) of the corresponding target ball: $\sqrt{(x - x_t)^2 + (y - y_t)^2}$. Root Mean Squared Error (RMSE) of these distances (in meters) across all successful collections are reported. Lower values indicate higher accuracy in reaching the target location before the collection decision and maneuver.
  - We use the straight-line distances using point distance formula to determine the optimal visiting order of the target points, since it reflects the most efficient sequence without considering obstacles. For example, given three coordinates $A$, $B$, and $C$, we calculate the total length for all possible visiting orders (e.g., $A \to B \to C$, $B \to A \to C$, etc.) to determine the optimal path or the optimal visiting order and compare this to A* result.

## IV. Results

### A. Object Detection Accuracy

Because we lack pixel-wise ground truth annotations in our simulated environment, it is infeasible to compute standard detection metrics such as false-positive rate, precision/recall, or mAP/AP. Nonetheless, qualitative inspection (Fig. 3) reveals that under strong shading the bottom portion of the yellow sphere fails to satisfy our hue/saturation thresholds and is omitted from the binary mask. This manifests as a systematic under-segmentation in high-contrast regions. While our morphology pipeline recovers some of the missing pixels, residual gaps remain.

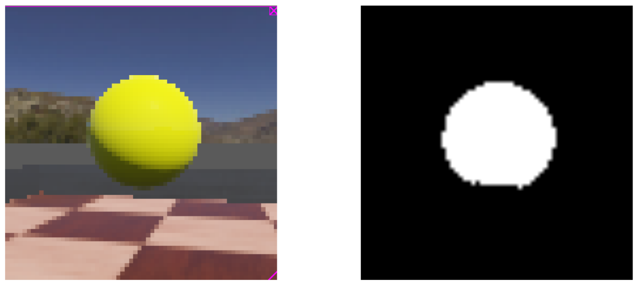


Fig. 3. Detection and Localization Framework

### B. Filter Performance Comparison

To evaluate the effectiveness of the state estimation filters, the filtered horizontal pixel error outputs from the EKF and the UKF were compared against the raw measured error derived directly from the vision system. Figure 4 presents a time-series plot illustrating these three signals during a representative multi-ball collection run.

Qualitatively, both the EKF (orange trace) and UKF (red trace) significantly reduce the high-frequency noise present in the raw measured error (blue trace), providing a much smoother signal suitable for control. The plot shows periodic oscillations corresponding to the robot's search-and-approach cycles for multiple balls. Visually, both filters appear to track the general trend of the measured error reasonably well, although the EKF trace appears noticeably smoother than the UKF trace in this particular experiment. Neither filter exhibits excessive lag in response to changes in the measured error during turns or approaches.

Quantitatively, the variance of the filtered error signals confirms the visual observation of noise reduction. The raw measured error exhibited a variance of $142.28\,\text{pixels}^2$. The EKF achieved a substantially lower variance of $81.66\,\text{pixels}^2$, indicating significant smoothing. Interestingly, the UKF produced a higher variance of $106.65\,\text{pixels}^2$, which, while still much lower than the raw measurement, was less smooth than the EKF output in this run.

Based on these results, the EKF demonstrated superior noise reduction (lower variance) compared to the UKF for this specific experimental run and parameter tuning, while both filters effectively tracked the target signal compared to the raw measurements.

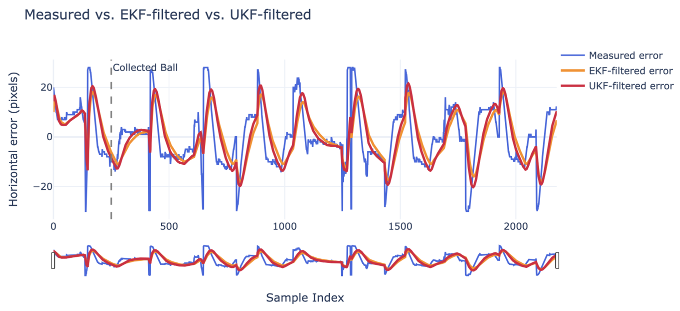


Fig. 4. Comparison of Measured Error and Filtered Error

### C. Object Localization GPS Accuracy

The positional precision with which the explorer robot stops in front of each target is evaluated in the *world* reference frame. For every successful collection we store the robot's GPS fix $\mathbf{p}_{\text{curr}} = [x, y, z]^{\top}$ at the instant the ball triggers the "collected" flag and compare it against the ground-truth ball coordinates $\mathbf{p}_{\text{true}} = [x_t, y_t, z_t]^{\top}$ taken from the Webots world file. The planar localisation error for the $i^{\text{th}}$ ball is therefore

$$e_i = \left\| \mathbf{p}_{\text{curr},i} - \mathbf{p}_{\text{true},i} \right\|_2 = \sqrt{(x_i - x_{t,i})^2 + (y_i - y_{t,i})^2}\ [\text{m}].$$

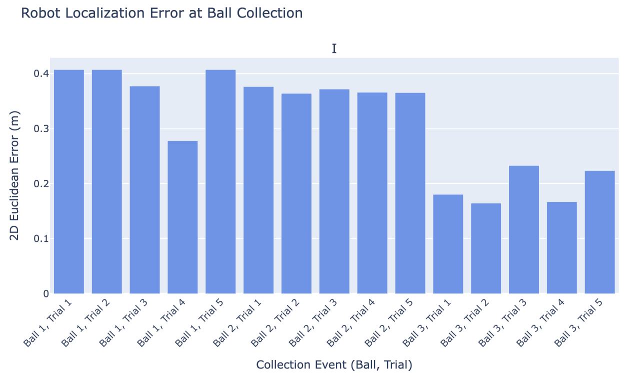


Fig. 5. Per-ball localization error distribution. Most stops are within one ball radius of the true center; the tail corresponds to occasions where partial occlusion reduced visual feedback.

Figure 5 shows individual errors and their distribution. Errors cluster tightly around 0.4m, confirming that the EKF- and UKF-filtered visual servoing reliably brings the robot to within $0.3255 \pm 0.0906$ meters (close to the ball radius) of the ground-truth ball position averagly. Remaining deviations arise when the ball is momentarily lost in shadow during the final approach, causing the robot to execute a corrective turn before stopping.

### D. Path Planning Performance

Given the three GPS coordinates collected by Robot 1, the A* path planning algorithm generates an optimal path:

$B \rightarrow C \rightarrow A$, as shown in Figure 6, with a total length of 345.89 grid units. To validate this result, we calculated the total distances for all possible visiting orders using straight-line (Euclidean) distance between points, as shown in Figure 7. The A* output matches the optimal visiting order, confirming that it selects the correct path sequence. However, the total distance is about 18 units longer. This is expected, as A* moves along a grid and cannot travel in a perfectly straight line between targets—unlike the validation method, which directly computes the shortest possible path using the point-to-point distance formula. Figure 6 illustrates how A* takes slight turns and detours, contributing to the extra distance. Another reason for the discrepancy is that A* operates on a grid-based map, where the environment is discretized into cells. This can introduce path inefficiencies compared to continuous space. One way to reduce this effect is by increasing the resolution used during grid conversion, allowing for finer movement and more accurate path representation.

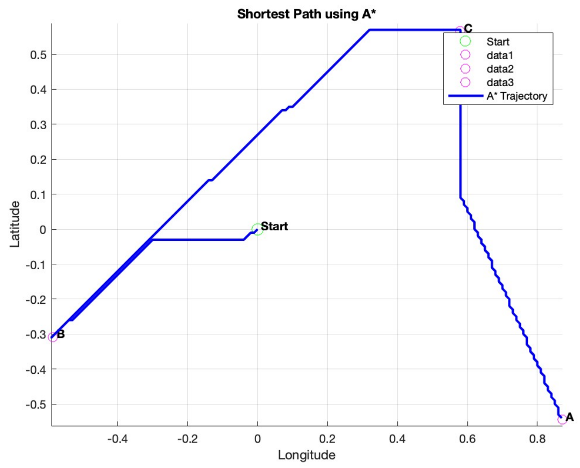
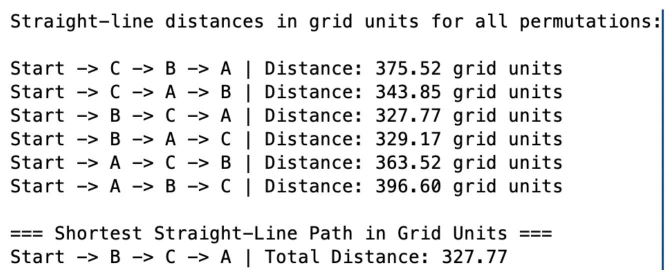


Fig. 6. The A* path planning algorithm suggests a optimal path $B \rightarrow C \rightarrow A$.



```
Straight-line distances in grid units for all permutations:

Start -> C -> B -> A | Distance: 375.52 grid units
Start -> C -> A -> B | Distance: 343.85 grid units
Start -> B -> C -> A | Distance: 327.77 grid units
Start -> B -> A -> C | Distance: 329.17 grid units
Start -> A -> C -> B | Distance: 363.52 grid units
Start -> A -> B -> C | Distance: 396.60 grid units

=== Shortest Straight-Line Path in Grid Units ===
Start -> B -> C -> A | Total Distance: 327.77
```

Fig. 7. All possible path with their length from Matlab.

## V. Conclusion & Discussion

Our two-robot system – vision-based detection/localization by Robot A and A* path planning by Robot B – demonstrates that a lightweight HSV+EKF pipeline can achieve sub-meter stopping accuracy in real time. The EKF's low variance compared to the UKF highlights its suitability under moderate nonlinearity and measurement noise. In summary, our key findings include:

- *Detection Quality:* While our simple color-based method operates in real time with minimal overhead, it under-segments shaded regions, motivating future deep-learning detectors (e.g., YOLO).
- *Filter Performance:* Both EKF and UKF substantially reduce measurement noise; the EKF yielded lower variance (81.7px$^2$) than the UKF (106.6px$^2$) in our trials, making it more suitable for stable servoing under our assumptions.
- *Localization Accuracy:* The average stopping error relative to the true ball centers was $0.33 \pm 0.09\,\mathrm{m}$, within the ball's radius, demonstrating reliable end-effector positioning.
- *Path Planning:* The key finding is that the A* algorithm successfully identifies the optimal visiting order of GPS targets, and while its total path length is slightly longer than the straight-line baseline due to grid constraints, this gap can be minimized with higher resolution or post-processing.

However, we also have some failure cases. We observed under-segmentation when strong shadows dropped portions of the ball outside our hue thresholds, leading to missed detections and filter "resets". Occasionally, these dropped measurements caused the EKF to overshoot or reverse spinning, since the simple constant-velocity model could not anticipate sudden loss of visual feedback.

Among all the experiments and cases, the system excelled in uniformly lit scenes with low background clutter. In these runs, the EKF's noise covariance ($\mathbf{Q} = \mathrm{diag}(0.01, 0.01, 0.001)$) and measurement covariance ($\mathbf{R} = \mathrm{diag}(100, 0.01)$) provided a good balance of smoothness and responsiveness, giving us stable servoing with minimal lag.

Considering the existing failure cases and according to what we observed in the experiments, we propose the following improvements for Future Iterations:

- *Robust Detection:* Replace HSV thresholding with a deep detector (YOLO/Faster-RCNN) to handle shading and complex backgrounds.
- *Augmented Filter Model:* Fuse IMU/odometry yaw rates into the prediction step or adopt a particle filter to manage multimodal uncertainty when measurements drop.
- *Physical Validation:* Deploy on hardware (TurtleBot/AlphaBot) to evaluate performance under real-world lighting and dynamic obstacles.

Finally, we want to discuss on our ethical considerations: Real-world deployment in disaster zones raises safety and trust issues: false positives could waste precious time or endanger bystanders; privacy concerns arise if cameras record sensitive scenes; and reliability under variable conditions (dust, smoke, debris) must be guaranteed. Rigorous field testing, transparent failure reporting, and consent protocols will be essential before any rescue-robot system is entrusted with human lives.

The source code is stored in GitHub repository.

## References

[1] M. T. Shahria, M. S. H. Sunny, M. I. I. Zarif, J. Ghommam, S. I. Ahamed, and M. H. Rahman, "A comprehensive review of vision-based robotic applications: Current state, components, approaches, barriers, and potential solutions," *Robotics*, 2022, affiliations: Computer Science, University of Wisconsin-Milwaukee; Marquette University; Sultan Qaboos University; Mechanical Engineering, University of Wisconsin-Milwaukee. Correspondence to M.H. Rahman. [Online]. Available: https://www.mdpi.com/2218-6581/11/6/139

[2] S. Daftry, M. Das, J. Delaune, C. Sorice, R. Hewitt, S. Reddy, D. Lytle, E. Gu, and L. Matthies, "Robust vision-based autonomous navigation, mapping and landing for mavs at night," in *Proceedings of the 16th International Symposium on Experimental Robotics (ISER)*, Buenos Aires, Argentina, 2018. [Online]. Available: https://www-robotics.jpl.nasa.gov/media/documents/ISER_2018_Final_with_copyright.pdf

[3] Y. Zhang, H. Yan, D. Zhu, J. Wang, C.-H. Zhang, W. Ding, X. Luo, C. Hua, and M. Q.-H. Meng, "Air-ground collaborative robots for fire and rescue missions: Towards mapping and navigation perspective," *arXiv preprint arXiv:2412.20699*, 2024. [Online]. Available: https://arxiv.org/abs/2412.20699

[4] K. Fransen and van Eekelen, "Efficient path planning for automated guided vehicles using a* (astar) algorithm incorporating turning costs in search heuristic," *International Journal of Production Research*, 2021. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/00207543.2021.2015806#abstract

[5] N. Krishnaswamy, "A comparison of efficiency in pathfinding algorithms in game development," DePaul University, Honors Senior Thesis, 2009. [Online]. Available: https://www.cdm.depaul.edu/academics/research/Documents/TechnicalReports/2009/TR09-002.pdf

[6] Webots, "https://cyberbotics.com," 2025, commercial Mobile Robot Simulation Software. [Online]. Available: https://cyberbotics.com

[7] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[8] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*. Orlando, FL: International Society for Optics and Photonics, 1997.

[9] A. R. Smith, "Color gamut transform pairs," in *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '78. New York, NY, USA: Association for Computing Machinery, 1978, p. 12–19. [Online]. Available: https://doi.org/10.1145/800248.807361

[10] R. R. L. Jr., "Filterpy: Kalman filtering and optimal estimation library in python," https://github.com/rlabbe/filterpy, 2023, version 1.4.5.