

Think Fast: Real-Time Kinodynamic Belief-Space Planning for Projectile Interception

Gabriel Olin, Lu Chen, Nayesha Gandotra, Maxim Likhachev, Howie Choset
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract—Intercepting fast moving objects, by its very nature, is challenging because of its tight time constraints. This problem becomes further complicated in the presence of sensor noise because noisy sensors provide, at best, incomplete information, which results in a distribution over target states to be intercepted. Since time is of the essence, to hit the target, the planner must begin directing the interceptor, in this case a robot arm, while still receiving information. We introduce a tree-like structure, which is grown using kinodynamic motion primitives in state-time space. This tree-like structure encodes reachability to multiple goals from a single origin, while enabling real-time value updates as the target belief evolves and seamless transitions between goals. We evaluate our framework on an interception task on a 6 DOF industrial arm (ABB IRB-1600) with an onboard stereo camera (ZED 2i). A robust Innovation-based Adaptive Estimation Adaptive Kalman Filter (RIAE-AKF) is used to track the target and perform belief updates.

I. INTRODUCTION

Intercepting fast-moving objects requires split-second decisions under incomplete information. While a robot must begin moving toward an intercept even before the object's trajectory is fully observed, sensors provide noisy and partial estimates, producing a range of possible future states. This makes the interception problem not only a matter of planning dynamically feasible motions but also of reasoning under uncertainty in real time.

Prior work in robotic interception has shown how dynamics and timing can be incorporated into planners for interception [1], [2], but these methods often have a narrow success rate when dealing with noisy sensing, as they do not reason about uncertainty. In other motion planning problems with moving obstacles, such as autonomous driving, [3]–[5] planners explicitly reason about evolving beliefs of moving agents to plan robust collision-avoiding trajectories, but often these planners can only run at several Hz. For close range interception, where planning and execution need to be done within a fraction of a second, it becomes necessary to develop even more efficient algorithms.

In this work, we introduce a tree-based kinodynamic planner in state-time space, constructed from motion primitives rather than extend operators. This structure compactly encodes reachability to multiple goals, supports fast value updates as the target belief evolves, and enables seamless switching between intercept candidates. We combine this planner with an Innovation-based Adaptive Estimation Adaptive Kalman Filter (IAE-AKF) to maintain a belief over target trajectories. We evaluate the approach with hardware experiments on a 6-DOF ABB IRB-1600 robot arm and a ZED 2i stereo camera, demonstrating robust interception under uncertainty.

II. RELATED WORK

A. Projectile Interception and Robotic Catching

Recent efforts on projectile interception with robot manipulators have explored both optimization-based planning and learning-based approaches. Bäuml et al. [1] present one of the earliest demonstrations of kinematically optimal ball catching with a 7-DOF arm and 12-DOF hand. Their approach leverages stereo vision and an extended Kalman filter (EKF) for trajectory prediction, while solving for catch time and configuration as a nonlinear constrained optimization problem. Real-time feasibility is achieved via parallel SQP, supporting multiple catching behaviors.

Sintov and Shapiro [2] propose the Time-Based RRT (TB-RRT), which augments RRT with temporal information to enable rendezvous planning under timing constraints. The algorithm is validated on interception tasks such as a 1-DOF bat striking a ball and a 3-DOF robotic arm catching a moving object, demonstrating precise time-synchronized planning.

Natarajan et al. [6] propose a preprocessing-based kinodynamic motion planning framework that leverages the IN-SAT planner and dome discretization to precompute smooth trajectories offline, enabling online retrieval of collision-free intercept motions in under a millisecond. While effective under strict real-time constraints, this approach relies on open-loop planning, which only uses one trajectory prediction to plan, and does not reason about uncertainty.

Kim et al. [7] present a programming-by-demonstration method that learns models of object dynamics and grasping configurations, using dynamical systems to generate rapidly adaptable catching motions. Their method achieves successful catching of diverse non-spherical objects but requires extensive demonstration data and motion capture infrastructure to train reliable models. Although these approaches consider dynamics and time constraints in the planning problem, they do not explicitly reason about uncertainty.

B. Belief-Space Planning under Uncertainty

In certain planning under uncertainty problems, the belief evolution of objects in the robot's environment is decoupled from action selection, which enables efficient planning approximations. In the autonomous driving space, the problem of online belief space planning has been applied to driving scenarios with uncertain obstacle motion or pedestrian intention. Aoude et al. [3] introduce RRT-Reach, a sampling-based threat assessment framework for driver assistance at intersections. Coupled with a Bayesian intention predictor,

RRT-Reach evaluates escape trajectories in real time to recommend evasive maneuvers. Their Threat Assessment Module (TAM) is validated in both autonomous and human-driven vehicle experiments, showing effective detection and avoidance of intersection violations. Aoude et al. [4] also extend this work to handle dynamic obstacles with uncertain motion patterns, combining Chance-Constrained RRT (CC-RRT) with Regression-Reduced Gaussian Processes (RR-GP). This approach enforces per-step probabilistic safety guarantees under Gaussian uncertainty and outperforms naive and nominal planners in real-time simulations.

Bai et al. [5] apply POMDP planning to autonomous driving in crowded environments, explicitly reasoning about pedestrian intentions. Their intention-aware online planner achieves real-time performance (3 Hz) and demonstrates robustness in dynamic multi-agent scenarios.

III. PROBLEM FORMULATION

Consider a small projectile that is launched towards a vertical plane 0.8 meters from the base of fixed-based robot manipulator. The state of the projectile is represented as $\rho = (\rho_p, \rho_v)$, where $\rho_p \in \mathbb{R}^3$ denotes its position and $\rho_v \in \mathbb{R}^3$ its velocity. As noisy sensor readings describing the position of the projectile come in, the manipulator must move its end effector to block it before it passes the plane, which covers the region we want to protect.

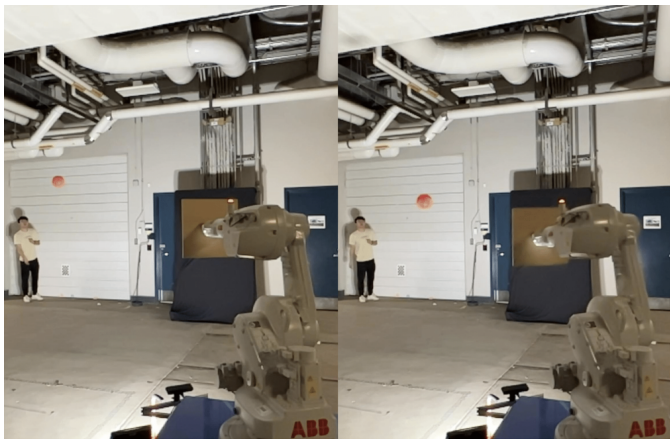


Fig. 1: ABB arm with a shield attached to its end-effector blocking a thrown projectile

A. System Overview

Our manipulator is an ABB IRB-1600 robot arm with six DOF velocity control equipped with a rigidly attached shield to its end effector (Fig. 1). To track and infer the state of the projectile, a ZED 2i stereo camera is attached to the base of the robot which gives regular RGB-D readings at 50Hz, which can be segmented into timestamped (x, y, z) coordinates. The intercept plane is chosen to be orthogonal to the camera’s direction of view. To make the problem tractable in real time, we introduce the following simplifying assumptions:

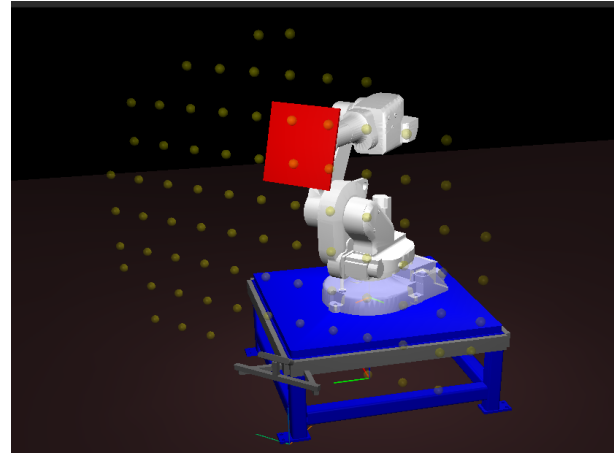


Fig. 2: Task Space Goal Poses

- The manipulator always starts execution from a fixed “home” configuration q_{home} .
- Projectiles are launched from within the field of view of the onboard stereo camera, ensuring early state estimates of their trajectories.
- At any given time, only a single projectile is present in the environment.

B. Goal States

To cover the maximum amount of the robot’s workspace, we compute inverse kinematics solutions for the end effector for (y, z) coordinates at a 0.75 meter discretization in the plane frame (Fig. 2). The orientation of the end effector is constrained so that the shield is coplanar with the plane. The joint configurations computed by a numerical IK solver are saved as goal states for the planner.

IV. COMPUTING BELIEF OVER GOAL STATES

We consider a stochastic projectile state under gravitational acceleration g represented by a Gaussian belief, updated online by an Adaptive Kalman Filter (see Appendix)

$$s = \begin{bmatrix} X \\ Y \\ Z \\ V_x \\ V_y \\ V_z \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma),$$

where (X, Y, Z) are positions and (V_x, V_y, V_z) are velocities, all expressed in the plane frame. We are interested in the time τ at which the trajectory crosses the interception plane $x = x^*$, as well as the corresponding lateral coordinates (Y, Z) at that crossing.

Although computing the probability distribution over the projectile’s state and time when it crosses the intercept plane is generally computationally intractable due to its nonlinear dependence on V_x , the strict time budget of our problem necessitates a cheap analytical approximation to evaluate our belief over goal states online.

A. First-order Delta Method

The crossing time at the plane of interception in the robot's workspace is given by

$$\tau = \frac{x^* - X}{V_x}.$$

Using a first-order linearization about the mean state μ , the distribution of τ is approximated as Gaussian:

$$\mathbb{E}[\tau] \approx \bar{\tau} = \frac{x^* - \mu X}{\mu V_x}, \quad \text{Var}[\tau] \approx g_\tau \Sigma g_\tau^\top,$$

where the gradient of τ with respect to the state is

$$g_\tau = \begin{bmatrix} -\frac{1}{V_x} & 0 & 0 & -\frac{x^* - X}{V_x^2} & 0 & 0 \end{bmatrix}.$$

At the crossing time, the lateral coordinates are

$$Y_{\text{plane}} = Y + V_y \tau, \quad Z_{\text{plane}} = Z + V_z \tau + \frac{1}{2} g \tau^2,$$

where g is the gravitational acceleration (with the sign chosen by convention). Defining the output vector:

$$o = \begin{bmatrix} Y_{\text{plane}} \\ Z_{\text{plane}} \\ \tau \end{bmatrix},$$

we introduce the mapping $h : \mathbb{R}^6 \rightarrow \mathbb{R}^3$ such that

$$o = h(s),$$

Because $h : \mathbb{R}^6 \rightarrow \mathbb{R}^3$ is a nonlinear mapping between the current state and the crossing state/time, the distribution of $o = h(s)$ is non-Gaussian. The *delta method* [8] provides a first-order approximation by linearizing h about the mean μ :

$$h(s) \approx h(\mu) + J(s - \mu),$$

where

$$J = \left. \frac{\partial h}{\partial s} \right|_{s=\mu}$$

is the Jacobian of h evaluated at μ . The rows of $J \in \mathbb{R}^{3 \times 6}$ are given by the partial derivatives of $(Y_{\text{plane}}, Z_{\text{plane}}, \tau)$:

$$\frac{\partial Y_{\text{plane}}}{\partial s} = \begin{bmatrix} -\frac{V_y}{V_x} & 1 & 0 & -\frac{V_y \tau}{V_x} & \tau & 0 \end{bmatrix},$$

$$\frac{\partial Z_{\text{plane}}}{\partial s} = \begin{bmatrix} -\frac{V_z}{V_x} - \frac{g\tau}{V_x} & 0 & 1 & -\frac{V_z \tau}{V_x} - \frac{g\tau^2}{V_x} & 0 & \tau \end{bmatrix},$$

$$\frac{\partial \tau}{\partial s} = \begin{bmatrix} -\frac{1}{V_x} & 0 & 0 & -\frac{x^* - X}{V_x^2} & 0 & 0 \end{bmatrix}.$$

Under this linear approximation, o is approximately Gaussian:

$$o \sim \mathcal{N}(h(\mu), J \Sigma J^\top).$$

We apply this construction with $h(s)$ defined as the mapping from the projectile state $s = (X, Y, Z, V_x, V_y, V_z)$ to the lateral intercept coordinates and crossing time $o = (Y_{\text{plane}}, Z_{\text{plane}}, \tau)$.

B. Joint Gaussian at Plane

This construction yields a joint Gaussian approximation for the lateral coordinates at plane crossing and the crossing time:

$$(Y_{\text{plane}}, Z_{\text{plane}}, \tau)^\top \sim \mathcal{N}(\bar{o}, \Sigma_o),$$

which captures not only the individual variances but also the correlations between τ , Y_{plane} , and Z_{plane} induced by the uncertain initial state.

V. STATE-TIME BELIEF SPACE PLANNING

A. Action Tree

Offline, we construct an *action tree* rooted at the robot's initial state in \mathbb{R}^9 , where each node encodes a kinematic state consisting of joint position, velocity and acceleration (p, v, a) . Even though the ABB robot arm has six degrees of freedom, we plan only for the first three degrees of freedom because they account more most of the task space positional movement, which simplifies the planning problem. The orientation is controlled separately by moving the last three joints to align the shield with the incoming angle of the projectile. This decoupling is possible because the last three joints are able to move far faster and thus are not the limiting factor in the time-optimal control problem to move from one configuration to another.

From the start state, we iteratively expand the tree using minimum-time motion primitives (see Section B), steering towards sub-goals that span a discretization of the robot's reachable front workspace plane. Primitives that are in collision are discarded and cut off at a maximum duration ϵ . Each expansion adds dynamically consistent primitives up to a fixed maximum depth, while storing cost (time-to-come) at each successor node. We choose the tree depth to limit the number of direction changes the robot makes while the projectile is in motion - more decision nodes would result in highly jerky motion and increased strain on the motors. Nodes at the maximum depth are then connected to nearby goal states with zero velocity.

This procedure allows the tree to cover the state-time space between the start configuration and goal configurations, allowing the robot to consider paths during execution that maximize expected reachability to each of the goals (see Section C). The pseudocode for tree construction is shown in Algorithm 1. An example action tree for a 2D double integrator is shown in Fig. 3.

B. Minimum-time Optimal Control

Kinodynamic planning requires satisfaction of both kinematic constraints (e.g., position and velocity limits) and dynamic constraints (e.g., actuation, force, or torque bounds). However, for fully actuated robotic manipulators, exact dynamic feasibility is often unnecessary because low-level joint controllers can accurately track reference trajectories as long as they respect kinematic limits. For this reason, we approximate each joint's dynamics as a *triple integrator system*:

$$\dot{p} = v, \quad \dot{v} = a, \quad \dot{a} = u,$$

Algorithm 1 Action Tree Construction

- 1: **Input:** Start state s_0 , goal states G , goal regions \mathcal{R} , state limits \mathcal{X} , control limits \mathcal{U} , horizon ϵ , maximum depth d_{\max}
 - 2: **Output:** Action tree \mathcal{T}
 - 3: Initialize tree \mathcal{T} with s_0
 - 4: Initialize queue $q \leftarrow \{s_0\}$
 - 5: **while** $\mathcal{T}.\text{max_level} < d_{\max}$ **and** $q \neq \emptyset$ **do**
 - 6: $s \leftarrow q.\text{pop}$
 - 7: **if** $s.\text{level} = d_{\max}$ **then**
 - 8: CONNECTTOCLOSEGOALS(s, G)
 - 9: **continue**
 - 10: **end if**
 - 11: **for all** $g \in \text{SUBGOALS}(s, \mathcal{R})$ **do**
 - 12: $\mathcal{T}.\text{STEER}(s, g, \epsilon)$ $\triangleright \epsilon$ is max duration
 - 13: $q.\text{push}(g)$
 - 14: **end for**
 - 15: **end while**
-

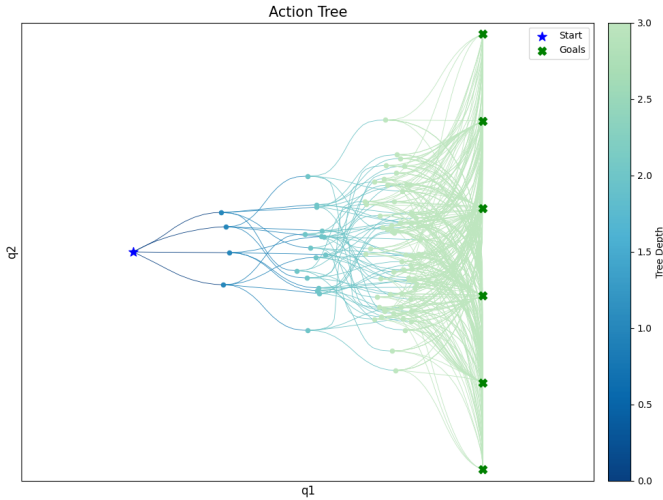


Fig. 3: Action Tree for 2D Double Integrator

where u is the jerk control input.

The objective is to compute the minimum-time, jerk-limited trajectory connecting an initial state (p_0, v_0, a_0) to a terminal state (p_f, v_f, a_f) subject to state and control bounds. The problem can be formulated as:

$$\begin{aligned}
 & \min_{u(\cdot), T} J = T \\
 & \text{s.t. } \dot{a}(t) = u(t), \quad |u(t)| \leq u_{\max}, \\
 & \quad p_{\min} \leq p(t) \leq p_{\max}, \\
 & \quad v_{\min} \leq v(t) \leq v_{\max}, \\
 & \quad a_{\min} \leq a(t) \leq a_{\max}, \\
 & \quad p(0) = p_0, \quad v(0) = v_0, \quad a(0) = a_0, \\
 & \quad p(T) = p_f, \quad v(T) = v_f, \quad a(T) = a_f.
 \end{aligned}$$

From Pontryagin's Minimum Principle, this formulation admits a bang-off-bang profile for the optimal control $u(t)$, which can be solved in closed form or via efficient numerical

methods, providing minimum-time steering primitives for the action tree. We use the highly optimized Ruckig implementation [9] as our solver, because it can handle higher-order non-zero terminal constraints.

C. Value Function

To evaluate the quality of a candidate action, we define a *value function* over the state-time belief space. Each *terminal node* in the tree corresponds to a unique sequence of motion primitives that reaches a particular goal state $g \in \mathcal{G}$ with an associated arrival time τ_g .

Let the belief over the lateral intercept coordinates and crossing time be represented by a Gaussian distribution

$$b(o) = \mathcal{N}(o \mid \bar{o}, \Sigma_o), \quad o = \begin{bmatrix} Y_{\text{plane}} \\ Z_{\text{plane}} \\ \tau \end{bmatrix}.$$

Because our camera is fixed to the base frame, our actions are not coupled to the observations we receive. This allows for the use of the QMDP approximation of the underlying POMDP, where our actions are the motion primitives in the tree, the beliefs are the Gaussian distribution over lateral intercept coordinates and time, and reward is a binary signal of whether the shield blocks the projectile before intercept time. Under the QMDP approximation, the value of a node n under belief $b(o)$ can be written as

$$V(n, b(o)) = \mathbb{E}_{o \sim b} \left[R(n, o) + \max_{a \in \mathcal{A}} V(n', b(o)) \right],$$

where $R(n, o)$ is the reward of being at node n

We define the reward function only for terminal nodes. For a terminal node n , let S denote the set of lateral coordinates (Y, Z) covered by the shield at the intercept plane. Then

$$R(n, o) = \begin{cases} 1, & (o_Y, o_Z) \in S \text{ and } n.\text{time} < o_\tau, \\ 0, & \text{otherwise,} \end{cases}$$

where $o = (o_Y, o_Z, o_\tau)$ are the intercept coordinates and the crossing time. This value function represents the expected probability of successfully intercepting the projectile, given the current distribution over intercept coordinates and crossing time.

The tree structure of our decision process allows terminal node values to be computed with each belief update and then propagated backward through the tree in a single online pass. During execution, the next primitive is selected by choosing the successor of the current node that maximizes the value function:

$$a^* = \arg \max_{a \in \mathcal{A}(n)} V(n', b),$$

where $\mathcal{A}(n)$ denotes the set of available successors of node n , and n' is the child node reached by applying action a . If a terminal node is reached while projectile measurements are still being received, the planner tells the robot to move towards the most probable goal.

The closed-form Gaussian approximation of the belief state, along with efficient value updates in our action tree, allows for

online planning. Our planner consistently computes the next action online in under 10ms on an AMD Threadripper Pro 5995WX workstation.

VI. EXPERIMENTS

We tested the full system with integrated planning, perception, and control on an IRB-1600 robot arm, equipped with a ZED 2i stereo camera in an indoor environment. Balls were thrown towards the robot's front plane from distances from 6-8m. The time of flight of the throws were typically between 400ms and 800ms. We compared our method to a naive baseline, in which the robot moves (under kinematic constraints) as fast as possible to the current most likely goal configuration at each belief update.

Out of 50 throws of varying initial velocity and distance, our method achieved a blocking success rate of 74%. The naive approach still achieved 63% blocking success. The main cause of failure was insufficient execution time. While the perception module eventually returned an accurate intercept location, the end effector was too far away to reach it in time under its kinematic limits.

VII. DISCUSSION

While both our method and the naive approach did not achieve 100% success rate, the improvement shows a benefit to considering uncertainty in extreme time-constrained interception problems. By encoding time-constrained reachability in the value function, our planner is able to hedge against a distribution of future observations, making it more robust than the naive approach which commits immediately to the first information it receives. Further work will include optimizing for manipulability of the robot arm in the motion primitives, which may allow for more agility when needing rapidly changing directions in task space. Additionally, the discrete action space may be limiting to when and how the robot can move, so optimization or learning based approaches may provide a strong continuous action space analog to our method.

VIII. CONCLUSION

We presented a real-time kinodynamic belief space planning framework for robotic projectile interception under uncertainty. By constructing an action tree in state-time space with jerk-limited motion primitives, our method enables fast online value updates as the projectile belief evolves, allowing the robot to seamlessly adapt its interception strategy. The QMDP-based value function, defined over the Gaussian distribution of intercept states, provides a principled approximation to reason about success probabilities under partial information.

Our integrated system—combining robust RIAE-AKF state estimation with kinodynamic motion primitives—demonstrated improved interception success over a naive baseline on hardware experiments with a 6-DOF ABB IRB-1600 manipulator. Despite inherent time constraints and occasional failures due to execution limits, the results highlight the importance of explicitly modeling uncertainty close range interception tasks.

Future work will explore incorporating robot manipulability into the value function, expanding from discrete primitives to continuous optimization or learning-based action spaces, and extending the framework to multi-interceptor scenarios. These directions could further enhance agility and robustness in interception tasks, bringing robotic systems closer to human-level performance in highly dynamic environments.

ACKNOWLEDGMENT

This work was supported by the Air Force Office of Scientific Research (AFOSR) under Grant M2403339 and, in part, by the Army Research Office (ARO) through the DURIP program under Grant W911NF-21-1-0050.

REFERENCES

- [1] B. Bäuml, T. Wimböck, and G. Hirzinger, "Kinematically optimal catching a flying ball with a hand-arm-system," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010, pp. 2592–2599.
- [2] A. Sintov and A. Shapiro, "Time-based rrt algorithm for rendezvous planning of two dynamic systems," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014, pp. 6745–6750.
- [3] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Threat assessment design for driver assistance system at intersections," in *13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. Funchal, Madeira Island, Portugal: IEEE, 2010, pp. 1855–1862.
- [4] —, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, July 2013.
- [5] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, 2015, pp. 454–460.
- [6] R. Natarajan, H. Yang, Q. Xie, Y. Oza, M. P. Das, F. Islam, M. S. Saleem, H. Choset, and M. Likhachev, "Preprocessing-based kinodynamic motion planning framework for intercepting projectiles using a robot manipulator," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 1–8, arXiv:2401.08022.
- [7] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.
- [8] M. Reith-Braun, F. Pfaff, J. Thumm, and U. D. Hanebeck, "Approximate first-passage time distributions for gaussian motion and transportation models," in *Proceedings of the 26th International Conference on Information Fusion (FUSION)*. IEEE, 2023, pp. 1–8.
- [9] L. Berscheid and T. Kröger, "Jerk-limited real-time trajectory generation with arbitrary target states," *arXiv preprint arXiv:2105.04830*, 2021. [Online]. Available: <https://arxiv.org/abs/2105.04830>
- [10] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>
- [11] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.05499>
- [12] X. ZhiWen, H. XiaoPing, and L. JunXiang, "Robust innovation-based adaptive kalman filter for ins/gps land navigation," in *2013 Chinese Automation Congress*, 2013, pp. 374–379.
- [13] D. C. Silva, A. Frutuoso, L. F. Souza, and E. A. de Barros, "Comparative analysis of innovation-based adaptive kalman filters applied to auvs navigation," in *2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)*, 2022, pp. 31–36.
- [14] Y. Geng and J. Wang, "Adaptive estimation of multiple fading factors in kalman filter for navigation applications," in *GPS Solution*, 2008, pp. 273–279.

Perception and State Estimation

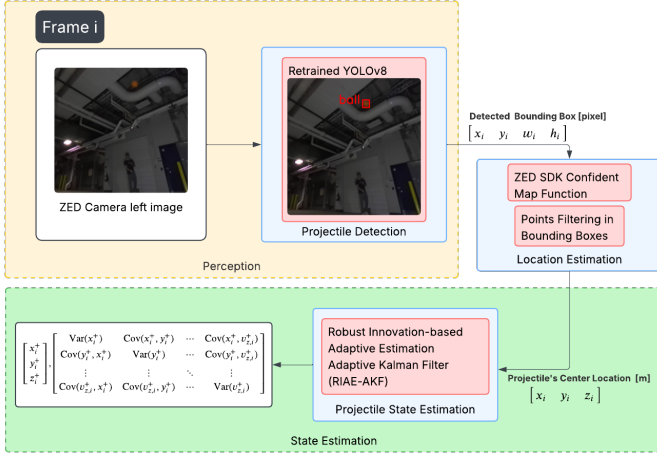


Fig. 4: Flowchart showing Perception and State Estimation

Perception is affected by multiple sources of noise, including low-resolution images, lighting variations, and complex backgrounds. Simple detection algorithms like a color filter, as used in [6], can struggle with robustness to these conditions. To reduce these effects and improve detection accuracy, we use a retrained YOLOv8 (You Only Look Once, version 8) [10] to detect the projectile in each frame using laboratory videos. The DINO grounding [11] was used as the base model, with AutoDistill CaptionOntology used to generate the training dataset. A custom annotation GUI was developed, allowing human verification and refinement of the ground truth. For each frame from the left ZED camera, the perception module detects the projectile and outputs its bounding boxes $[x_i, y_i, w_i, h_i]$, which contain the center point in the x and y axis and its width and height. To obtain the 3D location of the projectile, the detected bounding boxes are adjusted using a reverse sigmoid function based on the ratio of the bounding box area and image area. It keeps shrinking as the ball approaches the camera: 5.3% shrinkage in width and length when the ball is about 7-8 meters away, and up to 27.6% shrinkage when it is very close. These modified bounding boxes are then used to retrieve the projectile locations from ZED SDK's confidence map and point clouds [6]. This adjustment reduces the noise from motion blur when the ball moves rapidly near the camera. The module outputs the projectile's estimated 3D center location $[x_i, y_i, z_i]$.

A total of 1,408 images were used for training, including both high- and low-resolution cases, as well as scenarios in which the ball was held in the hand. The proposed perception module can detect the ball earlier than the color filter method (that is, before it is thrown), achieving an 81.3% detection success rate across 385 evaluation images, with an average center difference of 0.97 pixel units compared to the ground-truth bounding box center. In contrast, the color filter method achieved only a success rate of 38.1% and a difference of 1.94

pixel units. An overview of the perception and state estimation pipeline is given in Fig. 2.

While detecting and segmentating the position of a projectile at several timesteps, a Kalman Filter (KF) can optimally estimate its state accounts for measurement and process noise with covariance matrices Q and R, but only when both are constant. In our case, different throws produce varying measurement noise (Fig. 5), which makes the filtering process difficult. A constant process noise covariance is suitable since the model is only influenced by gravity ($g = 9.81 \text{ m/s}^2$). The innovation-based estimate adaptive Kalman filter (IAE-AKF) algorithm adjusts Q and R using the innovation history. However, when there is disturbed noise or outliers, its residual can exceed Gaussian assumptions, leading to a degradation of performance [12]. To consider the varying measurement noise covariance and in the projectile's locations, we use Robust IAE-AKF (RIAE-AKF) [12] [13]

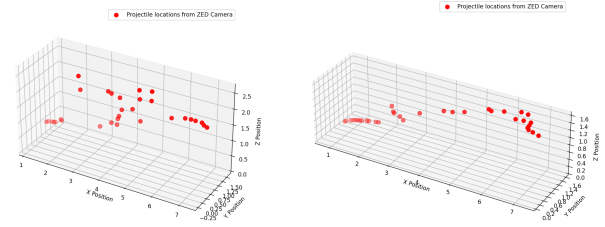


Fig. 5: The left and right figures show 3D locations of the projectile at different throws. Measurement noise varies between throws.

Similar to the Kalman Filter, we need to calculate raw innovation ν_k and covariance matrix after getting standard kalman prediction $\hat{\mathbf{x}}_k^-$ and \mathbf{P}_k^- .

$$\nu_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^- \quad (1)$$

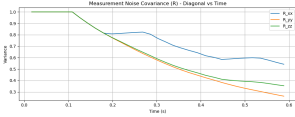
Where \mathbf{H}_k is the observation matrix. For each element, calculate the chi-square statistic $\kappa_k(i)$ [14]:

$$\kappa_k(i) = \frac{\nu_k^2(i)}{\mathbf{C}_{\nu_{k-1}}(i, i)} \sim \chi^2(1) \quad (2)$$

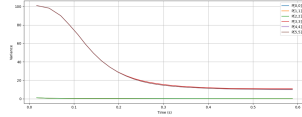
$\nu_k(i)$ is the i-th element of the vector ν_k , $\mathbf{C}_{\nu_{k-1}}(i, i)$ is the i-th diagonal element of empirical innovation covariance matrix $\mathbf{C}_{\nu_{k-1}}$, shown in Eq. (4). $\chi^2(m)$ is a Chi-square distribution with m degree of freedom, and m is the number of innovation vector's elements which we define as 1, assuming each dimension is independent to the other.

Next, we revise the innovation to reduce the influence of abnormal innovation vectors and update its covariance using the innovation history. Here, α represents the confidence level. The revised innovation is computed as

$$\hat{\nu}_k(i) = \begin{cases} \nu_k(i), & \text{if } 0 \leq \kappa_k(i) < \chi_\alpha^2(1) \\ \nu_k(i) \exp\left(-\frac{\kappa_k(i) - \chi_\alpha^2(1)}{\chi_\alpha^2(1)}\right), & \text{if } \kappa_k(i) \geq \chi_\alpha^2(1) \end{cases} \quad (3)$$



(a) Measurement Noise Covariance Matrix R



(b) State Covariance Matrix P

Fig. 6: Measurement noise covariance matrix R and state covariance matrix P over time for the throw shown on the left in Fig. 5. The diagonal values of R vary due to inaccurate locations in the middle of the throw; the diagonal values of P decrease as expected when more information is given.

When $0 \leq \kappa_k(i) < \chi_{\alpha}^2(1)$, the sensor reading is consistent with expectations. If $\kappa_k(i) \geq \chi_{\alpha}^2(1)$, the innovation is abnormal and will need to be modified using Eq. (3). Update Innovation covariance matrix $C_{v,k}$:

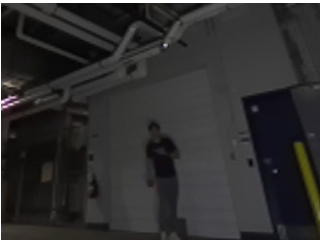
$$C_{v,k} = \frac{1}{N} \sum_{j=k-N+1}^k \hat{\boldsymbol{v}}_j \hat{\boldsymbol{v}}_j^{\top} \quad (4)$$

N is the window size and $\hat{\boldsymbol{v}}_j$ are the revised innovation vectors. Then we update the measurement noise covariance matrix \mathbf{R}_k .

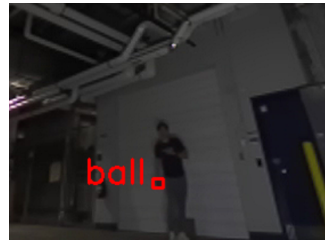
$$\mathbf{R}_k = C_{v,k} - \mathbf{H}_k \mathbf{P}_k^{-} \mathbf{H}_k^{\top} \quad (5)$$

The State Estimation module adapts the measurement noise, recursively updates the state, and outputs the projectile state $\mathbf{x}_i^+ = [x_i^+, y_i^+, z_i^+, v_{x,i}^+, v_{y,i}^+, v_{z,i}^+]^T$ and the covariance matrix P :

$$P = \begin{bmatrix} \text{Var}(x_i^+) & \text{Cov}(x_i^+, y_i^+) & \cdots & \text{Cov}(x_i^+, v_{z,i}^+) \\ \text{Cov}(y_i^+, x_i^+) & \text{Var}(y_i^+) & \cdots & \text{Cov}(y_i^+, v_{z,i}^+) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(v_{z,i}^+, x_i^+) & \text{Cov}(v_{z,i}^+, y_i^+) & \cdots & \text{Var}(v_{z,i}^+) \end{bmatrix}.$$



(a) An example of a zoomed in frame received by the left ZED camera.



(b) Same frame with the projectile detected, which the Color Filter fails.

Fig. 7: Unlike the Color Filter [6], the retrained YOLOv8 can detect the projectile in the early stage, even before the ball was thrown.

To evaluate performance compared to the work [6], given the absence of ground-truth 3D projectile locations, we used the estimated depth results to compute the intersection with a virtual vertical wall placed 2 meters from the camera.

This intersection point serves as a reference point for ground truth, since the measurements are empirically closer to the real trajectories of the projectile. The intersection predicted by RIAE-AKF is then calculated, and its distance from the reference point of the ground truth is calculated to compare with the work [6]. shows that the work [6] has fluctuated prediction compared to the proposed work.