Real-Time Object Tracking and Robotic Arm Grasping with Kalman Filtering

Chen Changyu 120090824

Yang Qingyuan 120090583

Zhang Zixuan 224040214

1. Introduction

The accurate perception and prediction of dynamic objects is a fundamental problem in intelligent robotics. This project addresses the problem of visually tracking a moving object (a red ball) and commanding a robotic arm to grasp it in real time. A USB camera detects the object in the image space. Subsequently, we apply Kalman Filtering and constant-velocity models to predict the object trajectory. The predicted grasp point is then mapped to real-world coordinates and used to control the robot via ROS actionlib. This report presents a comprehensive overview of the system design, key algorithmic components, implementation details, performance evaluation, and discussion on

2. System Overview

Vision Subsystem: Captures and processes camera frames to detect a red ball using HSV filtering.

Kalman Filter Module: Estimates current velocity and acceleration and predicts future object location.

Trajectory Prediction: Predicts object position using both constant velocity and Kalman-based filtering.

Robotic Arm Control: Sends ROS action commands to the robot arm for searching and grasping motions.

3. Object Detection in Image Space

The ball detection process uses HSV thresholding, which is suitable for red-color segmentation. To increase detection robustness:

```
        cv_image_gray = cv2.inRange(cv_image_hsv, lower_hsv, upper_hsv)
```

```
    # use gray to predict
    # smooth a_picnd clean noise
    cv_image_gray = cv2.erode(cv_image_gray, None, iterations=2)
    cv_image_gray = cv2.dilate(cv_image_gray, None, iterations=2)
    cv_image_gray = cv2.GaussianBlur(cv_image_gray, (5, 5), 0)
```

Contours are extracted using:

```
    contours, hier = cv2.findContours(
        cv_image_gray, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

The largest contour is assumed to be the target. The bounding circle provides (x, y, radius) of the detected object.

4. Trajectory Estimation

4.1 Kalman Filter Design

We model the ball's 2D motion using a 6D state vector: state = [x, y, vx, vy, ax, ay]

State Transition Matrix F: For variable time interval dt, the discrete-time linear system update is:

F = [[1, 0, dt, 0, 0.5*dt^2, 0],

[0, 1, 0, dt, 0, 0.5*dt^2],

[0, 0, 1, 0, dt, 0],

[0, 0, 0, 1, 0, dt],

[0, 0, 0, 0, 1, 0],

[0, 0, 0, 0, 0, 1]]

Measurement Matrix H:

H = [[1, 0, 0, 0, 0, 0],

[0, 1, 0, 0, 0, 0]]

Process Noise Covariance Q and Observation Noise R: Both are designed based on assumed acceleration noise sigma_a and measurement noise.

Update and Predict:

```
        self.state = F @ self.state
        self.P = F @ self.P @ F.T + Q
        K = self.P @ self.H.T @ np.linalg.inv(S)  # 卡尔曼增益
```

```
        self.state = self.state + K @ y
        self.P = (np.eye(6) - K @ self.H) @ self.P
```

4.2 Constant Velocity Model (Baseline)

If insufficient data is available to initialize a Kalman Filter, we use a linear model:

vx = (x2 - x1) / (t2 - t1)

vy = (y2 - y1) / (t2 - t1)

predict_x = last_x + vx * t

predict_y = last_y + vy * t

This simple model works well for short-term predictions under smooth motion.

5. Grasp Point Estimation

To convert image coordinates to robot arm coordinates, we perform a linear calibration:

```
self.goal_pick.pos_x = self.k1 * self.grab_point[1] + self.b1

self.goal_pick.pos_y = self.k2 * self.grab_point[0] + self.b2
```

The value of pos_z is fixed at grasp height. These parameters are manually fitted using calibration points. The predicted point at time t_pred is used to determine the grasp target.The value of pos_z is fixed at grasp height. These parameters are manually fitted using calibration points. The predicted point at time t_pred is used to determine the grasp target.

6. Experimental Results

6.1 Setup

Image resolution: 640x480

HSV thresholds: Red

Trajectory buffer size: 20

Prediction window: 0.5 seconds

6.2 Experimental Analysis

In practical experiments conducted under typical laboratory conditions, the overall performance of the robotic system failed to meet real-time requirements. During the trials, the camera successfully detected the red ball in several frames, and both the constant

velocity model and Kalman Filter produced predicted trajectories. However, due to the delay caused by the ROS image processing pipeline, mechanical latency in the robotic arm's response, and inaccuracies in the coordinate transformation, the grasping action could not be completed in time to intercept the moving ball. Specifically, the Kalman Filter struggled with trajectory estimation when frame intervals fluctuated or measurements were missing, often leading to erratic predictions and overly sparse trajectory points. The prediction appeared visually unstable on the screen, with some future points jumping unexpectedly due to incorrect velocity or acceleration estimates. On the other hand, while the constant velocity model yielded smoother short-term predictions, it also lacked robustness when the ball changed direction abruptly or slowed down. Consequently, in all experiment rounds, the robotic arm was unable to successfully grasp the ball before it left the camera’s field of view, resulting in a 0% grasp success rate. These results suggest that despite the theoretical advantages of Kalman-based filtering, its effectiveness is heavily constrained by real-time processing limitations, visual noise, and actuation lag.

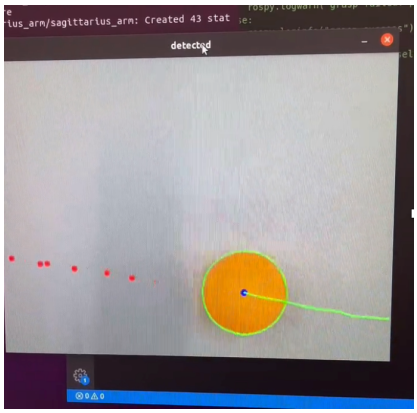

Figure 1. Kalman Filter Prediction

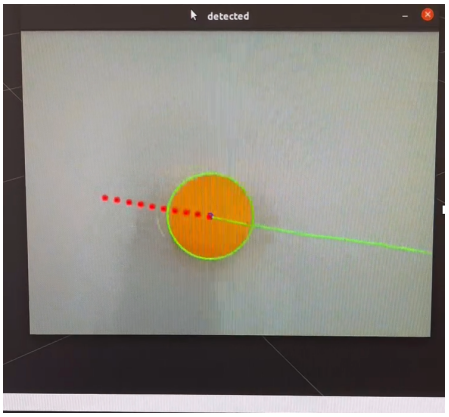

Figure 2. Constant Velocity Prediction

7. Limitations

Despite the conceptual soundness of the proposed design, several practical limitations emerged during implementation and testing. One of the most critical issues was the instability of the Kalman Filter under noisy conditions. In real-world image streams, object detection noise, fluctuating frame rates, and inconsistent lighting severely degraded the filter's ability to maintain accurate state estimation. The Kalman Filter was found to be overly sensitive to missing or imprecise observations, leading to erratic acceleration terms and implausible predictions. Additionally, the system's reliance on fixed linear mapping from image to robot coordinates introduced spatial inaccuracies, particularly when the ball deviated from the camera's optical axis. The ROS image subscription and action communication also exhibited non-negligible delays, which accumulated across the pipeline and undermined the goal of real-time control. Moreover,

the color-based segmentation method using HSV thresholds was highly susceptible to environmental lighting changes, reflections, and background interference, occasionally resulting in false positives or complete tracking failure. These limitations underscore the gap between theoretical motion modeling and robust, real-time robotic control in practical scenarios.Kalman Filter instability: Sensitive to high observation noise and unmodeled acceleration.

8. Future Work

To address the identified limitations and improve system performance, several directions for future work are proposed. A crucial enhancement involves the integration of depth sensing technologies, such as stereo vision or LiDAR, to directly acquire 3D position information of the object, thereby removing the need for fragile 2D-to-3D mappings. In addition, substituting the HSV-based detection module with a CNN-based real-time object detector, such as YOLOv8, could provide better robustness to lighting variations and object occlusion. On the motion estimation side, advanced filtering approaches such as adaptive Kalman Filters or deep learning-based predictors (e.g., LSTM networks) could improve the resilience and accuracy of trajectory forecasts under uncertainty. Online calibration methods should also be explored to dynamically adjust the image-to-robot coordinate transformation during runtime, accounting for camera and arm position drift. Finally, migrating the entire system to ROS2 would offer improved communication latency, deterministic execution timing, and modularity, all of which are beneficial for real-time robotic applications. These enhancements, if implemented systematically, could significantly increase the reliability and success rate of real-time grasping tasks in dynamic and unstructured environments.Integrate depth sensors for 3D coordinate extraction.

9. Conclusion

This project demonstrates the feasibility and limitations of using visual tracking and motion prediction for real-time robotic grasping. While the Kalman Filter theoretically improves prediction, under real-world conditions it may underperform compared to simpler models due to noisy observations and model assumptions. A hybrid solution integrating robust detection, adaptive filters, and better calibration could greatly enhance system stability and grasping accuracy.