Lab 5 Report - Group 18

Qinghua He - qh2297
Jaisel Singh - js6897
Huan Gu - hg2721

Probabilistic Robotics (EEME 6911) Lab 5: EKF Localization

Lab Objective: Derive and implement the image processing algorithm that produces the distance and bearing measurements

Landmark cylinders:

- radius = 0.1m
- height = 0.5m

Assignment 1:

We are given a set of 2D pixel coordinates corresponding to detected corners of a colored landmark in the image. Our goal is to estimate:

- The landmark's height in pixels
- The horizontal position of its vertical symmetry axis in pixel space

We receive an input of N corner points, which has coordinates (x, y) in pixels. x increases to the right and y increases downward. The algorithm works using the following logic:

Edge-case (empty input)

- If there are no points (N = 0), we cannot compute a valid estimate and return no result.

Compute pixel height

- The pixel height of the landmark is computed from the vertical span of all corner points:
  - height_px = (maximum y among all points) minus (minimum y among all points)
- This corresponds to the top and bottom of the detected landmark in the image.

Estimate vertical symmetry axis

- We estimate the x-coordinate of the vertical symmetry axis using a simple rule based on how many points we have:
- **If the number of points is less than 8 (rectangular case):**
  - Let x_min be the minimum x
  - Let x_max be the maximum x
  - The symmetry axis is taken as the midpoint:
    - x_sym = (x_min + x_max) / 2

    - The intuition here is that with fewer points, we approximate the landmark as a rectangle and use its left and right edges.

- **If the number of points is 8 or more (cylindrical case)**:
    - We use the average x of all points:
        - x_sym = (x_1 + x_2 + ... + x_N) / N
    - The intuition with the second approach is that with many points sampled around the contour of a cylindrical landmark, the mean x-coordinate approximates the center line.

The algorithm in the end outputs:

- height_px: estimated landmark height in pixels
- x_sym: estimated x-coordinate of the vertical symmetry axis in pixels

Code Commit Link:
https://github.com/jaiselsingh1/prob_rob_labs_ros_2_group18/commit/0cd288b03bd984e49ffeebd0b183bcd57fd6f1c6

Assignment 2:

The goal of assignment 2 is based on computing the distance and bearing from the camera to a chosen colored landmark, whilst using the camera intrinsics from /camera/camera_info, the detected corner points of that landmark from /vision_<color>/corners and the known physical height of the landmark.

The inputs here are based on:

- subscribe to /camera/camera_info and then recover the camera parameters: fx, fy, cx and cy
- subscribe to the node "/vision_<color>/corners", where color is also a parameter
- Use a parameter landmark_height for the real-world landmark height in meters

Landmark height in pixels:

- From all received corner points:
    - We first compute: height_px = max(y_i) - min(y_i)
    - If height_px <= 0 or too small, skip the measurement then it is unreliable:

Vertical symmetry axis in pixels:

- Sort points by y
- take a few points near the top and a few near the bottom.

The finally you compute:

- x_top = average x of top band
- x_bot = average x of bottom band
- x_sym = (x_top + x_bot) / 2
- This reduces sensitivity to noise and missing corners. If x_top and x_bot differ too much (large skew), we drop the measurement as unreliable (e.g., when one side of the landmark is cut off near the edge of the image).

For the bearing computation, we use the pinhole model:

- (cx - x_sym) / fx = tan(theta)
- theta = atan((cx - x_sym) / fx) where theta is the bearing of the landmark center relative to the camera optical axis.

For the distance computation:

- From similar triangles and accounting for the bearing:
    - height_px ≈ (landmark_height * fy / distance) * cos(theta)
    - So:
        - distance = landmark_height * fy / (height_px * cos(theta))
        - If cos(theta) is too close to zero, skip (numerically unstable)

Publishing

- Publish distance and bearing on a dedicated topic, for example:
    - “/vision_<color>/measurement”
- A simple encoding is to use geometry_msgs/PointStamped with:
    - point.x = distance
    - point.y = bearing
    - point.z = 0
- When the landmark goes out of sight or the geometry checks fail, do not publish a new measurement for that frame and do not crash.

The way we handle FOV edge-cases is:

- the landmark is near the edge of the camera’s field of view, some corners on one side may be missing. This causes x_top and x_bot to be inconsistent and leads to bad x_sym, hence bad distance. The node detects this via a horizontal skew check and drops those measurements, so the distance plot stays stable except in clearly unreliable conditions.

Plot Links: [prob_rob_labs_ros_2_group18/pictures/L5A2.webm at master · jaiselsingh1/prob_rob_labs_ros_2_group18]

Code Commit Link:
https://github.com/jaiselsingh1/prob_rob_labs_ros_2_group18/commit/6a7dcbffc9b0f2b777baa93575fcb1c49c49892f

Assignment 3:

prob_rob_labs_ros_2_group18/pictures/L5A3.webm at master · jaiselsingh1/prob_rob_labs_ros_2_group18

For assignment 3, the goal is to compare the vision-based landmark measurements (distance and bearing) against ground truth and analyze the error. From the link above you can see the behavior and error being plotted which highlights the key behavior of the robot:

- When the robot faces the landmark and moves closer/farther, d_meas tracks d_true reasonably well.
- When the robot rotates in place at a fixed distance:
    - d_meas remains approximately constant in the central field of view.
    - theta_meas varies smoothly with rotation.
- When the landmark moves toward the edge of the camera image:
    - Missing or biased corner detections cause incorrect x_sym.
    - This can distort both theta and d.
    - Our skew and minimum-height checks drop such frames, so we avoid large outliers instead of publishing bad measurements.

This node provides both real-time measurements and error signals that we use in Assignment 4 to model the measurement covariance.

Code Commit Link:
https://github.com/jaiselsingh1/prob_rob_labs_ros_2_group18/commit/6a7dcbffc9b0f2b777baa93575fcb1c49c49892f (As a note, both assignment 2 and 3 were committed together at once)

Assignment 4:

To use the vision-based landmark measurements in the EKF, we need an empirical measurement noise model R(d, θ) for distance and bearing.
We used the landmark_positioner node for a single landmark (cyan). The node first computes the measurement (distance, bearing) from image corners and camera intrinsics. It also computes ground truth distance and bearing from /gazebo/link_states

Once it completes the computations, it proceeds to log:

- measured_d, measured_theta
- true_d, true_theta

- err_d = measured_d – true_d
- err_theta = measured_theta – true_theta

These logs were then saved into landmark_data_cyan.csv while driving and rotating the robot around the landmark. In order to make the model more visually identifiable, we used LaTeX to type up the mathematical structure:

Noise model $R(d, \theta)$

We model the measurement noise as diagonal and piecewise, based on the observed behavior. Let $\sigma_d^2(d)$ and $\sigma_\theta^2(\theta)$ denote the distance and bearing variances.

For distance:

$$\sigma_d^2(d) = \begin{cases} \sigma_{d\text{base}}^2 = 0.0176, & 1.0 \le d \le 10.0, \\ k_d\, \sigma_{d\text{base}}^2, & \text{otherwise}, \end{cases}$$

for bearing:

$$\sigma_\theta^2(\theta) = \begin{cases} \sigma_{\theta\text{base}}^2 = 1.57 \times 10^{-4}, & |\theta| \le 0.6, \\ k_\theta\, \sigma_{\theta\text{base}}^2, & \text{otherwise}, \end{cases}$$

where k_d and k_θ are inflation factors. In our implementation we use $k_d = k_\theta = 3$ to downweight measurements obtained at extreme distances or near the image boundaries.

The resulting measurement covariance used in the EKF is

$$R(d, \theta) = \begin{bmatrix} \sigma_d^2(d) & 0 \\ 0 & \sigma_\theta^2(\theta) \end{bmatrix}.$$

This model reflects that measurements are most reliable when the landmark is within a moderate range and near the center of the camera image, and less reliable when it is too close, too far, or near the edge of the field of view.

The results from our script are shown below to validate the model:

```
N samples = 1842
base var_d     = 1.761867e-02
base var_theta = 1.573656e-04

example sigma_d^2(d):
  d= 0.5 -> 5.285602e-02
  d= 2.0 -> 1.761867e-02
  d= 5.0 -> 1.761867e-02
  d=12.0 -> 5.285602e-02

example sigma_theta^2(theta):
  theta= 0.00 -> 1.573656e-04
  theta= 0.30 -> 1.573656e-04
  theta= 0.80 -> 4.720968e-04

example R(3.0, 0.2):
[[0.01761867 0.        ]
 [0.         0.00015737]]
```

Code and data:
Record data:
[Enable CSV recording, add diagnostics, and extend launch args · jaiselsingh1/prob_rob_labs_ros_2_group18@a84e2b1]
Analyze:
[Add variance_model utility and add L5A4.png · jaiselsingh1/prob_rob_labs_ros_2_group18@f39e42f]