Marker Follower

A Python-based system for detecting ArUco markers in images or video, and computing robot motion commands to follow a specific marker. This repository includes:

- A reusable marker detection class
- A standalone video processing script
- A ROS node for real-time robot control

Contents

`marker_detector.py`

- `MarkerDetector` class.
- Member methods:
  - `detect()`: Detect ArUco markers in the image frame:
    - Uses OpenCV's ArUco module to:
    - Detect markers in images.
    - Estimate each marker's 3D pose (`rvec`, `tvec`) relative to the camera.
  - `compute_velocity()`: Compute linear and angular velocities based on:
    - Marker distance from the camera.
    - Pixel offset of the marker's center from the image center.
- Parameters:
  - Camera intrinsic parameters: `camera_matrix`, `dist_coeffs`

`run.py`

- A python script to:
  - Load a video file.
  - Detect markers frame-by-frame.
  - Compute desired robot velocities for following marker ID 1.
  - Overlay:
    - Marker bounding boxes.
    - Marker ID text.
    - Velocity vectors and debug info directly on the video frames.
- Draws arrows or curves representing motion commands:
  - Straight arrows for pure linear movement.
  - Curved paths for simultaneous rotation and translation.
- Inputs:
  - `video_path`: (e.g.,`"../examples/2025-06-30-08-20-43.mp4"`)

PROF

▶ View the example input video

- Parameters:
    - Controller parameters:
        - `desired distance`
        - `k_linear`
        - `k_angular`
        - `max_linear_speed`
        - `max_angular_speed`

How to run:

```
python3 run.py
```

▶ View the visualization of the result

`marker_follower_ros.py`

A ROS wrapper of `marker_detector.py`

- Subscribed rostopic:
    - Camera images from `/usb_cam/image_raw`.
    - How to rosrun usb_cam in Clearpath Jackal:

      ```
      rosrun image_view image_view image:=/usb_cam/image_raw
      ```

- Published rostopic:
    - Publish robot velocity commands on `/cmd_vel`.
    - Publish annotated images for debugging on `/marker_follower/annotated_image`.

- Parameters:
    - Controller parameters:
        - `desired distance`
        - `k_linear`
        - `k_angular`
        - `max_linear_speed`
        - `max_angular_speed`

How to run:

```
rosrun marker_follower marker_follower_ros.py
```

✅ Next Steps

1. Check out `TODO` in each script.
2. Here are suggested next tasks for this marker follower project:

---

1. Try the Example Video

- Run `run.py` with the provided example video:

```
python3 run.py
```

2. Test with Your Own Videos

- Record your own ArUco marker footage.
- Save the video in the `examples/` folder or Modify the video path.
- Update the `video_path` in `run.py`:

```
video_path = "examples/your_new_video.mp4"
```

3. Tune Control Parameters, Find the optimal camera/marker positions

- Optimize the following parameters in both scripts:
  - `desired_distance`
  - `k_linear`
  - `k_angular`
  - `max_linear_speed`
  - `max_angular_speed`
- Observe how these affect:
  - Robot's following distance
  - Smoothness of motions
- Find the best camera mounting position on the robot.
- Find the best marker height/orientation in the field.

4. Improve Camera Calibration

- Replace default `camera_matrix` and `dist_coeffs` with real calibration data.

5. Extend to Multiple Marker IDs

PROF

- Currently, the code tracks **only marker ID 1**.
- Enhance it to:
    - Track multiple IDs simultaneously.
    - Use different control strategies per ID.
    - Stop or switch targets if the desired marker disappears.

6. Add Logging or Data Collection

- Record multiple videos at ACRE to see if the detection performance degrades in the field or not.