University of Missouri

Vision-Guided Intelligent Robotics Lab (ViGIR)

EECS 4340/7340 Building Intelligent (Vision-Guided) Robots

Lab 2

Camera Calibration and Object Detection

Submission Due : 09/23/2025

Objective

In this experiment, students will use stereo cameras, ROS 2 camera calibration tools, and AR tag detection to achieve the following goals:

- Find intrinsic parameters of each camera.
- Calibrate the stereo camera rig to find relative rotation and translation.
- Define the world coordinate system using an AR tag.
- Develop an image processing pipeline to detect objects and compute their (X, Y, Z) coordinates in the world frame.
- Measure calibration and reconstruction errors.

Reference Materials

- Lecture notes on reference frames, homogeneous transformations, and camera models.
- Zhang, Z. *A Flexible New Technique for Camera Calibration.*
- Corke, P. *Robotics, Vision and Control*, Ch. 11–12.
- ROS 2 Camera Calibration package: `https://index.ros.org/p/camera_calibration/`
- ROS 2 AR Tag detection (e.g., `fiducial_slam` or `apriltag_ros`).

Pre-Lab

1. What is the pinhole camera model? Write its equation. How many parameters do we have to estimate in a pinhole camera calibration? How many points do we need to estimate them?
2. Using OpenCV library (Google it!) and classical image processing techniques, write a pseudo-code that detects a black cube on a white background, and determines the coordinates of its centroid.
3. Let's assume that you have detected the centroid of the desired object, how do you use the camera calibration to find the coordinates of the object in real world?

Background

Camera calibration determines the mapping from 3D world coordinates to 2D image coordinates. Intrinsic parameters model the camera, while extrinsic parameters relate the camera to the world frame. In this lab, the world coordinate system is defined by a fixed AR tag placed on the table. The transformation between the camera and the AR tag is estimated using `apriltag_ros`.

The pinhole camera model is:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R \mid T \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

where K is the intrinsic matrix, R and T are rotation and translation (extrinsics), and (u, v) are image coordinates.

Lab Procedure

Part 1: Camera and Stereo Calibration (Week 1)

1. Use ROS 2 `camera_calibration` to compute intrinsic and distortion parameters for each camera.
2. Save calibration results as YAML files (e.g., `left.yaml`, `right.yaml`).
3. Perform stereo calibration using both YAML files to estimate relative pose between cameras.
4. Detect an AR tag fixed to the table. The pose of the AR tag defines the world coordinate system.

To start camera node:

```
# Pick your device path
DEVICE=/dev/video0

ros2 run v4l2_camera v4l2_camera_node \
  --ros-args -p video_device:=${DEVICE} \
             -p image_size:="[1280,720]" \
             -p pixel_format:="YUYV" \
             -p frame_rate:=30.0 \
             -r image_raw:=/camera/image_raw \
             -r camera_info:=/camera/camera_info \
             -r image:=/camera/image_raw
```

To see the images being published:

```
ros2 run rqt_image_view rqt_image_view
```

To start the camera calibration node (make sure to modify the specifications based on the actual calibration pattern that you have available):

```
ros2 run camera_calibration cameracalibrator \
  --size 8x6 --square 0.024 --pattern chessboard --no-service-check \
  --ros-args -r image:=/camera/image_raw -r camera:=/camera
```

after the calibration is done, a tar.gz file will be saved into `/tmp/`. Copy that file into a proper location and decompress it using `tar -xvf [file]`. The `yaml` inside the compressed file is the calibration results.

To do the stereo calibration, you need both cameras to publish under `/left` and `/right` topics. You can do it with this commands:

```
# Left
ros2 run v4l2_camera v4l2_camera_node \
  --ros-args -p video_device:=/dev/video0 \
             -p frame_rate:=30.0 \
             -r image_raw:=/stereo/left/image_raw \
             -r camera_info:=/stereo/left/camera_info

# Right
ros2 run v4l2_camera v4l2_camera_node \
  --ros-args -p video_device:=/dev/video2 \
             -p frame_rate:=30.0 \
             -r image_raw:=/stereo/right/image_raw \
             -r camera_info:=/stereo/right/camera_info
```

Then to run the stereo calibration (again, make sure to modify the specifications based on the actual calibration pattern that you have available)::

```
ros2 run camera_calibration cameracalibrator \
  --size 8x6 --square 0.024 --pattern chessboard --approximate 0.1 \
  right:=/stereo/right/image_raw left:=/stereo/left/image_raw \
  right_camera:=/stereo/right left_camera:=/stereo/left
```

Now that you have calibrated the data, you can relaunch the camera node with the proper calibration info using this command:

```
ros2 run v4l2_camera v4l2_camera_node \
  --ros-args -p video_device:=/dev/video0 \
 -p camera_info_url:="file:///home/$USER/<your_camera_calibration_path>.yaml" \
 -r image_raw:=/camera/image_raw -r camera_info:=/camera/camera_info
```

To do the AR tag detection, first rectify the images that you are publishing using this node:

```
ros2 run image_proc image_proc \
  --ros-args \
  -r image:=/camera/image_raw \
  -r camera_info:=/camera/camera_info \
  -r image_rect:=/camera/image_rect
```

Then create a minimal apriltag parameter yaml file:

```
apriltag:                  # node name
  ros__parameters:
    # setup (defaults)
    image_transport: raw  # image format: "raw" or "compressed"
    family: 36h11         # tag family name: 16h5, 25h9, 36h11
    size: 1.0             # default tag edge size in meter
    profile: false        # print profiling information to stdout

    # tuning of detection (defaults)
    max_hamming: 0
    detector:
      threads: 1
      decimate: 2.0
      blur: 0.0
      refine: true
      sharpening: 0.25
      debug: false

    pose_estimation_method: "pnp"

    # (optional) list of tags
    # If defined, 'frames' and 'sizes' must have the same length as 'ids'.
    tag:
      ids:    [1]          # (MAKE SURE TO USE THE PROPER ID)
      frames: ["1"]   # frame names
      sizes:  [.144]
```

Then:

```
ros2 run apriltag_ros apriltag_node \
  --ros-args \
    --params-file apriltag_params.yaml \
    -r image_rect:=/camera/image_rect \
    -r camera_info:=/camera/camera_info
```

You should be able to see the transformations under the `/tf` topics.
Visualize the transformation in RVIZ.

Part 2: Object Detection (Week 2)

1. Place black cubes on the table. Use white paper for background contrast.
2. Acquire images from both cameras (subscribe to the camera node).
3. Apply an image processing pipeline: thresholding, component labeling, edge detection, and Hough transforms.
4. Extract centroid (u, v) for each object.
5. Using calibration parameters and stereo geometry, reconstruct (X, Y, Z) in camera coordinates.
6. Transform coordinates into the world frame using the AR tag pose.

Post-Lab Questions

1. Derive the equations used to compute (X, Y, Z) from stereo correspondences.
2. How does using an AR tag simplify defining the world coordinate system?
3. How do shadows, different object sizes, or occlusion affect detection accuracy?

Post-Lab Accuracy Reporting

- Report reprojection error of calibration for both cameras. Devise an experiment to measure that (hint: placing the cubes on multiple known positions ...). Report the error in both (u, v) and (X, Y, Z).
- Discuss possible causes of error (e.g., lens distortion, stereo matching, AR tag detection noise).