TORONTO METROPOLITAN UNIVERSITY
Department of Aerospace Engineering
AER627 – Introduction to Space Robotics
Dr. Anton de Ruiter
Project 1
Due: Week of Feb. 3, 2025 by 5pm the day of your lab

Overview

This is the first robotics project. It serves as an introduction to the Mindstorms, computer vision, the application of robotics theory, and the general structure of the labs. I've tried to make the actual deliverables for this project a little lighter than 'normal', recognizing that you will be spending your time acquainting yourselves with how to use these kits. Each project will have a theme — this one's theme is 'learn how to use this stuff'. Please take some extra time and try working with the kits, even beyond the explicit requirements spelled out in this document.

It is vitally important that both partners make an effort to learn **all** the lab material (working with the vision system, programming the Lego) — you will be working together on all aspects of the project.

Tools and Resources

Each group will be issued Lego Mindstorms EV3 Kit. This contains structural components, sensors, and actuators, as well as the computer module (the 'brick'). Please try to take good care of your kits as we will need them back. Report any broken or missing pieces to Cliff, Joel or Nima (we won't hold this against you, but we'd like to keep tabs on any losses).

This year we will be programming the Mindstorms exclusively with Matlab. You can install Matlab on your own laptops for free. Make sure you install:

- *Image Acquisition Toolbox*. This is a primary toolbox from the main installer.
- *Image Acquisition Toolbox Support for OS Generic Video Interface.* This is a secondary add-on (see 'Manage Add-ons')
- *MATLAB Support Package for USB Webcams*. This is another secondary add-on

You will also need access to a camera and means of holding it in place (e.g., mini-tripod, selfie stick, or improvised version thereof). Instructions are provided a little later on how to use your mobile phone as a camera, but you may also use a dedicated webcam or digital camera (provided that they can be accessed from withing Matlab)

In addition to the physical kits, we recommend downloading and installing one of the Lego CAD packages. BrickLink Studio is the preferred choice, but you may also want to explore LDraw.

- `https://www.bricklink.com/v3/studio/download.page`
- `https://www.ldraw.org/`

Tutorials

The LEGO kits contain a number of tutorials and sample projects. Work through some of the sample projects to understand how to assemble components, hook up sensors and actuators, and use Simulink to help program the robots. You should probably be able to proceed pretty quickly through these tutorials. I would encourage you to get a handle on the programming environment. It's intended to be simple to pick up, but it's different than many low-level environments. Look at the samples and see if you can understand how they work. You can also try building a very simple robot (maybe nothing but motors and sensors hooked up to the brick). See if you can create programs from scratch that exhibit particular behaviours.

You should also try working with BrickLink. Start with some of the simple examples and try see how you can document the models that you build. Practice exporting figures to other software like Microsoft Word.

Building Assignment

After you have worked through enough of the tutorials to be comfortable with the kit, you can move onto the actual assignment for this project. Each project will have a section like this one that describes the robot that you need to build. We may include some example sketches or photographs of robots that we have built. In most cases they are intended to be illustrations, not building instructions. Feel free to improve on these designs (and don't try to get yours to be exactly the same). We'll also describe any software or hardware that we've put together to make your lives easier.

For the first project you will need to build and operated the following robots:

A. A one-joint rotary actuator with a different motion ratio than the base motor. You will need to use gears or some other mechanism to change the motion ratio. Program the robot so it can be commanded in terms of the ***output angle*** in degrees. Do this programming using Matlab. The 'robot' must include an automated mechanism for 'homing' the mechanism. The encoders in the joints record relative motion, but in order to establish absolute motions you will need to incorporate a limit switch into your design. Upon initializing the robot, a homing routine starts moving the mechanism slowly in a known direction. When the switch is triggered, the software assumes that the robot is at a fixed absolute position. Typically this is either the minimum or maximum position, but you could put a limit switch in the middle of the range. Any further motions can be computed relative to this known position. Your design should try to achieve uniform performance over 360° of rotation (or as near to that as possible).

B. A prismatic joint. There are a variety of ways you can get linear motion out of the Lego components. You may look online for ideas, but please credit any sources that you use. Program the robot so it can be commanded in terms of the ***output position*** in centimeters. This robot must also include a homing mechanism (and software routine). Your priority should be to have the most *precise* and *repeatable* motion possible subject to the constraint that you need to be able to measure the performance. Comment on you achieved this.

See the *Operations* section for insight into what you have to do with these simple robots.

Vision Assignment

In this part of the project you will perform a camera calibration using Matlab's Camera calibration toolbox and then use the calibrated camera to recognize objects in a structured scene in view of the came. You will also be working with a popular fiducial marker system called AprilTags (more info at `https://april.eecs.umich.edu/software/apriltag`).

You may use any camera that you have access to: your phone, a webcam, or a regular camera. Whichever you chose, the camera must be accessible from within Matlab. We have instructions here on using a mobile phone as a webcam; if you are working with a different type of camera, you will have to adapt these instructions.

A. Connect your camera to your computer using any available app like camo, DroidCam or similar. You can directly use the camera however, you will need to manually upload the images to access them in Matlab.

B. If using an app, verify that you can run the client and access the image feed from your camera. You can access the camera from within Matlab's 'Image Acquisition' app.

C. To calibrate your camera you will need to print out a checkerboard calibration target. Follow the instructions in the Matlab documentation and take a sequence of pictures of the calibration target. You can do this offline or directly with the Camera Calibrator app (within Matlab, this requires a direct connection via the preinstalled app). Comment on any trends in the residuals as you increase the number of calibration images.

D. Compare your calibration parameters with your partner. How do the cameras compare (what are the relative distortion coefficients look like)?

E. Copy the vision directory from the shared Matlab Drive (see D2L for link). **Make sure you extract the files from the zip**. Verify that the code is working by running the following.

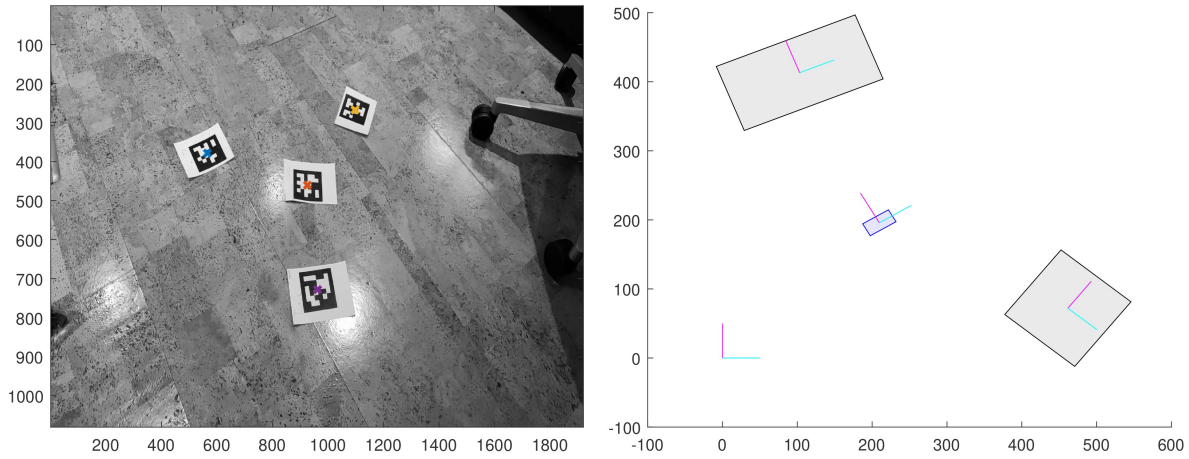
```
apriltag_test.m
```

The returned structure contains the ID number of the detected tags (e.g., `tags(1).id`), as well as their position (`tags(1).p`) and orientation (`tags(1).R`) relative to the camera.

F. Print out one (or more) of the AprilTags. There's a PDF in the vision directory with four AprilTag images in a one-page document. There's another PDF archive that has a whole family of possible tags if you would like more options, but all tags in the image must be of the same size. Note: You probably don't want to print the whole document as it is several hundred pages long. Attach the tag to a flat surface and measure the size of the tag with a ruler.

G. Adapt the `setup_script.m` based on the calibration structure for your camera. Position the tags in view and using the setup_script *as a guide,* capture and interpret the tags thate are in view. The idea with this setup is that one tag will be used to mark the origin, and the other tags will represent objects in the scene. You can get the pose relative to the origin by looking at the homogeneous transforms for the apriltag-objects (e.g., `lv.vehicle_tag.get_pose`). You should get two figures like the ones shown in 1.

H. Devise an experiment to measure the rotational and positional accuracy of the pose estimates. How far away can you get a positive read on the tags?



(a) (b)

Figure 1: Illustration of the vision processing system including a raw bitmap image, showing marked apriltags (a), and a processed schematic version of the same scene (b). Note that the origin tag is on the left.

We'll be using this software again later in the semester, but in the short term your primary objectives are to calibrate your cameras and evaluate the accuracy of the scene decomposition.

Problems and Analysis

Copy the Livescript example from the Matlab Shared Drive. That file contains the analysis problems required for this project. Remember *each* student must complete this individually.

Design Documentation

This section is pretty simple for this project. In future projects we may be a little more explicit about exactly what we want to see. The basic principal is that you want to explain how you arrived at your design. You can highlight parts of the design based on analysis or on trial-and error (what works and what didn't). You will probably have elements of both.

Package-A (Mindstorms)

For the student with the Mindstorms kit, document the design of *one* of the actuators that you built (see Build Assignment). Use appropriately chosen figures from LDD/LDRAW to show how it works. Show your program for the robot and describe how it works. Please discuss any planned features of the design (e.g., gear ratios, mechanism behaviour, etc.). Note: You still need to build, program and evaluate both actuators, but you only need to provide detailed documentation for one of them.

Package-B (Vision)

Document the camera calibrations carried out by you and your partner. Compare the calibration constants between the two cameras and comment on the residual calibration errors. Which camera

do you expect to give better performance? Describe and document the scene calibration procedure (with the Apriltags). Explain how you will assess translational and rotational accuracy.

Project Operations

A. For the two simple robots that you built, how would you assess accuracy and repeatability of the manipulator motion in response to commanded motion? I.e., if you asked for 10°, do you get exactly 10°, or do you get 11°? Is the motion biased? Devise some simple experiments to measure how well these simple manipulators perform and discuss your findings. Do some experiments with both actuators. How repeatable is your homing mechanism and software? I.e., once you have homed the mechanism once, how much variability do you see in the platform position when the limit switch is triggered again? Are there any changes to the software or hardware design that can improve the performance?

B. Evaluate the precision and accuracy of the image analysis software. Is there much temporal variation in the reported positions (with no motion)? How uniform is the motion measurement throughout the field of view? Can you estimate the accuracy (rotation and translation) of your system? Document any strategies that you may have tried to improve the accuracy and consistency of the vision system.