TTK4255 Robotic Vision, Spring 2020

Exercise 1: Image Formation and Rigid-Body Modeling

Authored by Simen Haugo (simen.haugo@ntnu.no)

Due: See Blackboard

This exercise is a mix of programming problems and theory questions. You are free to use any programming language, but solutions will be provided in Python and Matlab, and we may not be able to help you effectively if you use a different language. We therefore recommend that you use either Python or Matlab. You should also do exercise 0 to get familiar with language features and libraries that are useful for the exercises.

Instructions

This exercise is **approved** / **not approved**. To get your exercise approved, submit your answers to the questions as a **PDF** to Blackboard. You **don't** have to submit your code. Feel free to use LaTeX, Word, handwritten scans, or any other tool to write your answers.

Getting help

If you have questions about the exercise, you can:

- Post a question on Piazza (you can post anonymously if you want)
- Ask for help during the tutorial sessions (see Piazza for time and place)

Relevant resources

Below are the relevant slides and chapters from the course syllabus for this exercise (the course material is accessible on Piazza under Resources).

- Lecture 1 and 2
- *Robotics, Vision and Control 2nd Edition.*
  - Ch. 2.1-2.2: Representing position and orientation
  - Ch. 11.1: Perspective camera
- *An Invitation to 3D Vision.*
  - Ch. 2: Representation of a three-dimensional moving scene
  - Ch. 3: Image formation

Both textbooks cover the same material, but Corke's book might be easier to read.

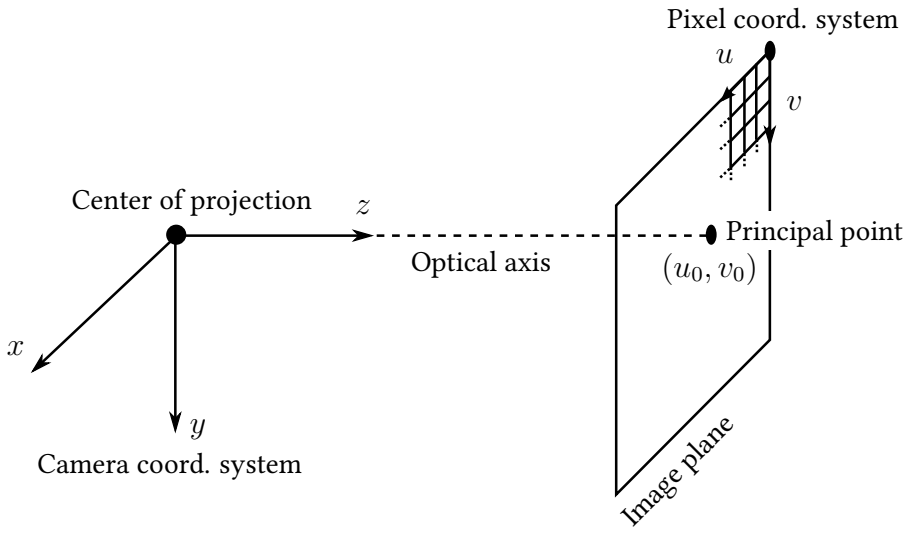

Figure 1: Pinhole camera model

1 Image formation

The pinhole camera model is commonly used in computer vision. You may encounter different formulations of the model, but in this exercise we will assume that a point in camera coordinates (X, Y, Z) is mapped to pixel coordinates (u, v) as follows:

$$u = c_x + d_x^{-1} f \frac{X}{Z} \tag{1}$$

$$v = c_y + d_y^{-1} f \frac{Y}{Z} \tag{2}$$

Here, the focal length f is the distance between the center of projection and the image plane. d_x and d_y is the distance between adjacent pixels in the horizontal and vertical directions, respectively, and are usually equal to each other. (c_x, c_y) is where the optical axis intersects the image plane and is called the principal point. For brevity we define

$$f_x = d_x^{-1} f \quad \text{and} \quad f_y = d_y^{-1} f \tag{3}$$

as the "horizontal and vertical focal length", although this is not physically meaningful.

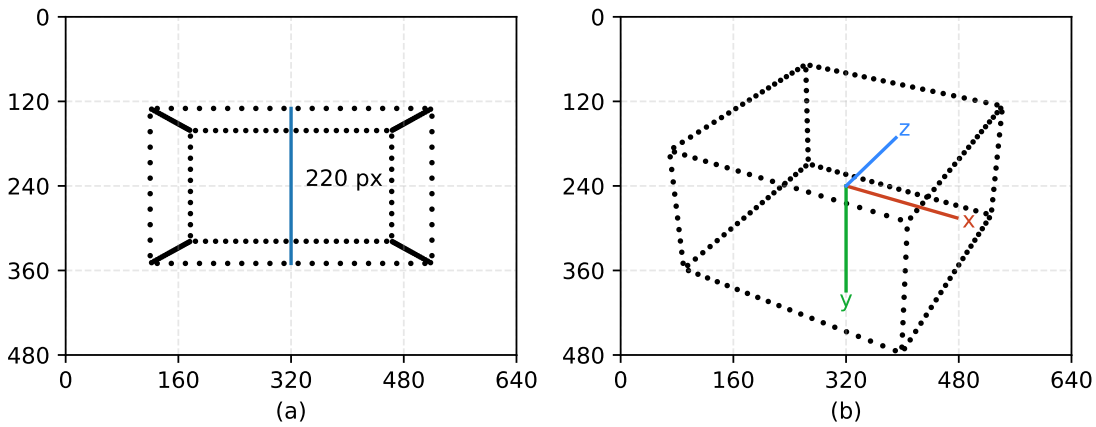

Figure 2: (a) Image of a box object. (b) Rotated box and the object coordinate frame.

(a) Fig. 2a is an image of a box from a simulated pinhole camera with parameters

$$f_x = 1000 \quad f_y = 1100 \quad c_x = 320 \quad c_y = 240 \tag{4}$$

The box is 1 meter tall in physical units, and 220 pixels tall in the image as indicated by the vertical line passing through the center. Using equations (1)-(2), determine how far away the closest face of the box is from the camera.

Solution: Since the line is vertical and passes through the center, we have from (2) a relationship between the v-coordinate of the points on the line in the image, and the corresponding top and bottom of the box in camera coordinates:

$$220 = v_{\text{bottom}} - v_{\text{top}} = f_y \frac{Y_{\text{bottom}}}{Z_{\text{bottom}}} - f_y \frac{Y_{\text{top}}}{Z_{\text{top}}} \tag{5}$$

Since the distance to the camera is the same for both points, $Z_{\text{bottom}} = Z_{\text{top}} = Z$, we can simplify this to

$$220 = f_y \frac{Y_{\text{bottom}} - Y_{\text{top}}}{Z} \tag{6}$$

Since we know the dimensions of the box, $Y_{\text{bottom}} - Y_{\text{top}} = 1$, and the camera parameters $f_y = 1100$, we have

$$Z = \frac{1100}{220} = 5 \tag{7}$$

(b) Write a program to verify your answer in (a) by reproducing Fig. 2a. Include your figure in your submission.

The provided file `box.txt` contains an array of points in the box coordinate frame. The box is $2 \times 1 \times 2$ meters and the points are relative to its center. The points must be translated by an appropriate amount to obtain their camera coordinates, before applying the projection equations.

Tip: You can load the point data into an n x 4 array using `load('box.txt')` (Matlab) or `numpy.loadtxt('box.txt')` (Python).

Solution: Note that since the box points are referenced in the box's center, and we found the distance to the front face in the previous task, we need to add 1 to get the correct distance from the camera center and the box center.

(c) Define the following homogeneous transformation matrices:

$$\mathbf{T}_z(t_z) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R}_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

In Fig. 2b the box is at the same position as in (a) but rotated around its origin. Here, the transformation $\mathbf{T}_o^c$ from object to camera coordinates is a product of *one* translation along z and *two* rotations, by the same angle $\theta = 30°$. For example,

$$\mathbf{T}_o^c = \mathbf{R}_z(\theta)\mathbf{T}_z(t_z)\mathbf{R}_y(\theta) \tag{8}$$

is a valid, but incorrect, transformation. Find the correct expression for $\mathbf{T}_o^c$.

Solution: The correct transformation is

$$\mathbf{T}_o^c = \mathbf{T}_z(6)\mathbf{R}_x(30°)\mathbf{R}_y(30°) \tag{9}$$

(d) Write a program to verify your answer in (c) by reproducing Fig. 2b. Include your figure in your submission.

You need to apply the correct transformation before projecting the points using equations (1)-(2). Additionally, write a function that takes a transformation matrix $\mathbf{T}_*^c$ and visualizes the coordinate frame. Use it to draw the object coordinate frame as in the figure.

2 Rigid-body modeling

You may already be familiar with the Quanser 3-DOF Helicopter from previous courses. In this part, you will create a piecewise-rigid model of the helicopter and verify your model by superimposing it on a photograph.

Figure 3: Quanser 3-DOF helicopter and its three degrees of freedom. The arrows indicate the direction of increasing yaw, pitch and roll.

The helicopter consists of an arm with a counterweight and two rotors. As the name suggests, it has three degrees of freedom, indicated in Fig. 3:

- *Yaw* (ψ): Rotation around an axis perpendicular to the base platform
- *Pitch* (θ): Rotation of the arm up or down
- *Roll* (ϕ): Rotation of the rotor carriage around the arm

In a later exercise you'll implement a visual tracking algorithm to estimate the yaw, pitch and roll from fiducial markers[1] attached to the helicopter. In order to do this, we first need to create a mathematical model that describes how points on the helicopter (which are known) and points in the image (which can be observed) are related.

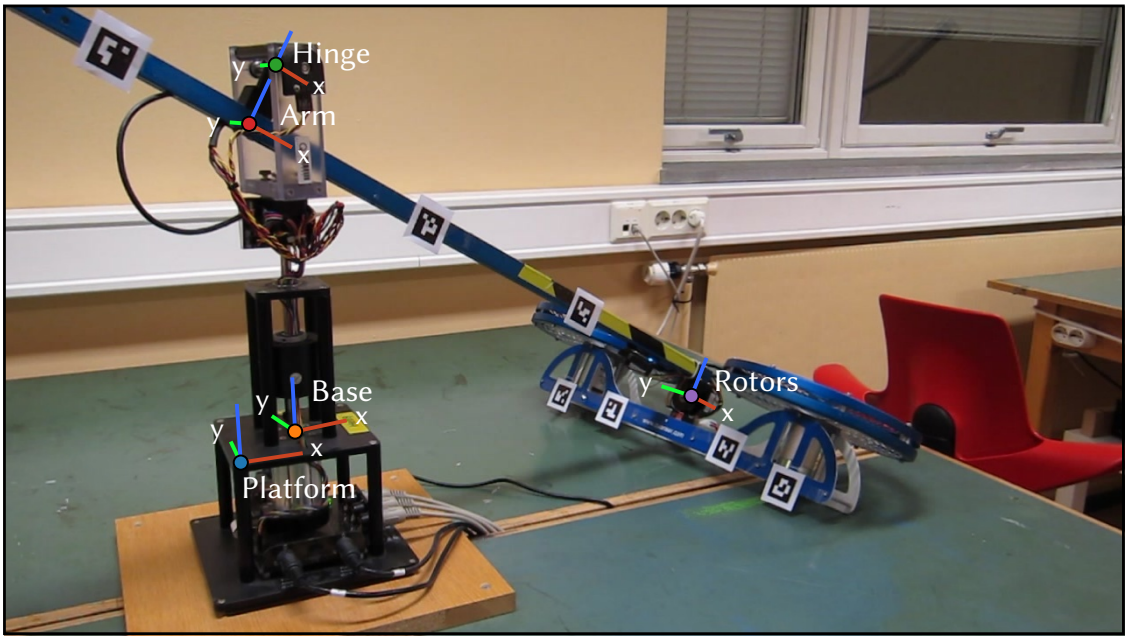

Figure 4: Helicopter coordinate frames

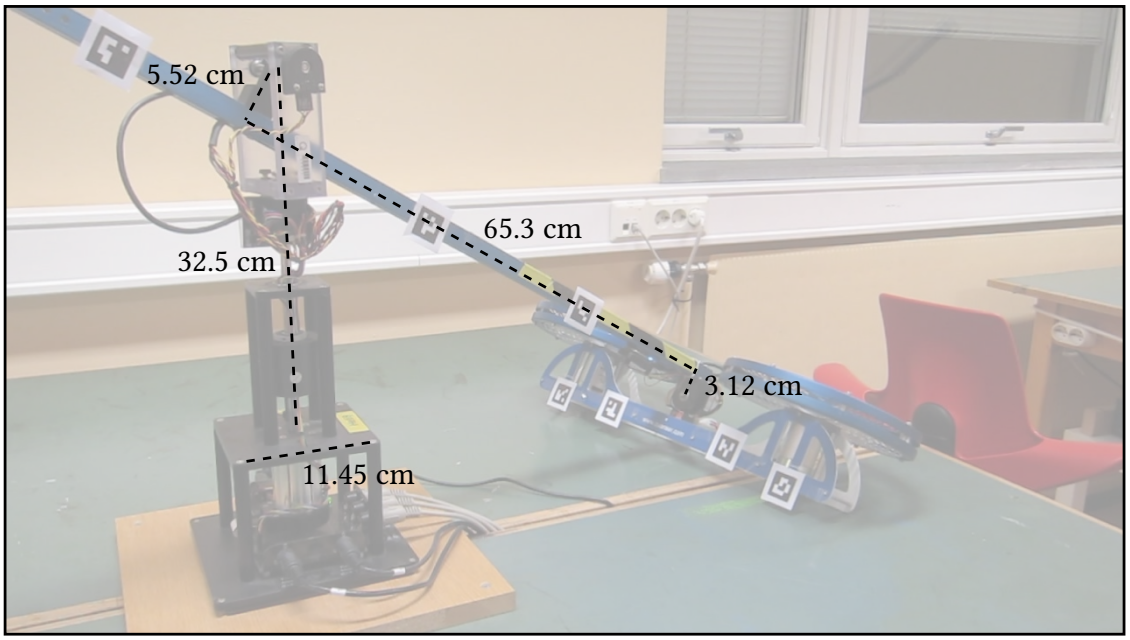

Figure 5: Helicopter dimensions

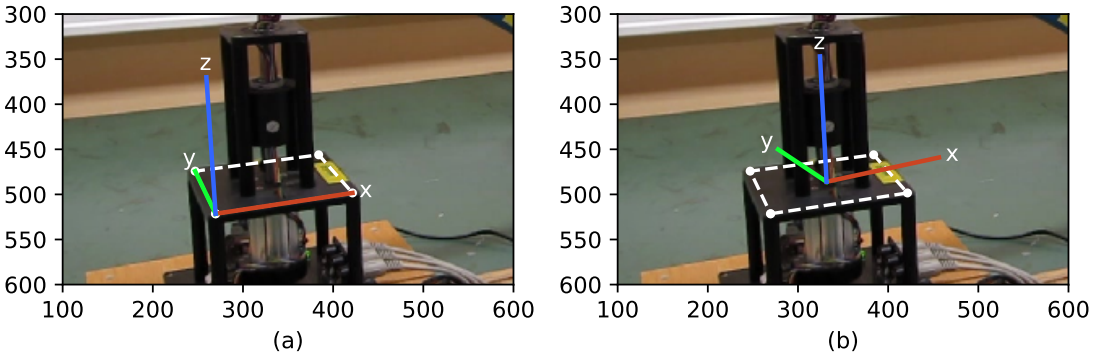

Figure 6: (a) Platform coordinate frame and points located on the four screws. (b) Base coordinate frame, located in the platform center with the z-axis perpendicular to the platform.

Important coordinate frames and dimensions are indicated in Fig. 4 and Fig. 5. Creating the mathematical model involves putting several pieces together correctly, which we'll do step-by-step.

(a) A platform with four screw holes is shown close-up in Fig. 6. The platform is square and the distance from screw to screw is 11.45 cm. Create a model of the platform by defining four 3D points corresponding to the screw locations. The points should be expressed in the coordinate frame shown in Fig. 6a, which is aligned with the sides of the platform and has its origin on one of the screws.

Solution: In the coordinate frame indicated in Fig. 6a, the screw locations have

coordinate vectors:

$$\boldsymbol{p}_1^{\text{platform}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \boldsymbol{p}_2^{\text{platform}} = \begin{pmatrix} 0.1145 \\ 0 \\ 0 \end{pmatrix} \tag{10}$$

$$\boldsymbol{p}_3^{\text{platform}} = \begin{pmatrix} 0.1145 \\ 0.1145 \\ 0 \end{pmatrix} \qquad \boldsymbol{p}_4^{\text{platform}} = \begin{pmatrix} 0 \\ 0.1145 \\ 0 \end{pmatrix} \tag{11}$$

(b) Verify that your model in (a) is correct by reproducing Fig. 6a:

1. Assume the camera follows equations (1)-(2) and use the camera parameters provided in `heli_intrinsics.txt`.
2. Load and draw the image `quanser.jpg`.
3. Use the platform-to-camera matrix $\mathbf{T}_{\text{platform}}^{\text{camera}}$ provided in `heli_pose.txt`[2].
4. Project and draw the four model points.
5. Draw the platform coordinate frame using your function from task 1d.

If your model is correct, the projected points should coincide with the screw holes. Include your figure in your submission. You don't have to reproduce Fig. 6 exactly, e.g. the labels and stippled lines can be omitted.

Tip: The matrix can be loaded from file using `numpy.loadtxt('heli_pose.txt')` *in Python or* `load('heli_pose.txt')` *in Matlab.*

(c) Yaw motion is enabled by a shaft passing through the platform center. The "base" frame follows the shaft: Its z-axis is parallel to the platform z-axis and its origin is in the middle of the platform. The remaining axes are obtained by a counter-clockwise rotation by ψ around z.

Find an expression for the transformation $\mathbf{T}_{\text{base}}^{\text{platform}}(\psi)$ from the base frame to the platform frame.

Solution: Denoting

$$\mathbf{T}_t(x, y, z) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

then the correct transformation is

$$\mathbf{T}_{\text{base}}^{\text{platform}}(\psi) = \mathbf{T}_t(0.1145, 0.1145, 0)\mathbf{R}_z(\psi) \tag{13}$$

(d) Pitch motion is enabled by a hinge at the top attached to the shaft. The "hinge" frame is obtained by translating 32.5 cm along the base frame z-axis (i.e. up the shaft), and rotating counter-clockwise by θ around the y-axis.

The arm is rigidly attached to the hinge. As indicated in Fig. 4, the "arm" frame is centred in the arm cross-section with its x-axis pointing down the arm toward the rotors. It is obtained by translating -5.52 cm along the hinge frame z-axis. Assume that both the hinge and the arm frames are aligned with the base y-axis.

Find an expression for $\mathbf{T}^{\text{base}}_{\text{hinge}}(\theta)$ and $\mathbf{T}^{\text{hinge}}_{\text{arm}}$.

Solution:

$$\mathbf{T}^{\text{base}}_{\text{hinge}}(\theta) = \mathbf{T}_t(0, 0, 0.325)\mathbf{R}_y(\theta) \tag{14}$$

$$\mathbf{T}^{\text{hinge}}_{\text{arm}} = \mathbf{T}_t(0, 0, -0.0552) \tag{15}$$

(e) Finally, the rotor carriage is allowed to rotate around a shaft parallel to the arm, located slightly below the arm centerline. The "rotors" frame is obtained from the arm frame by translating 65.3 cm along the x-axis, -3.12 cm along the z-axis, and rotating counter-clockwise by ϕ around the x-axis.

Find an expression for $\mathbf{T}^{\text{arm}}_{\text{rotors}}(\phi)$.

Solution:

$$\mathbf{T}^{\text{arm}}_{\text{rotors}}(\phi) = \mathbf{T}_t(0.653, 0, -0.0312)\mathbf{R}_z(\phi) \tag{16}$$

(f) Verify your answers by drawing all coordinate frames together as in Fig. 4. Use

$$\psi = 11.77^\circ \quad \theta = 28.87^\circ \quad \phi = -0.5^\circ \tag{17}$$

Load the model-relative fiducial marker coordinates in `heli_points.txt`. The first three are in the arm frame, and the last four are in the rotors frame. Project and draw the points by applying the correct transformation. Your figure should look like Fig. 7. Include your figure in your submission.

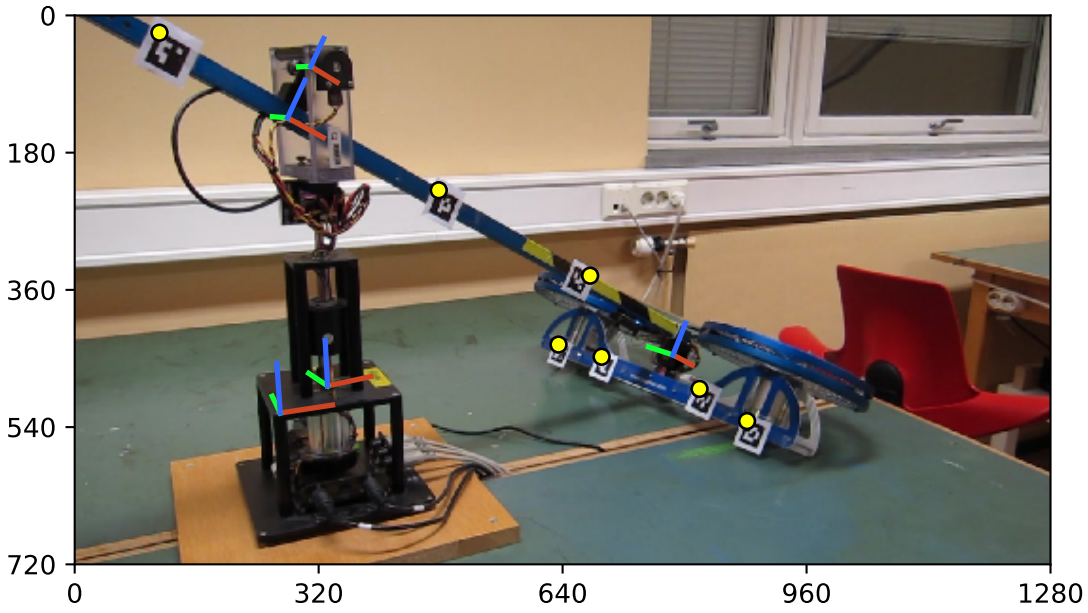

Figure 7: Helicopter frames and reprojected fiducial marker points

Notes

[1]A fiducial marker is an object placed in the scene that can be easily recognized by a computer vision system. The markers you see on the helicopter are called AprilTags. These are easily detected in images thanks to their appearance and they provide a unique ID for each marker.

[2]This was estimated using pose estimation techniques, which you will learn about later in the course and implement yourself in a later exercise.