TTK4255: Robotic Vision

# Homework 3: Geometric image formation

Originally created by Simen Haugo, and modified by Mau Hing Yip.



## Instructions

First make sure to read `about.pdf` in the "Course work" page on Blackboard. To get your assignment approved, you need to complete any 60%. Upload the requested answers and figures as a single PDF. You may collaborate with other students and submit the same report, but you still need to upload individually on Blackboard. Please write your collaborators' names on your report's front page. If you want detailed feedback, please indicate so on the front page.

## Recommended literature

This assignment assumes you have received an introduction to rigid transformations in a previous course, e.g. TTK4130 or TTK4190. If you are currently taking a relevant course, you may want to read ahead in the book used there. If you need a refresher, you can read §2.1.1–§2.1.3 in Szeliski, or §7–§8 in Förstner and Wrobel.

The perspective camera model is presented in Szeliski §2.1.4, Förstner and Wrobel §6+§11.1+§12.1, and Hartley and Zisserman §6. However, this assignment has a summary of the most important parts needed to solve the assigmnent, written with the notation that we will use throughout the course.

Homogeneous coordinates are given a detailed introduction and overview in Förstner and Wrobel §5 (in particular §5.1.2), but we will return to the various applications of homogeneous coordinates later in the course. The brief summary provided in the assignment should be all you need for now.

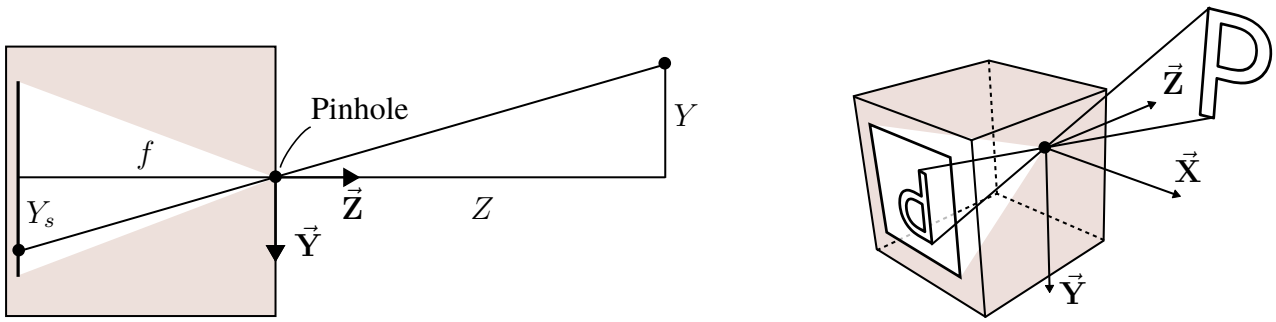


Figure 1: Diagram of an ideal pinhole camera seen from the side and in 3D.

## The perspective camera model

A pinhole camera is a box with a small opening (the pinhole) through which light can enter and be recorded on a surface (e.g. a digital CMOS sensor or a photosensitive film). The *ideal pinhole camera* is a theoretical device in which the opening is infinitely small, with the effect that each point on the sensor receives light from just one direction in the scene. Let $f$ be the distance between the pinhole and a planar sensor in an ideal pinhole camera. The relationship between the coordinates $(X, Y, Z)$ of a point in the scene and the coordinates $(X_s, Y_s, Z_s)$ of the observed point on the sensor can be derived by a consideration of similar triangles (see Fig. 1):

$$\frac{X}{Z} = \frac{-X_s}{f} \Rightarrow X_s = -f\frac{X}{Z} \tag{1}$$

$$\frac{Y}{Z} = \frac{-Y_s}{f} \Rightarrow Y_s = -f\frac{Y}{Z} \tag{2}$$

and finally $Z_s = -f$. These equations will differ based on how you place the camera's axes. This course uses the convention in which positive $Z$ is forward, positive $X$ is to the right, and positive $Y$ is downward, forming a right-handed coordinate system. Regardless of the chosen convention, these equations imply that the image is rotated 180 degrees compared with what you would see through the pinhole. This inversion also occurs in a modern digital camera, but the firmware and your operating system work together to ensure that the rotation is undone when the image is displayed.

We seldom measure coordinates on the physical sensor surface — which the above equations describe. Instead we measure pixel coordinates in the recorded digital image. With few exceptions, the pixel coordinate system is placed in the upper left corner of the upright image, with the horizontal axis pointing to the right in the scene and the vertical axis pointing down. Let $(u, v)$ be the horizontal and vertical pixel coordinates. The relationship between sensor coordinates $(X_s, Y_s)$ and pixel coordinates $(u, v)$ is often modeled by a negation, scaling, and offset:

$$u = c_x - s_x X_s \quad \text{and} \quad v = c_y - s_y Y_s, \tag{3}$$

which can be simplified and written in terms of the original 3D point coordinates as:

$$u = c_x + s_x f\frac{X}{Z} \quad \text{and} \quad v = c_y + s_y f\frac{Y}{Z}. \tag{4}$$

Note that the negation is mathematically equivalent to placing the sensor in front of the camera (i.e. at $Z = f$), but this makes no sense physically, and misleadingly suggests that points with $Z < f$ are not visible. The parameters $c_x, c_y, s_x, s_y, f$ are given the following names:

- $c_x, c_y$: *Principal point* — the image coordinates where the $\vec{\mathbf{Z}}$ axis intersects the sensor.
- $s_x, s_y$: *Pixel density* — the number of discrete sensing elements horizontally and vertically per unit length of the sensor (often in pixels per micrometer).
- $f$: *Principal distance* — the distance between the pinhole and the sensor (often in millimeters).

The principal point should not be confused with the *optical center* or *center of projection*, which is the point of origin of the rays going back into the scene. In the ideal pinhole camera, the center of projection coincides with the pinhole. A real camera will not have a single point that can be identified as the center of projection, but one may more or less approximate it as such.

You may see $s_x$ and $s_y$ as their inverse quantities, which measure the distance between horizontally and vertically adjacent pixels on the sensor, respectively. In a sensor datasheet, these are usually specified as the *pixel size* or *pitch* in micrometers (microns). In modern digital cameras, pixels are usually square ($s_x = s_y$), so you may only see a single number listed.

The mapping from $(X, Y, Z)$ to $(u, v)$ in Eq. 4 is a form of perspective (or rectilinear) projection, and has the property of preserving straight lines. The most general form of a perspective projection can be obtained by adding just one more term to the $u$ coordinate:

$$u = c_x + s_x f X/Z + s_\theta Y/Z, \quad \text{and} \quad v = c_y + s_y f Y/Z, \tag{5}$$

where the new parameter $s_\theta$ describes non-perpendicularity in the pixel grid axes, and is often zero for a modern sensor. This general formulation, and specializations of it with parameters set to zero, are often referred to as instances of a pinhole/perspective camera model/projection. Rectilinear projection is sometimes used synonymously, and emphasizes the property that straight lines are preserved.

A modern digital camera is also roughly a box with an opening at one end, which allows light to enter and reach a sensor. The difference is that the opening is made large to collect more light, and a lens is placed in front of the opening to focus the light. Still, the mapping of points in the scene to observed points in the image can often be approximated well by a perspective camera model. This is somewhat surprising, considering that modern cameras have complex optics, but not so surprising when one understands that lens designers often (though not always) strive to produce a rectilinear projection.

In a high-quality camera, the perspective camera model may be a sufficiently accurate approximation. Otherwise, small deviations can often be modeled, and points in the image (or the whole image itself) can be transformed so as to satisfy a perspective projection. Some cameras, e.g. fisheye lens cameras, may not permit a correction to a perspective projection — at least not of the entire image — and may thus warrant the use of more general algorithms derived for a different camera model.

## Homogeneous coordinates

The motivation for using homogeneous coordinates can perhaps only be appreciated after seeing them used in different contexts, which this and later homework assignments will hopefully provide.

Syntactically, we will denote homogeneous coordinates using the tilde "$\sim$" symbol. Semantically, defining a coordinate vector to be homogeneous means that we consider any non-zero scalar multiple of the vector to be equivalent, in the sense that they represent the same point in Euclidean space.

A homogeneous coordinate vector has dimension one more than the Euclidean space of the point it represents. When the last component of this vector is not zero, the Cartesian coordinates of the point it represents in Euclidean space are obtained by dividing each component by the last, and dropping the last component:

$$\tilde{\mathbf{u}} = \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \Rightarrow \qquad \mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \tilde{u}/\tilde{w} \\ \tilde{v}/\tilde{w} \end{bmatrix}, \text{ if } \tilde{w} \neq 0.$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} \Rightarrow \qquad \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \tilde{x}/\tilde{z} \\ \tilde{y}/\tilde{z} \end{bmatrix}, \text{ if } \tilde{z} \neq 0.$$

$$\tilde{\mathbf{X}} = \begin{bmatrix} \tilde{X} \\ \tilde{Y} \\ \tilde{Z} \\ \tilde{W} \end{bmatrix} \Rightarrow \qquad \mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \tilde{X}/\tilde{W} \\ \tilde{Y}/\tilde{W} \\ \tilde{Z}/\tilde{W} \end{bmatrix}, \text{ if } \tilde{W} \neq 0.$$

When the last component is zero, the point is infinitely far away from the origin of the Euclidean space (we say the point is "at infinity"), and its Cartesian coordinates are $\pm\infty$. If the last component is zero, but not all components are zero, then the homogeneous coordinate vector excluded the last component can be interpreted as a (not necessarily unit-length) direction vector in Euclidean space.

Notation-wise, we will use the same symbol (and the same sub- and superscripts, if present), but without the tilde, to indicate the associated Cartesian coordinates, and vice versa. We will not always make this explicit in the text. Thus, if we define $\tilde{\mathbf{X}}_j^i$ in the text, then we may later refer to $\mathbf{X}_j^i$ without having explicitly defined it, but with the implicit assumption that it is related to $\tilde{\mathbf{X}}_j^i$ as described above — with awareness that the Cartesian coordinates may be infinite.

Likewise, if we define $\mathbf{X}_j^i$, then we may later refer to $\tilde{\mathbf{X}}_j^i$ without having explicitly defined it, but with the implicit assumption that it is some homogeneous coordinate vector for the same point. Given a Cartesian coordinate vector, a corresponding homogeneous coordinate vector can always be obtained by appending $1$ (exercise: why can't you append any other constant?).

Dividing a homogeneous vector by its last component is often called dehomogenization or Euclidean normalization. You may sometimes see the term projective coordinates, which is synonymous with homogeneous coordinates. You may also see the term inhomogeneous or non-homogeneous coordinates, which is synonymous with Cartesian coordinates (but emphasizes that they are obtained by dehomogenizing some homogeneous coordinates).

## Rigid transformations

A coordinate vector is meaningless without a coordinate system (aka frame). So far, $(X, Y, Z)$ has been a point's coordinates in the camera frame ("camera coordinates" for short). However, a point may be expressed in a different frame. If so, we need a transformation that relates this frame to the camera frame, so that we can compute its projection. Such a transformation is called a rigid transformation (or Euclidean transformation), and consists of a rotation and translation.

We use superscript to indicate the frame a coordinate vector refers to, unless it is unambiguous from the context. Given $\mathbf{X}^o$ (in some frame $o$), the same point's coordinate vector in frame $c$ is

$$\mathbf{X}^c = \mathbf{R}_o^c \mathbf{X}^o + \mathbf{t}_{o/c}^c, \tag{6}$$

where $\mathbf{R}_o^c$ is a rotation matrix and $\mathbf{t}_o^c$ is a translation vector (the origin of $o$ expressed in $c$). This can also be written as

$$\tilde{\mathbf{X}}^c = \mathbf{T}_o^c \tilde{\mathbf{X}}^o \quad \text{where} \quad \mathbf{T}_o^c = \begin{bmatrix} \mathbf{R}_o^c & \mathbf{t}_o^c \\ \mathbf{0} & 1 \end{bmatrix}. \tag{7}$$

The $4 \times 4$ matrix $\mathbf{T}_o^c$ transforms a coordinate vector referring to $o$ so as to instead refer to $c$. For short, we'll say it is the "transformation from $o$ to $c$".

Using $4 \times 4$ matrices lets us combine sequential transformations by matrix multiplication. For example, given the transformation from $a$ to $b$, and from $b$ to $c$, the composite transformation from $a$ to $c$ is $\mathbf{T}_a^c = \mathbf{T}_b^c \mathbf{T}_a^b$. The equivalent in terms of the constituent rotation matrices and translation vectors is

$$\begin{aligned} \mathbf{R}_a^c &= \mathbf{R}_b^c \mathbf{R}_a^b, \\ \mathbf{t}_a^c &= \mathbf{R}_b^c \mathbf{t}_a^b + \mathbf{t}_b^c. \end{aligned}$$

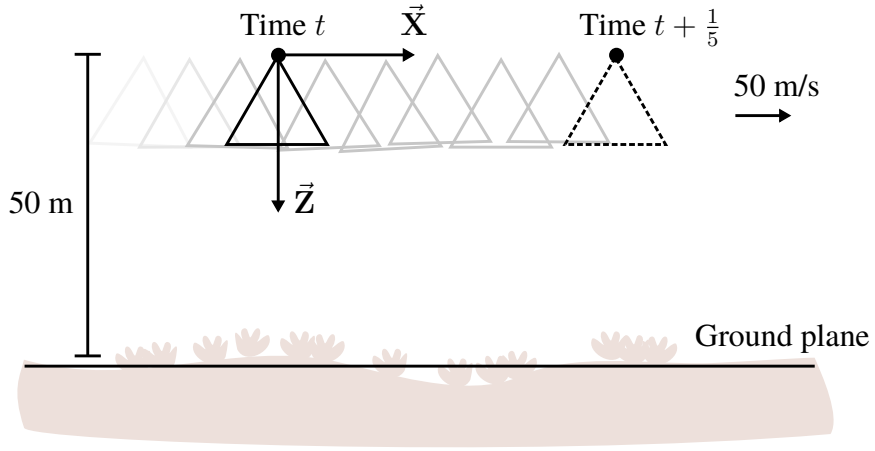


Figure 2: A camera moving over an approximately planar surface, as described in Task 1.1–1.2. Note that the camera is drawn as a triangle with the sensor in front of the camera, which makes no sense physically. However, its use as a visual symbol is somewhat standard in robotics and computer vision literature. In 3D it is often drawn as a pyramid.

## Part 1 Choosing a sensor and lens (20%)

You may at some point have to choose a sensor and lens for a particular application. Some camera vendors offer calculators that can help with this process. These are often based on the pinhole camera model, so it is possible to do this analysis yourself using the equations presented previously.

Some parameters of a sensor are its shutter speed, resolution and pixel size, which are specified in its datasheet. For a lens, the primary parameter is its focal length. The pixel size corresponds to $1/s_x$ and $1/s_y$. The focal length of a rectilinear lens is often a decent approximation of the principal distance $f$, at least when deciding which camera to purchase. (Szeliski §2.2.3 provides a brief motivation for when and why this approximation is reasonable.)

**Task 1.1:** (10%) An aerial vehicle will capture images of a crop, as illustrated in Fig. 2. The camera has a sensor with square pixels of size $10 \times 10$ microns, and is looking straight down from a height of 50 meters. Assume that the ground is flat, and estimate an appropriate focal length for a rectilinear lens, in order for one pixel in the image to cover a ground distance of one centimeter.

**Task 1.2:** (10%) Having overlap between images is important for image stitching (e.g. creating one large image of the entire crop). Suppose that the camera captures 5 images per second at a resolution of $1024 \times 1024$ pixels, and suppose that the camera is moving 50 meters per second along one of the lateral camera axes (e.g. $\vec{\mathbf{X}}$ as in Fig. 2). Determine the area of overlap between two consecutive images, using the focal length you found above. Give your answer as a percentage of the total image area.

Hint: Compute the distance (in pixels) by which any pixel in the image gets displaced from one image to the next, and divide the remainder by the image size (1024).

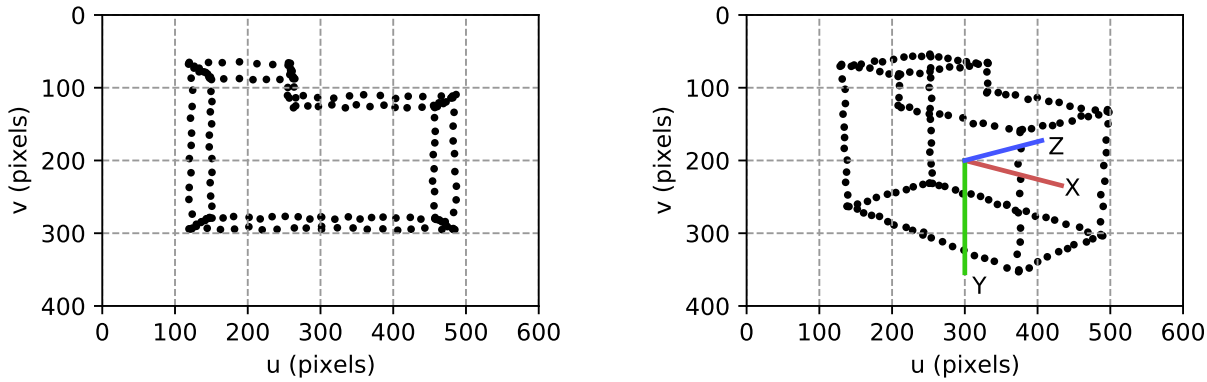


Figure 3: Projection of points in Task 2.2 (left) and transformed points in Task 3.2 (right).

## Part 2 The perspective camera model (10%)

Although the perspective camera model involves a non-linear operation (division by $Z$), it is often written as a linear relationship using homogeneous coordinates:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \underbrace{\begin{bmatrix} s_x f & s_\theta & c_x \\ 0 & s_y f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{8}$$

or simply

$$\tilde{\mathbf{u}} = \mathbf{K}\mathbf{X}, \tag{9}$$

where $\mathbf{K}$ is called the intrinsic matrix.

**Task 2.1:** (5%) The text above defines a variable $\tilde{\mathbf{u}}$ as a function of the variable $\mathbf{X}$. According to our notation, the text therefore implicitly defines a variable $\mathbf{u} = (\tilde{u}/\tilde{w}, \tilde{v}/\tilde{w})$ as a function of $\tilde{\mathbf{u}}$. However, it remains to be shown that $\mathbf{u}$ is related to $\mathbf{X}$ as in Eq. (5). Show that this relation holds when $Z \neq 0$. That is, prove the claim that a perspective projection can indeed be written as Eq. (9) using homogeneous coordinates.

**Task 2.2:** (5%) The hand-out code has a stub function called `project` in `project.m` (Matlab) and `common.py` (Python). The function should take an intrinsic matrix $\mathbf{K}$ and a $3\times N$ array of points $(X, Y, Z)$, and return a $2\times N$ array of pixel coordinates $(u, v)$, computed with the perspective model.

Implement the function and test it on the points and intrinsic matrix provided in `task2points.txt` and `task2K.txt`. Include a figure showing the projection of the 3D points. The `task2` script provides helper code to load the data and generate the required figure. Your figure should look similar to the left figure in Fig. 3 (minus the grid and labels).

## Part 3 Rigid transformations (20%)

**Task 3.1:** (5%) Suppose you have a homogeneous coordinate vector $\tilde{\mathbf{X}}$, representing some point in the camera frame. Since the vector does not necessarily have a last component equal to $1$, it must be dehomogenized to obtain the corresponding Cartesian coordinates.

Dehomogenization is necessary if you want to e.g. measure the distance between points, or recenter a set of points by subtracting their mean coordinates. However, show that dehomogenization is not necessary to compute a point's perspective projection. That is, show that the perspective projection of $(\tilde{X}, \tilde{Y}, \tilde{Z})$ is the same as that of $(\tilde{X}, \tilde{Y}, \tilde{Z})/\tilde{W}$.

This is very useful, since it allows us to compute the projection of points at infinity (which can be represented by finite homogeneous coordinates, with $\tilde{W} = 0$).

**Task 3.2:** (5%) Decide if the above is true of a homogeneous coordinate vector $\tilde{\mathbf{X}}^c$ given by

$$\tilde{\mathbf{X}}^c = \mathbf{T}\tilde{\mathbf{X}}^o,$$

where $\mathbf{T}$ is a rigid transformation, and $\tilde{\mathbf{X}}^o$ is a homogeneous coordinate vector in some frame $o$, with $\tilde{W}^o$ not necessarily equal to $1$.

**Task 3.3:** (10%) The points in Task 2 were in camera coordinates and had been pre-translated $6$ units along the camera $\vec{\mathbf{Z}}$-axis to be in front of the camera. The points in `task3points.txt` are instead given in the object frame indicated in Fig. 3 (right). Here, transformation $\mathbf{T}_o^c$ is composed of one translation by $6$ units and two elemental rotations by $15°$ and $45°$, respectively. Find an expression for $\mathbf{T}_o^c$. Check your answer by modifying `task2` to load the new points and apply your transformation before projection. The generated figure should look identical to Fig. 3 (right).

You don't need to expand the matrix product; you can give your answer in terms of the elemental rotation matrices $\mathbf{R}_x(\cdot), \mathbf{R}_y(\cdot), \mathbf{R}_z(\cdot)$, and a translation matrix $\mathbf{T}_z(\cdot)$. You may want to modify the `project` function to take a $4 \times N$ array of homogeneous coordinates. You can find definitions of the elemental rotation matrices on Wikipedia. Finally, you may use the provided function `draw_frame` to visualize the coordinate frame associated with $\mathbf{T}_o^c$.


Figure 4: Quanser 3-DOF helicopter and its three degrees of freedom. The arrows indicate the direction of increasing yaw, pitch and roll.

## Part 4 Image formation model for the Quanser helicopter (50%)

The Quanser 3-DOF helicopter consists of an arm with a counterweight at the back and a rotor carriage with two rotors at the front. It has three rotational degrees of freedom (Fig. 4):

- Yaw ($\psi$): Rotation around an axis perpendicular to the mounting platform
- Pitch ($\theta$): Rotation of the arm up or down
- Roll ($\phi$): Rotation of the rotor carriage around the arm

In the next homework, you'll estimate these angles from markers attached to the helicopter. To do this, you need a mathematical model that relates points on the helicopter to pixel coordinates in the image. A reasonable solution is to model the helicopter as a piecewise rigid body and attach a coordinate frame to each part of interest. Your main task here is to define the $4 \times 4$ transformation matrices between the coordinate frames shown in Fig. 5, using the measurements indicated in Fig. 6.

**Task 4.1:** (5%) A square platform with four screw holes is shown in Fig. 7. The distance between adjacent screws is 11.45 cm. Define four coordinate vectors corresponding to the screw locations. The coordinate vectors should be expressed in the coordinate frame shown in Fig. 7a, which is aligned with the sides of the platform and has its origin on the closest screw.

**Task 4.2:** (10%) Verify that your answers above are correct by writing a script to reproduce Fig. 7a:

1. Load and draw the image `quanser.jpg`. Adjust the figure limits to zoom in on the platform.
2. Use the transformation $\mathbf{T}_{\text{platform}}^{\text{camera}}$ in `platform_to_camera.txt` to transform your coordinate vectors into camera coordinates. Assume that the camera satisfies a perspective model with $\mathbf{K}$ in `heli_K.txt`, and project the transformed coordinate vectors into the image.

Include the figure in your writeup. The drawn points should coincide with the screws. You don't have to reproduce Fig. 7 exactly, e.g. the labels and stippled lines can be omitted.

As in Task 3.2, for the following tasks you don't need to multiply and write out the entries of the $4\times4$ matrix. You can express your answer as a product of the elemental rotation matrices $\mathbf{R}_x(\cdot), \mathbf{R}_y(\cdot), \mathbf{R}_z(\cdot)$ and translation matrices $\mathbf{T}_x(\cdot), \mathbf{T}_y(\cdot), \mathbf{T}_z(\cdot)$.

**Task 4.3:** (10%) 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 $\psi$ around z. Find an expression for the transformation $\mathbf{T}_{\text{base}}^{\text{platform}}(\psi)$ that transforms coordinate vectors expressed in the base frame to coordinate vectors expressed in the platform frame. (For brevity we will say that this transforms base coordinates to platform coordinates.)

**Task 4.4:** (5%) Pitch motion is enabled by a hinge that rotates with 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 $\theta$ around the base frame y-axis. Find an expression for $\mathbf{T}_{\text{hinge}}^{\text{base}}(\theta)$.

**Task 4.5:** (5%) The arm is rigidly attached to the hinge. As indicated in Fig. 5, the "arm" frame is centered in the cross-section of the arm with its x-axis pointing down the arm toward the rotors. It is obtained by translating $-5$ cm along the hinge frame z-axis. Find an expression for $\mathbf{T}_{\text{arm}}^{\text{hinge}}$.

**Task 4.6:** (5%) 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 $65$ cm along the x-axis, $-3$ cm along the z-axis, and rotating counter-clockwise by $\phi$ around the x-axis. Find an expression for $\mathbf{T}_{\text{rotors}}^{\text{arm}}(\phi)$.

**Task 4.7:** (10%) Verify your answers by writing a script to reproduce Fig. 8:

1. Load and draw the image `quanser.jpg`.
2. Instantiate the transformation matrices using $\psi = 11.6°$, $\theta = 28.9°$ and $\phi = 0°$. Use the `draw_frame` function to draw the coordinate frames. (Set the scale argument to about 0.05.)
3. Load the marker coordinate vectors in `heli_points.txt`. The first three are given in the arm frame, and the last four are given in the rotors frame. Draw the points by applying the correct sequence of transformations, followed by projection.

Each projected point is meant to lie exactly on one of the corners of its associated marker. However, if you have done everything correctly, the projected points will be off by up to 10 pixels. This is due to measurement and modeling inaccuracies.

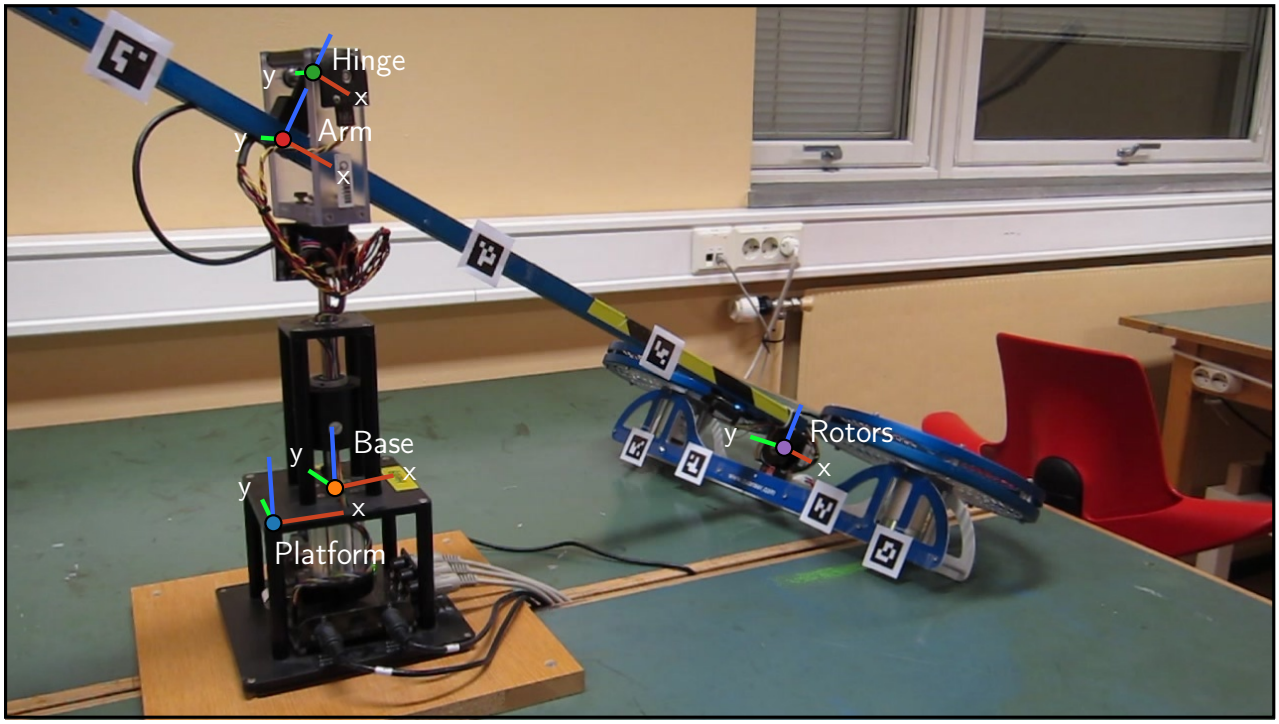


Figure 5: Helicopter coordinate frames

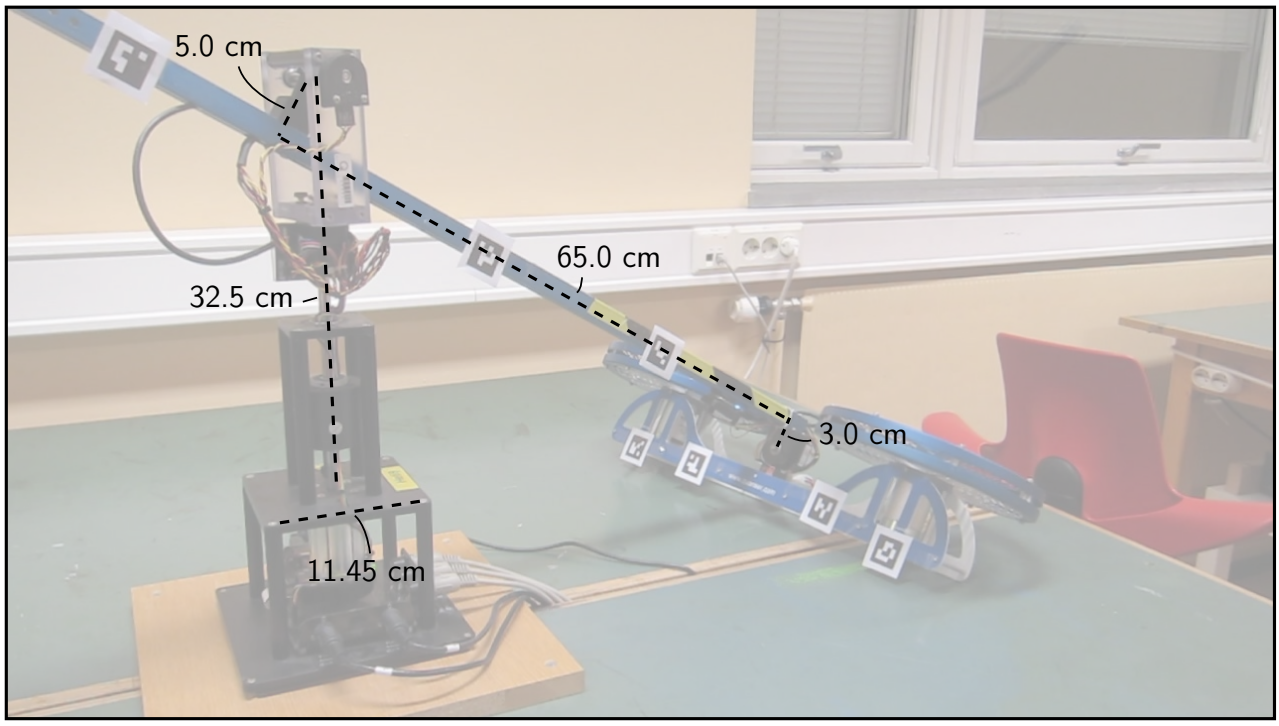


Figure 6: Helicopter dimensions

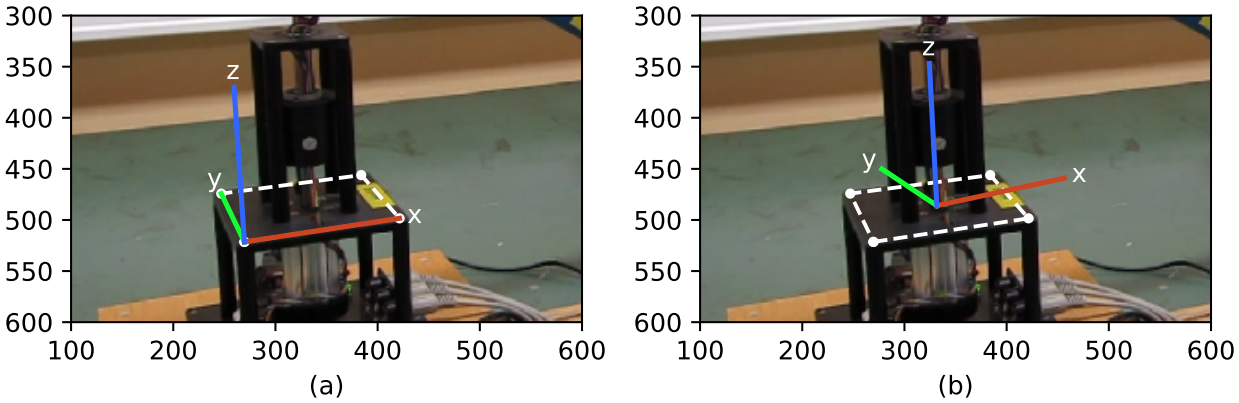


Figure 7: (a) Platform coordinate frame and points located on the four screws. (b) Base coordinate frame, located in the platform center with the z-axis perpendicular to the platform.

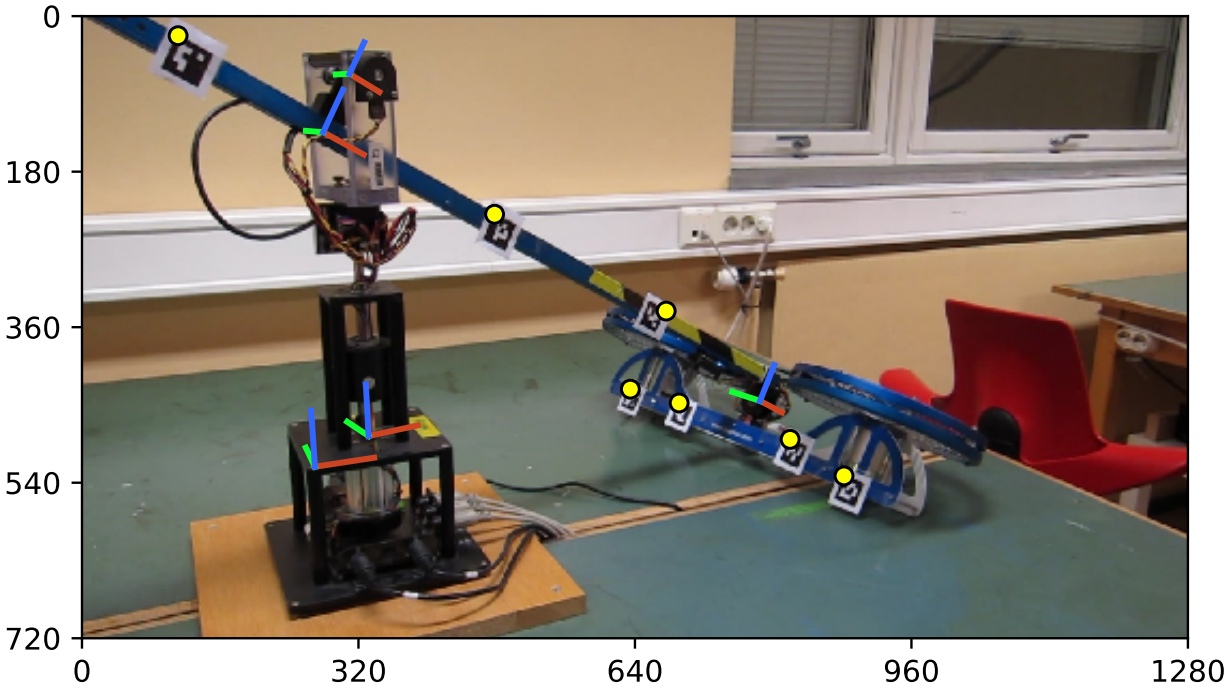


Figure 8: Helicopter frames and reprojected fiducial marker points