TTK4255: Robotic Vision

# Homework 3: Geometric image formation

## Instructions

For more information about the course work and how to get help, see the `Course work/about.pdf` document on BB. To get your assignment approved, you need to complete 60% (sum of task weights). Upload the requested answers and figures as a single PDF. You don't need to submit your code. You may collaborate with other students and submit the same report, but you still need to upload it individually on Blackboard. Please write your collaborators' names on your report.

## Relevant reading

For this assignment, you need an understanding of rigid body motion (3D rotation and translation). Szeliski 2.1.1-2.1.4 gives a summary of various geometric primitives and transformations, but most of you should have received an introduction to rigid body motion in an earlier course (TTK4135 or TTK4190), and can just skim over these sections. If you are currently taking either course, you may instead want to read ahead in the book used in those courses.

An introduction to the pinhole camera model can be found in countless textbooks and articles, with the primary differences being their chosen mathematical notation and writing style. While you are welcome to read Szeliski 2.1.5 for the course book's introduction, this assignment includes a summary of the most important parts, written with the notation that we will use throughout the assignments. Unlike Szeliski, we will distinguish between coordinates in the 3D world and coordinates in the image by upper case (e.g. $X, Y, Z$) and lower case (e.g. $u, v$), respectively.

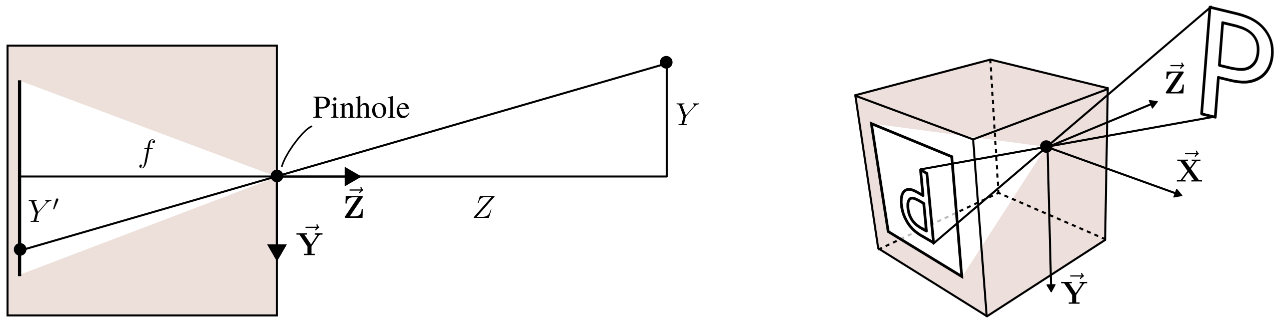

Figure 1: Diagram of an ideal pinhole camera seen from the side and in 3D.

## The pinhole camera model

A common model of geometric image formation is the *perspective projection*. The simplest imaging device that can be described by this model is the *pinhole camera*: a box with a small opening (the pinhole) at one end, through which light enters and reaches an imaging surface at the opposite end. The *ideal pinhole camera* is a theoretical device in which the opening is made infinitely small. This has the consequence that every point on the imaging surface receives light from a unique direction, all of which intersect at the pinhole. The resulting mapping from scene to image is a perspective projection.

Letting $f$ denote the distance between the pinhole and the imaging surface in an ideal pinhole camera, the relationship between a point $(X, Y, Z)$ in the scene and its projected location $(X', Y', Z')$ on the imaging surface can be derived by a consideration of similar triangles (see Fig. 1):

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

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

and finally $Z' = -f$. These equations will differ based on how you place the camera's axes. In this course we will follow the convention in which positive $Z$ is in front of the camera, 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 will appear to be inverted (i.e. rotated 180 degrees), compared with what you would see through the pinhole. This inversion also occurs in a modern digital camera, but the camera firmware and your operating system work together to ensure that the rotation is undone when the image is presented on your screen.

A modern digital camera similarly consists of a box with an opening at one end, which allows light to enter and hit an imaging surface (usually a CCD or CMOS sensor). The difference is that the opening is not infinitely small. Instead, it is made large to collect more light, and a lens is placed in front of the opening to focus the light. What is important to note is that the geometric transformation of points in the scene to points on the sensor can be—or at least, is often—modeled by the same equations (perspective projection). This is somewhat surprising when it is considered that modern cameras have complex optics which bend light at multiple stages. The fact that the model continues to be applicable is in part because camera and lens engineers strive to produce a perspective projection, as the resulting images appear most natural and pleasing to a human observer.

In a high-quality camera, the agreement can be good enough to not require further modeling. Otherwise, smaller errors can often be modeled during camera calibration, and corrected to produce an equivalent perspective projection image. Because of this, the perspective projection model is applicable for the majority of commercially available cameras. However, it is worth mentioning that some cameras, such as fisheye lens cameras, do not have as a goal to produce natural-looking images, and may deviate substantially from a perspective projection. In such cases, a different camera model may be warranted.

When working with a digital camera, we also need to model the transformation from locations on the physical sensor surface (in metric units) to the pixel coordinates (rows and columns) used to access the digital image. With few exceptions, the pixel coordinate system is normally placed in the upper left corner of the noninverted image, with the horizontal axis pointing to the right in the scene and the vertical axis pointing down in the scene. Letting $(u, v)$ denote the horizontal and vertical pixel coordinates, these are then related to $(X', Y')$ by a negation, scale and offset:

$$u = c_x - s_x X' \quad \text{and} \quad v = c_y - s_y Y', \tag{3}$$

which can be simplified and written in terms of the original $(X, Y, Z)$ 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 above is mathematically equivalent to placing the imaging surface in front of the camera (i.e. at $Z = f$). However, this makes no sense physically, and misleadingly suggests that points with $Z \in [0, f]$ are behind the imaging surface and therefore not visible, which is false.

The parameters $c_x, c_y, s_x, s_y, f$ are given the following names and have the following geometric interpretation in the ideal pinhole camera model:

- $c_x, c_y$: *Principal point* - the intersection point between the optical axis and the imaging surface (in pixels). The optical axis is the line that is orthogonal to the imaging surface and passes through the camera origin. In our chosen convention, this is the $\vec{\mathbf{Z}}$-axis.
- $s_x, s_y$: *Pixel density* - the number of discrete sensing elements horizontally and vertically per unit length of the imaging surface (often in pixels per micrometer).
- $f$: *Focal length* - the distance between the pinhole and the imaging surface (often in millimeters).

The principal point should not be confused with the *optical center* or *center of projection*, which is the 3D origin of the camera coordinate system, where the incoming light rays intersect. This distinction is made in the majority of computer vision literature, but Szeliski uses the term optical center for both quantities. In the ideal pinhole camera, the center of projection coincides with the pinhole. The center of projection in a real camera is less obvious, and may be behind, within or in front of the lens system.

You will more commonly see $s_x$ and $s_y$ as their inverse quantities, which represent the physical distance between horizontally and vertically adjacent pixels, 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.

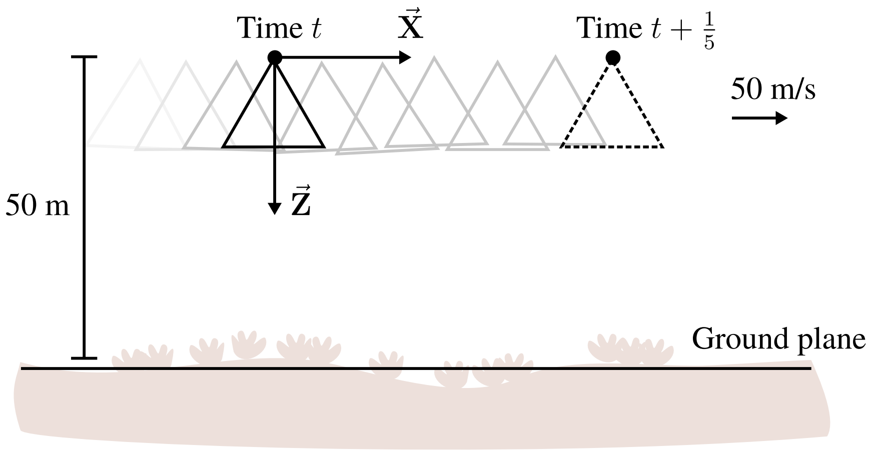

Figure 2: A camera moving over an approximately planar surface, as described in Task 1.1-1.2. Note that the camera is drawn as a triangle with the imaging surface *in front* of the camera, which, as mentioned before, makes no sense. However, its use as a visual symbol is somewhat standard in robotics 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 camera sensor and lens combination to satisfy the imaging requirements of a particular application. Many 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 key parameters for the sensor are its shutter speed, resolution and pixel size, which are specified in its datasheet. For the lens, the primary parameter is its focal length. The pixel size corresponds to $1/s_x$ and $1/s_y$ in the pinhole camera model. The focal length of a lens can to a first approximation be considered as equivalent to the identically named parameter $f$ in the pinhole camera model. (Szeliski 2.2.3 provides a brief motivation for when and why this approximation is reasonable.)

**Task 1.1:** (10%) Imagine you are designing an aerial vehicle to capture images of an agricultural crop as in Fig. 2. Suppose that you have a camera sensor with square pixels of size $10\times 10$ microns and suppose that the camera is looking straight down from a height of 50 meters. Assume that the ground is flat and determine what focal length you need in order for one pixel in the image to cover a ground distance of one centimeter. Give your answer in millimeters.

**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).

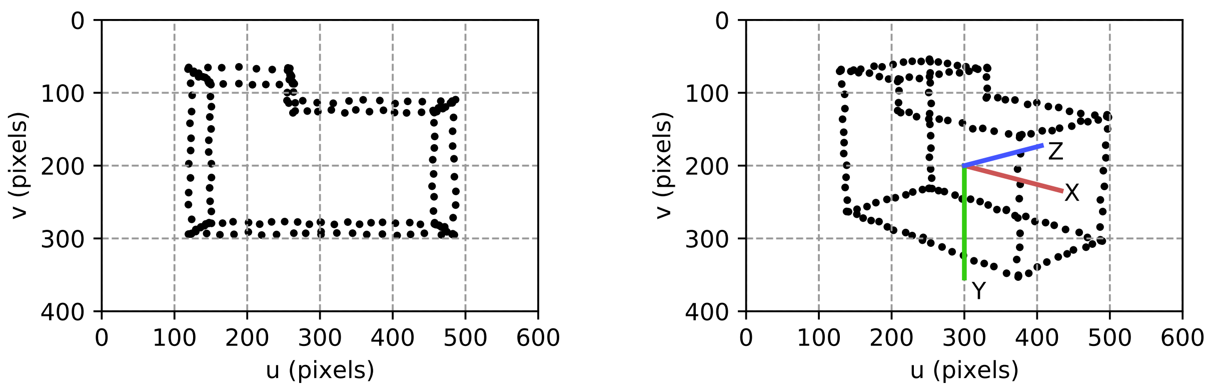

Figure 3: Projection of points in Task 2.2 (left) and transformed points in Task 3.2 (right).

## Part 2 Implementing the pinhole camera model (10%)

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

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} = \underbrace{\begin{bmatrix} s_x f & 0 & c_x \\ 0 & s_y f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5}$$

or simply

$$\tilde{\mathbf{u}} = \mathbf{KX} \tag{6}$$

where $\mathbf{K}$ is called the *calibration*- or *intrinsic matrix*. The vector $\tilde{\mathbf{u}} = (\tilde{u}, \tilde{v}, \tilde{w})$ is the *homogeneous* form of the pixel coordinate vector $\mathbf{u} = (u, v)$. These are implicitly related by

$$\tilde{w}u = \tilde{u} \quad \text{and} \quad \tilde{w}v = \tilde{v}. \tag{7}$$

Like Szeliski, we denote homogeneous coordinates using the tilde "$\sim$" symbol. The motivation for writing the equations in this form is probably only truly appreciated after taking a course in projective geometry. Nevertheless, you will frequently encounter the linear formulation in practice.

**Task 2.1:** (5%) Dividing a homogeneous vector by its last component is called dehomogenization. Show that the dehomogenized coordinates $\tilde{u}/\tilde{w}$ and $\tilde{v}/\tilde{w}$ are equal to $u$ and $v$ in Eq. 4, when $Z \neq 0$.

**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 a calibration 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 pinhole model.

Implement the function and test it on the points and calibration matrix provided in `task2points.txt` and `task2K.txt`. Include a figure showing the pinhole 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 Homogeneous coordinates and transformations (10%)

So far, $(X, Y, Z)$ has referred to a point's coordinates in the camera coordinate system. However, the point may be expressed in a different coordinate system, perhaps attached to the environment or to an object. If so, we need to include an additional transformation that relates the two coordinate systems. This transformation is called a rigid body motion or Euclidean transformation, and is described by a 3D rotation and 3D translation. Given a point $\mathbf{X}^o$ expressed in a coordinate system (or *frame*) "$o$", its coordinate vector in the camera frame "$c$" is

$$\mathbf{X}^c = \mathbf{R}\mathbf{X}^o + \mathbf{t}, \tag{8}$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is a translation vector. Note that we distinguish between coordinates in different frames with a superscript. Sometimes $\mathbf{R}$ and $\mathbf{t}$ are grouped into a single 4×4 matrix, such that the above can be written as

$$\begin{bmatrix} \mathbf{X}^c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}^o \\ 1 \end{bmatrix} \tag{9}$$

with a scalar 1 appended to both vectors. More concisely we may write

$$\tilde{\mathbf{X}}^c = \mathbf{T}_o^c \tilde{\mathbf{X}}^o \tag{10}$$

where $\mathbf{T}_o^c$ is read as "the transformation from $o$ to $c$". Using 4×4 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 $\mathbf{T}_a^c = \mathbf{T}_b^c \mathbf{T}_a^b$.

**Task 3.1:** (5%) The vectors in Eq. 10 are generally homogeneous with the last component not necessarily equal to 1. They must therefore be divided by the last component to obtain the actual 3D coordinates. However, show that this step can be skipped when computing the projection $(u, v)$ of a homogeneous vector $\tilde{\mathbf{X}} = (\tilde{X}, \tilde{Y}, \tilde{Z}, \tilde{W})$, i.e. show that the last component ($\tilde{W}$) can simply be dropped.

**Task 3.2:** (5%) 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 coordinate frame indicated in Fig. 3 (right). Here, the object-to-camera transformation $\mathbf{T}_o^c$ is a product 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). Note: 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 $\mathbf{T}_z(\cdot)$.

Tip: 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 $\mathbf{T}_o^c$, as in the figure.

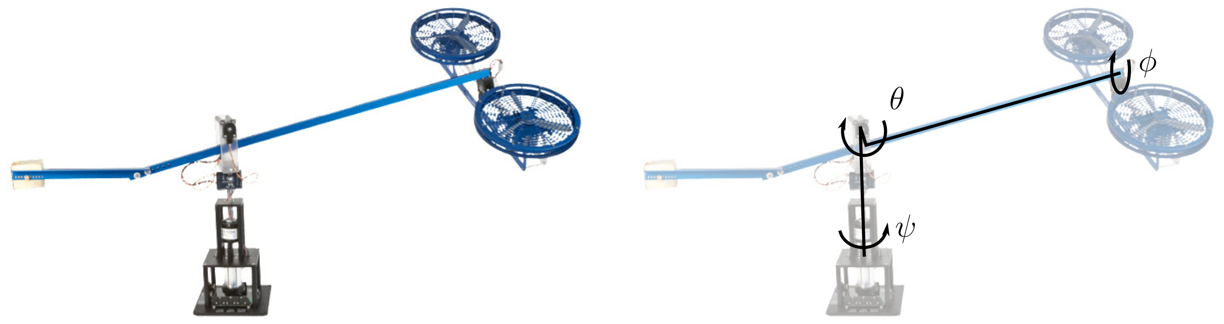

Figure 4: Quanser 3-DOF helicopter and its three degrees of freedom. The arrows indicate the direction of increasing yaw, pitch and roll.

## Part 4 Image formation model for the Quanser helicopter (60%)

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 midterm project, 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. The coordinate frames you should use are shown in Fig. 5. Your main task here is to define the $4 \times 4$ transformation matrices between the different coordinate frames, 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:** (5%) 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 limits of the figure to zoom in on the platform.
2. Use the transformation $\mathbf{T}^{\text{camera}}_{\text{platform}}$ in `platform_to_camera.txt` to transform your coordinate vectors into camera coordinates. Assume that the camera satisfies a pinhole model with the matrix $\mathbf{K}$ 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 $\mathbf{T}^{\text{platform}}_{\text{base}}(\psi)$ from the base frame to the platform frame.

**Task 4.4:** (10%) 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 $\mathbf{T}^{\text{base}}_{\text{hinge}}(\theta)$.

**Task 4.5:** (10%) 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 $\mathbf{T}^{\text{hinge}}_{\text{arm}}$.

**Task 4.6:** (10%) 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 $\mathbf{T}^{\text{arm}}_{\text{rotors}}(\phi)$.

**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 $\psi = 11.6^\circ$, $\theta = 28.9^\circ$ and $\phi = 0^\circ$. 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. (In the midterm project you will learn how to calibrate the model itself by tracking the helicopter through a long motion sequence.)

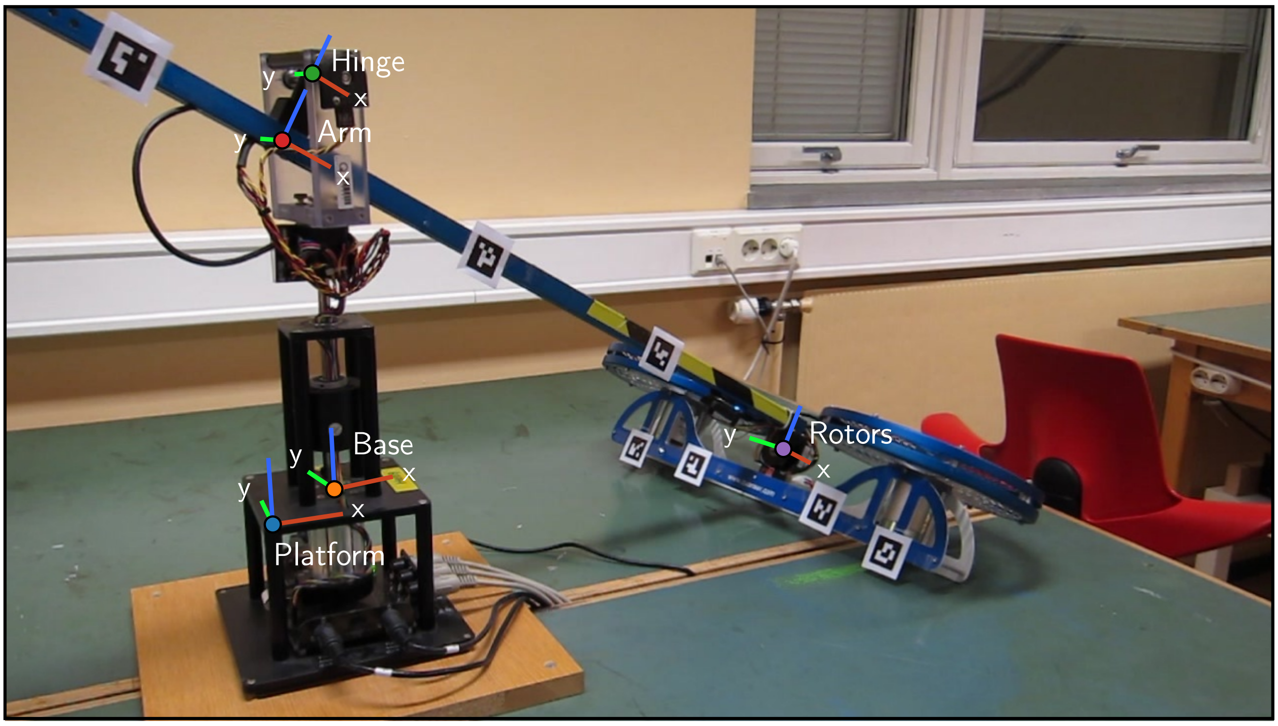

Figure 5: Helicopter coordinate frames

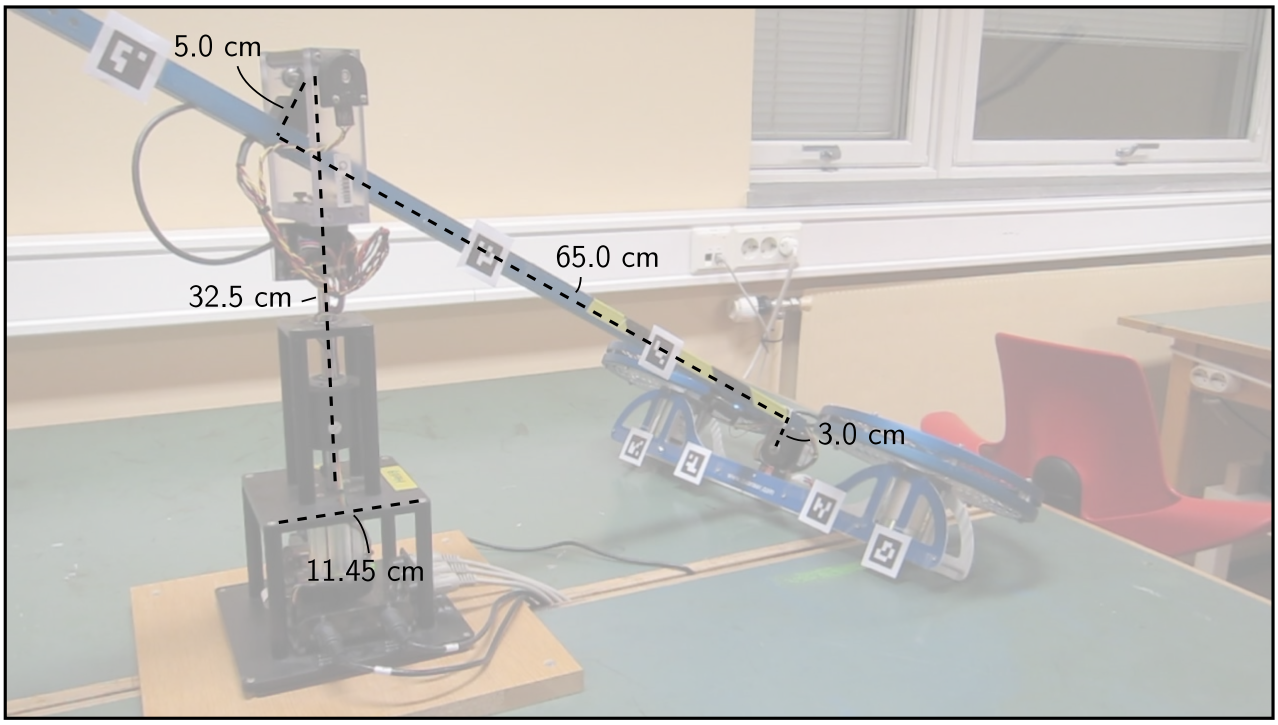

Figure 6: Helicopter dimensions

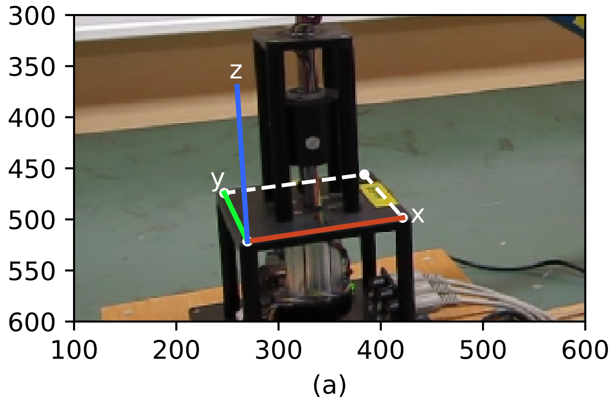

Figure 7: (a) Platform coordinate frame and points located on the four screws. (b) Base coordinate frame, located in the platform center with the z-axis perpendicular to the platform.

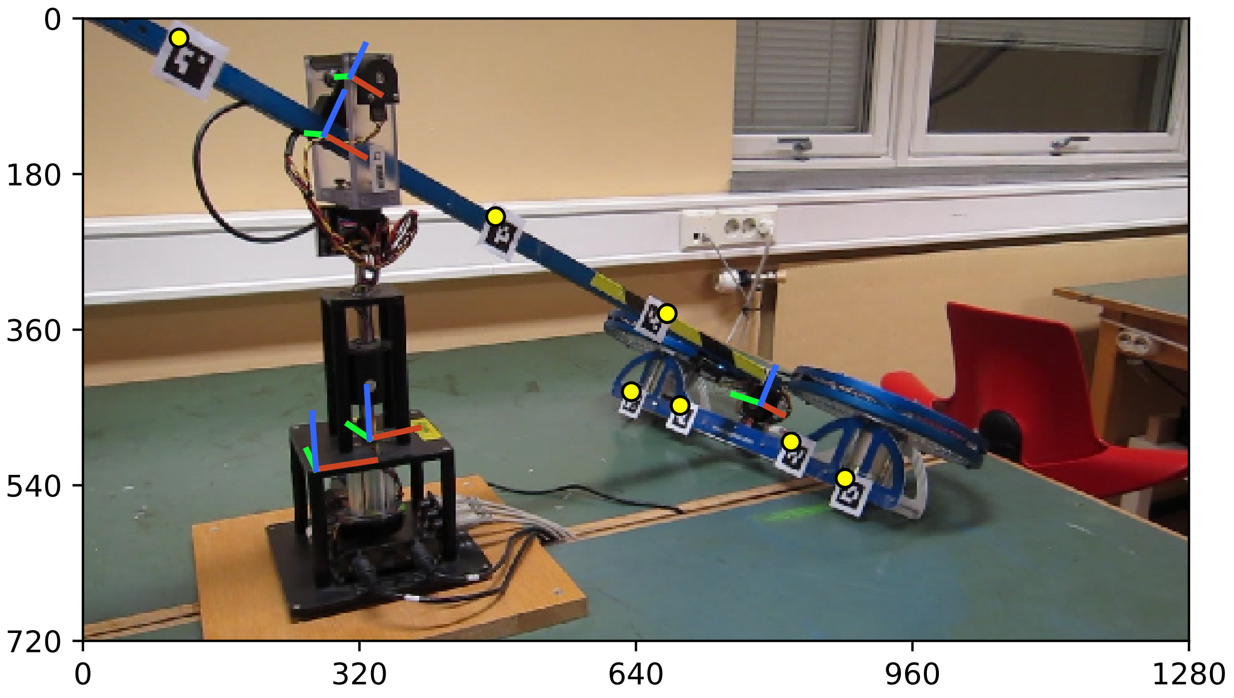

Figure 8: Helicopter frames and reprojected fiducial marker points