Robotics
Assignment 3
Group 1_Fri_G

Nicolas Marco Hahn (501703)

Foued Larbi (412927)

Daniel Papp (401765)

December 17, 2023

1 A Control Theory

a) Given $k = 4$, $m = 16$ and $b = 16$, the natural frequency ω_n and the natural damping ratio ζ_n can be calculated as follows:

$$\zeta_n = \frac{b}{2\sqrt{km}} = 1, \qquad \omega_n = \sqrt{\frac{k}{m}} = 0.5$$

In fact, the system is critically damped since the natural damping ratio $\zeta_n = 1$.

b) We can use the following equation from the tutorial. It is assumed that the closed loop stiffness is $k_{CLS} = 16$ and the desired position is $x_d = 0$:

$$m\ddot{x} = -(b + k_v)\dot{x} - (k + k_p)x$$

$$b' = (b + k_v)$$
$$k' = (k + k_p) = k_{CLS} = 16$$

$$\Rightarrow k_p = k' - k = 16 - 4 = 12$$

It is assumed that the system is critically damped :

$$b' = 2\sqrt{mk'} = 2\sqrt{m(k + k_p)} = b + k_v$$

$$\Rightarrow k_v = b' - k = 16$$

Hence, we can design the PD controller f as follows:

$$\Rightarrow f = -k_p x - k_v \dot{x} = -12x - 16\dot{x}$$

c) The error is specified as $e = x - x_d$ and it is assumed that $\alpha = m = 16$.

Assuming that $k_{CLS} = m \cdot k'_p = 16$ the coefficients k'_p and k'_v are given by $k'_p = 1$ and $k'_v = 2\sqrt{k'_p} = 2$.

Given the trajectory-following control equation $\ddot{e} + k'_v\dot{e} + k'_p e = 0$

We can use the following equation from the tutorial:

$$m\ddot{x} = \alpha(\ddot{x}_d - k'_v(\dot{x} - \dot{x}_d) - k'_p(x - x_d)) + \alpha\beta - (b\dot{x} + kx)$$

In order to design an m-mass control, it is assumed that $\alpha = m$ and $\beta = 30sign(\dot{x}) + kx$. The derivation of the error is given by $\dot{e} = \dot{x}_d - \dot{x}$.

We can use the following equation to design the controller f:

$$f' = \ddot{x}_d + k'_v\dot{e} + k'_p e = \ddot{x}_d + 2\dot{e} + e$$

$$f = \alpha f' + \beta$$

$$\Rightarrow f = 16\ddot{x}_d + 30 \cdot sign(\dot{x}) + 4x + 16(2\dot{e} + e)$$

d) In order to compute the steady-state error $e = x - x_d$ after being disturbed by a constant force $f_{dist} = 8$, we can use the following equation:

$$e = \frac{f_{dist}}{k'_p} = \frac{8}{1} = 8$$

2 B Visual Servoing

2.1 Image Features

In this section the objects $triangle$, $sphere$ and $square$ are analysed regarding their visual servoing compatibility. This includes determining the minimal parameterization in s vector, extraction difficulty and number of Degrees of Freedom (DoF) that can be controlled. Consequently, we can deduce whether the System of Equations $\dot{s} = J_I\dot{p}_c$ is under/-overconstrained or well-formed. The folllowing tables 1 - 3 outline respectively the aforementioned analysis:

Criteria	**Triangle**
(Minimal) Parameterization	Coordinates of the three vertices in the image space.
Dimensionality of Feature	6
Extraction Difficulty	complex and time-costly due to high number of parameters/ vertices of a triangle can be detected with standard image processing techniques.
DoF Controlled	6 ; given the non-symmetry of the triangle.
Image Jacobian Dimensionality	6x6, with the 2D movements of the vertices related to the 3D space positioning and orientation.
System of Equations	Well-formed if the triangle is non-equilateral or non-isosceles, otherwise orientation about the axis normal to the triangle may be underconstrained.

Table 1: Analysis of a triangle for visual servoing

Criteria	**Sphere**
(Minimal) Parameterization	Center of the sphere in the image (represented by its 2D coordinates) and radius.
Dimensionality of Feature	3
Extraction Difficulty	Easy; the spherical shape is distinct and can be identified using standard algorithms such as Hough transform or edge detection.
DoF Controlled	3 (x, y, z); orientation is not defined for a perfect sphere.
Image Jacobian Dimensionality	3x6 since the operational space is 6D
System of Equations	Well-formed for translation but not applicable for orientation, thus underconstrained

Table 2: Analysis of a sphere for visual servoing

Criteria	**Square**
(Minimal) Parameterization	2D Coordinates of one corner point, the angle α and the side length
Dimensionality of Feature	4
Extraction Difficulty	Easy; corners of a square are distinctive and can be detected using corner and edge detection algorithms if contrast between image and environment is sufficient.
DoF Controlled	4 (x, y, z, yaw)
Image Jacobian Dimensionality	4x6, mapping the 4 parameters' movements to 3D position and orientation changes.
System of Equations	underconstrained

Table 3: Analysis of a square for visual servoing

2.2 Find Circle

a) The OpenCV library already provides the Hough-Transformation function, which paves the way for the the circle feature's extraction, with respect to the following steps:

- The provided input image for the `findCircleFeature` is already in gray scale.
- A Gaussian blur `cv::GaussianBlur` is applied to filter out the noise and its caused error. Thus, our implementation is tolerant to adverse camera images.
- The OpenCV function `cv::HoughCircles` ensures optimally the feature extraction with the help of fine-tuned parameters like the accumulator threshold, min and max radius, and the resolution ratio based on the specific requirements of our image.

b) Since the transformation of the features from the OpenCV image frame into the feature frame relies solely on a translation, the following equations are implemented in `transformFromOpenCVFToFF`:

$$x = x_{cv} - \frac{\texttt{img\_width}}{2}, \quad y = y_{cv} - \frac{\texttt{img\_height}}{2}, \quad z = z_{cv}$$

2.3 Image Jacobian Computation

a) In order to compute the depth z of the circle feature, we use the formula and the parameters depicted in Fig. 1, which has been presented in the tutorial:

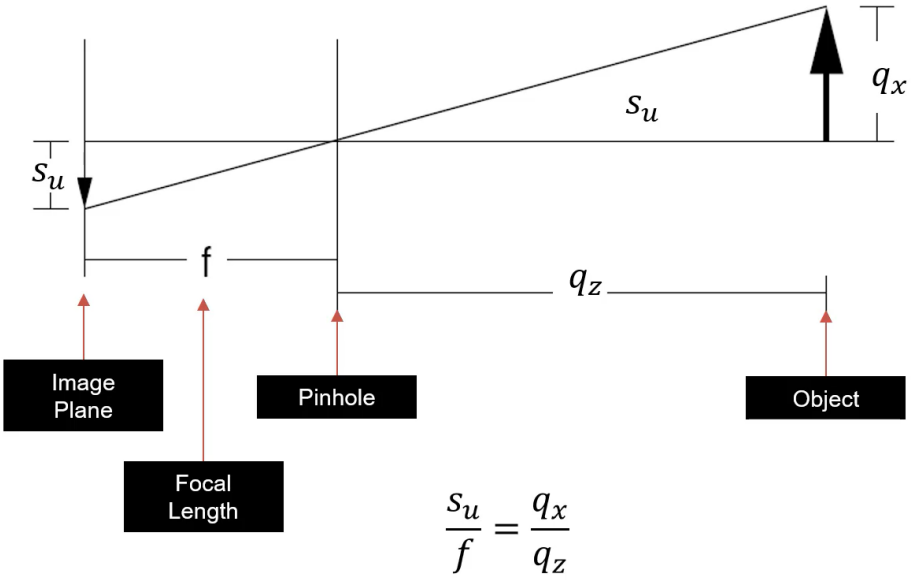

Figure 1: Perspective projection in 2D

The Hough-Transformation function returns the circle radius. Its value will be stored in the struct member `crcl.radius`.

- f: the focal length
- $q_x \sim \frac{diameter}{2}$: the object's real height
- $q_z \sim z$: the distance between the camera pinhole and the object
- $s_u \sim$ `crcl.radius`: the printed object's height on (virtual) image frame

Consequently, the depth z is given by:

$$\frac{\texttt{crcl.radius}}{f} = \frac{diameter}{2z}$$
$$\Rightarrow z = \frac{f \cdot diameter}{2 \cdot \texttt{crcl.radius}}$$

b) The focal length can be determined experimentally by calculating the depth difference Δz of two printed features from different distances regarding the camera. Both of the depths z_1 and z_2 are given by:

$$z_1 = \frac{diameter}{2} \cdot \frac{f}{\texttt{crcl.radius1}} \qquad z_2 = \frac{diameter}{2} \cdot \frac{f}{\texttt{crcl.radius2}}$$

$$\Delta z = z_2 - z_1$$

$$\Delta z = \frac{diameter}{2} \cdot f \cdot \frac{(\texttt{crcl.radius2} - \texttt{crcl.radius1})}{\texttt{crcl.radius1} \cdot \texttt{crcl.radius2}}$$

$$\Rightarrow f = 2\Delta z \cdot \frac{\texttt{crcl.radius1} \cdot \texttt{crcl.radius2}}{diameter \cdot (\texttt{crcl.radius2} - \texttt{crcl.radius1})}$$

c) The s vector for describing a point in the feature frame is given by:

$$s = \begin{bmatrix} s_u \\ \\ s_v \end{bmatrix} = f \cdot \begin{bmatrix} \frac{q_x}{q_z} \\ \\ \frac{q_y}{q_z} \end{bmatrix}$$

Considering the s_d parameter for describing the circle diameter, which is computed analogously to s_u in **a)** : $\frac{s_d}{f} = \frac{diameter}{q_z} \Leftrightarrow s_d = f \cdot \frac{diameter}{q_z}$

$$\Rightarrow s = \begin{bmatrix} s_u \\ \\ s_v \\ \\ s_d \end{bmatrix} = f \cdot \begin{bmatrix} \frac{q_x}{q_z} \\ \\ \frac{q_y}{q_z} \\ \\ \frac{diameter}{q_z} \end{bmatrix}$$

The derivation of the s vector over time $\frac{ds}{dt}$ is given by:

$$\dot{s} = \begin{bmatrix} \dot{s_u} \\ \\ \dot{s_v} \\ \\ \dot{s_d} \end{bmatrix} = \begin{bmatrix} -\frac{f}{q_z} \cdot v_x + \frac{s_u}{q_z} \cdot v_z \\ \\ -\frac{f}{q_z} \cdot v_y + \frac{s_v}{q_z} \cdot v_z \\ \\ \frac{f \cdot diameter}{q_z^2} \cdot v_z \end{bmatrix} = J_I \cdot \begin{bmatrix} v_x \\ \\ v_y \\ \\ v_z \end{bmatrix}$$

Hence, we can specify the Image Jacobian matrix $J_I \in \mathbb{R}^{3\times3}$ as follows and implement it in the `getImageJacobianCFToFF` method:

$$\Rightarrow J_I = \begin{bmatrix} -\frac{f}{q_z} & 0 & \frac{s_u}{q_z} \\ \\ 0 & -\frac{f}{q_z} & \frac{s_v}{q_z} \\ \\ 0 & 0 & \frac{f \cdot diameter}{q_z^2} \end{bmatrix}$$

2.4 Velocity Vector Transformation

As illustrated in the figure 2 in the task guidelines, the camera frame is rotated around the z axis by $\theta = 90°$ from the end-effector frame. In fact, this static transformation can be expressed as a rotation matrix and implemented in `transformVelocityFromCFToEEF`:

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The result of the transformation can be expressed as follows:

$$\begin{bmatrix} x_{eef} \\ y_{eef} \\ z_{eef} \end{bmatrix} = \begin{bmatrix} -y_{cf} \\ x_{cf} \\ z_{cf} \end{bmatrix}$$

Now we can transfrom the velocity from End Effector Frame (EEF) to Base Frame (BF) since the defined pose of the EE in the base frame is provided with the help of the 7 dimensional `x_current_bf` vector. By way of explanation, the parameters q_0, q_1, q_2 and q_3 are set to the vector's last four elements defining the rotation. Hence, the rotation transform of the Quaternions can be computed as follows [1]:

$$R(q) = \begin{bmatrix} 2q_0^2 + 2q_1^2 - 1 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 + 2q_2^2 - 1 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 + 2q_3^2 - 1 \end{bmatrix}$$

2.5 Workspace Limitation

In order to avoid singular configurations we assume a spherical workspace that limits the robot movements. It is defined by the center (the base frame origin) and the radius $0.85m$. It is particularly convenient to take advantage from the first three elements of the `x_current_bf`

vector, which describe translations on x, y and z axes between BF and EEF. With respect to the constraint above, the aforementioned vector components will form the equation for the spherical reachable workspace:

$$x^2 + y^2 + z^2 \leq (0.85m)^2$$

$$\Rightarrow \sqrt{x^2 + y^2 + z^2} \leq 0.85m$$

3 C Contribution

Table 4: Contribution

Student Name	B2	B3	B4	B5
Nicolas Marco Hahn	x		x	
Foued Larbi		x	x	x
Daniel Papp	x		x	

References

[1] J. B. Kuipers, 2002. Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality (Chapter 5, Section 5.14 "Quaternions to Matrices"). ISBN-10 : 9780691102986.