Robotics

Assignment 03
Report

1_TUE_I
Vasilije Rakcevic
Jingsheng Lyu
Prathamesh More

January 10, 2021

Contents

Chapter 1

Control Theory

For a 1 DOF spring mass damper system, following are the given quantities.
mass (m) = 16
friction coeff (b) = 16
spring constant (k) = 4
hence the equation of the motion is

$$16x'' + 16x' + 4x = f \tag{1.1}$$

1.1 Calculation of natural frequency and damping ratio

Natural frequency ω_n

$$\omega_n = \sqrt{\frac{k}{m}} = \sqrt{\frac{4}{16}} = \frac{1}{2} \tag{1.2}$$

Natural Damping ratio ζ_n

$$\zeta_n = \frac{b}{2\sqrt{k*m}} = \frac{16}{2\sqrt{4*16}} = 1 \tag{1.3}$$

The characteristics equation can also be written as..

$$16s^2 + 16s + 4 = 0 \tag{1.4}$$
$$4s^2 + 4s + 1 = 0 \tag{1.5}$$

The roots of the equations are

$$s1 = \frac{-16 + \sqrt{16^2 - 4 * 16 * 4}}{2} = -8 \tag{1.6}$$

$$s2 = \frac{-16 - \sqrt{16^2 - 4 * 16 * 4}}{2} = -8 \tag{1.7}$$

Therefore as the root of the equations are equal, Therefore the system is critically damped

1.2 Designing PD controller

The closed loop stiffness K_{KLS} is given as 16
Now the equation of the closed loop system is,

$$mx^{"} + bx^{'} + kx = -k_p x - k_v x \tag{1.8}$$

$$mx^{"} + (b + k_v)x^{'} + (k + k_p)x = 0 \tag{1.9}$$

Hence the new coefficients of the equation becomes
$k^{'}$ = k + k_p
$b^{'}$ = b + k_v
Hence for the critically damped system

$$b^{'} = 2\sqrt{mk^{'}} = 32 \tag{1.10}$$

Now for the trajectory equation let us calculate the proportional element

$$k_p = k^{'} - k = 16 - 4 = 12 \tag{1.11}$$

and for differential element

$$k_v = b^{'} - k_v = 32 - 16 = 16 \tag{1.12}$$

Hence the Trajectory equation looks like

$$16e^{"} + 16x^{'} + 12x = 0 \tag{1.13}$$

1.3 Response to change from linear to dynamic friction

The new friction is given by b = 30 sign $(x^{'})$ now applying control loop partitioning we will get,

$$b = 30sign(x^{'}) \tag{1.14}$$
$$f = \alpha f^{'} + \beta \tag{1.15}$$
$$f = mx^{"} + bx^{'} + kx \tag{1.16}$$
$$\alpha = m \tag{1.17}$$
$$f^{'} = x_d^{"} - e^{'}k_v^{'} - ek_p^{'} \tag{1.18}$$
$$\beta = bx^{'} + kx \tag{1.19}$$

hence,

$$e^{"} + k_v^{'}e^{'} + k_p^{'}e = 0 \tag{1.20}$$

The coefficients are of the following value,

$$k_p^{'} = k_{cls} = 16 \tag{1.21}$$
$$k_v^{'} = 2\sqrt{kp^{'}} = 8 \tag{1.22}$$

Therefore the characteristics equation becomes

$$f^{'} = x_d^{"} - 8e^{'} - 16e \tag{1.23}$$
$$f = 16f^{'} + 30sign(x^{'}) + 4x \tag{1.24}$$

1.4 Steady state error

The constant force of disturbance is given by,

$$f_{dist} = 8 \tag{1.25}$$

Also $e^{'}$ and $e^{'}$ are zero,hence in the servo level,

$$f_{dist} = e^{"} + k_v^{'} e^{'} + k_p^{'} e \tag{1.26}$$

$$f_{dist} = k_p^{'} e \tag{1.27}$$

$$e = \frac{f_{dist}}{k_p^{'}} = \frac{8}{16} = \frac{1}{2} \tag{1.28}$$

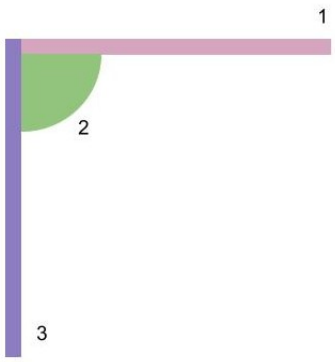
Chapter 2

Visual Servoing

2.1 Image Features

Possible tree features that we find most suitable for such task are two perpendicular lines and angle in between, as presented on Figure 2.1.

Figure 2.1: Proposed features as seen in picture frame, with labels 1,2,3



2.1.1 Difficulty of feature extraction

The features are chose in such a way that they would be extracted as easy as possible. OpenCV shouldn't have a problem extracting the features 1 and 3, from Figure 2.1 (two lines), as long as there is enough contrast on the image. Based on the known line end points, the feature 2 can be easily determined in the picture frame.

2.1.2 Parametrization of the features

All features can be considered as one dimensional, however, for feature 2 calculation, the image is percieved as the 2 dimensional space.

2.1.3 Degrees of freedom, Jacobian and system of equations

There are 6 Degrees of freedom that can be derived based on these features. The center of the image, as well as orientation must be agreed in advance, for example the center can be in the cross point of 2 lines, and the orientation representing the ideal orientation in the image frame, can be taken as presented on Figure 2.1.

Two degrees of freedom along x,y axis can be easily determined by comparing the position of the center in image frame and the desired center.

Degree of freedom along z axis (distance) can be perceived by comparing desired pixel size of the lines with actual in image frame.

One more degree of freedom - Rotational along z is determined by checking the deviation from the horizontal orientation of feature 1 and vertical of feature 3

in image frame.

Two more degrees of freedom - Rotational along x and y axis are a bit more complicated, and for determining it, relation among all three features must be taken into account. To put it simply, first, one can imagine the x axis to be represented by feature 3, with center in crossing point of two lines. Then, pure rotation around x axis can be determined by understanding the distance based on feature 3 length on image frame and then calculating the angle itself by comparing the length of the feature 1 (shorter feature 1 will mark the rotation in negative direction around x, and longer presents the positive rotation). Similar strategy can be adopted for pure rotation around y axis, in case the feature 1, as presented on Figure 2.1 is taken as reference.

In case of combined rotation, it will be impossible to understand the distortion, based on only the lengths of the features 1 and 3, that's why the feature 2 (angle) can be used to approximate the relative position. However, through this method it is not possible to determine exact relative position, its rather that through iterative method it is guaranteed to end up exactly following this marker for all 6DOF-s. Meaning that the resulting system of equation $\dot{s} = J_I \dot{p}_c$ is under-constrained (in some cases), but the system will transfer to determined as long as we follow iterative approach.

In case Euler angles are used for representing the End effector orientation, the vector representing End effector position and orientation would be 6 dimensional. As we have 3 features that are being extracted, Jakobian dimensionality would be $3x6$.

2.2 Find Circle

2.2.1 Implement findCircleFeature(...)

To ensure that our implementation is tolerant of adverse camera images. The picture provided was already a grayscale image of the captured camera inpute. We first used a Gaussian blur filter to filter out the noise and reduce the error caused by the noise. Then we used openCV's Hough Circle Transform function to find a circle in the grayscale image.

2.2.2 Implement transformFrom OpenCVToFF

This transformation maps the position in the **OpenCV frame** to the position in the **feature Frame**, which is simply a translation transformation between the origins in each coordinate. This can be done using the following transformation matrix.

$$\begin{bmatrix} x_{fF} \\ y_{fF} \\ z_{fF} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0.5 \cdot imgwidth \\ 0 & 1 & 0 & 0.5 \cdot imgwidth \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{CV} \\ y_{CV} \\ z_{CV} \\ 1 \end{bmatrix} \tag{2.1}$$

Or simply through formula:

$$\begin{aligned} x_{fF} &= x_{CV} + 0.5 \cdot imgwidth \\ y_{fF} &= y_{CV} + 0.5 \cdot imgwidth \\ z_{fF} &= z_{CV} \end{aligned} \tag{2.2}$$

2.3 Image Jacobian computation

2.3.1 Implement estimateCircleDepth(...)

From the way camera works, we know that our object size in image frame is directly proportional to the focal length of the camera and inversely proportional to the depth (distance between object and camera).

We use this prior knowledge and all other known parameters such as the actual diameter of the circle, the given focal length, and our own parameters. We use this prior knowledge and all other known parameters such as the actual diameter of the circle, the given focal length, and our diameter in the image. Frame returned by our Hough transform function. Then

$$\frac{s_d}{focalLength} = \frac{diameter}{depth} \tag{2.3}$$

Finally we get the **depth**

$$depth = \frac{diameter \cdot focalLength}{2 \cdot crcl.radius} \tag{2.4}$$

2.3.2 Determine the focal length experimentally

The focal length of the camera determines the size of the image for a given object, in our case the diameter of the circle. Simply put, if the camera moves towards the object, its size in the image will increase, and if it moves away from the object, the size will decrease.
This can be easily deduced from the Camera-Obscura equations, which we already know. Camera obscura equations, which give us the exact proportionality between the actual size and the image size as a function of the focal length and the distance between the camera and the object. between the camera and the object, which is given by Equation 2.3.
Since we know our actual diameter and the actual radius of the circle that the Hough transformation function returns, we can move our camera or the object so that both are equal at a given depth distance.
Now we can calculate the d_{des_px} (desired diameter of px)

$$d_{des\_px} = 2 \cdot crcl.radius \tag{2.5}$$

So we can make a conclusion

$$focalLength = depth \tag{2.6}$$

It means that the distance between the object and the camera is the same as the focal length. However, this approach is valid for the object of the smaller size only, since the usual camera focal length is around several cm.

The other approach would be to determine the *FocalLength* through measuring and calculating the difference of the object size in camera frame for two different distances from the camera. Based on Equation 2.3 the following can be concluded for 2 different distances measurements:

$$\begin{aligned} depth_1 &= \frac{diameter \cdot focalLength}{2 \cdot img.crcl.radius_1} \\ depth_2 &= \frac{diameter \cdot focalLength}{2 \cdot img.crcl.radius_2} \end{aligned} \tag{2.7}$$

If the difference of the two distances from the camera can be presented as:

$$delta.depth = depth_1 - depth_2 \tag{2.8}$$

Then, based on the Equations 2.8 and 2.7 the following formula can be derived to calculate the $focalLength$:

$$focalLength = -\frac{delta.depth}{diameter} \cdot \frac{img.crcl.radius_2 - img.crcl.radius_1}{img.crcl.radius_2 \cdot img.crcl.radius_1} \tag{2.9}$$

2.3.3 Specify the image Jacobian and implement getImageJacobianCFToFF(...)

With a 3×3 image Jacobian we can calculate the image features from the Cartesian frame.

$$S_u = \frac{f}{g_z} \cdot g_x \tag{2.10}$$

$$S_v = \frac{f}{g_z} \cdot g_y \tag{2.11}$$

$$S_d = \frac{f}{g_z} \cdot diameter \tag{2.12}$$

With the derivation of the equations 2.10, 2.11 and 2.12 we can get the relationship between the image frame and Cartesian frame. So we can get

$$\dot{S_u} = \frac{f}{g_z} \cdot V_x - \frac{S_u}{g_z} \cdot V_z \tag{2.13}$$

$$\dot{S_v} = \frac{f}{g_z} \cdot V_y - \frac{S_u}{g_z} \cdot V_z \tag{2.14}$$

$$\dot{S_d} = -\frac{f}{g_z^2} \cdot diameter \cdot V_z \tag{2.15}$$

With the velocity's vector $V_x, V_y and V_z$ we can get the elements of the image Jacobian J_v.

$$J_v = \begin{bmatrix} \frac{f}{g_z} & 0 & -\frac{s_u}{g_z} \\ 0 & \frac{f}{g_z} & -\frac{s_v}{g_z} \\ 0 & 0 & -\frac{f}{g_z^2} \cdot \text{ diameter} \end{bmatrix} \tag{2.16}$$

2.4 Velocity vector transformation

2.4.1 Implement $transformVelocityFromCFToEEF(...)$

This transformation between two frames is fixed and can be expressed through the following relations:

$$\begin{aligned} X_{eef} &= -Y_{cf}; \\ Y_{eef} &= X_{cf}; \\ Z_{eef} &= Z_{cf}; \end{aligned} \tag{2.17}$$

2.4.2 Implement $transformVelocityFromEEFToBF(...)$

The transformation from End effector to the **base frame** is not static and it depends of the current robot configuration. The end effector rotation can be determined through the quaternions, as presended in the Equation 2.18

$$q = w + xi + yj + zk \tag{2.18}$$

From quaternion, the rotation matrix can be derived.

$$R(q) = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2zw & 2xz+2yw \\ 2xy+2zw & 1-2x^2-2z^2 & 2yz-2xw \\ 2xz-2yw & 2yz+2xw & 1-2x^2-2y^2 \end{bmatrix} \tag{2.19}$$

The rotation matrix calculation from quaternions is available as the library object method *rotationMatrix*() from *PrMatrix*3 object. Then, **base frame** orientation can be calculated simply as:

$$Vec_{bf} = R(q) \cdot Vec_{eef} \tag{2.20}$$

2.5 Workspace Limitation

2.5.1 Avoid singular configurations

To avoid the singular configurations with the 0.85m radius of a sphere we should use the followed equation

$$(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2 \leq r^2 \tag{2.21}$$

Since we know that the radius center is in the base frame:

$$x_c = y_c = z_c = 0 \tag{2.22}$$

Maximum radius is $r = 0.85$. After substitution the following is true:

$$(x)^2 + (y)^2 + (z)^2 \leq (0,85)^2 \tag{2.23}$$

The end effector will be inside the sphere if the position of end effector satisfy the equation 2.23.

Chapter 3

submission

Student Name	A1	B1	B2	B3	B4	B5
Vasilije Rakcevic	x	x	x	x	x	x
Jingsheng Lyu	x	x	x	x	x	
Prathamesh More	x	x	x	x		x