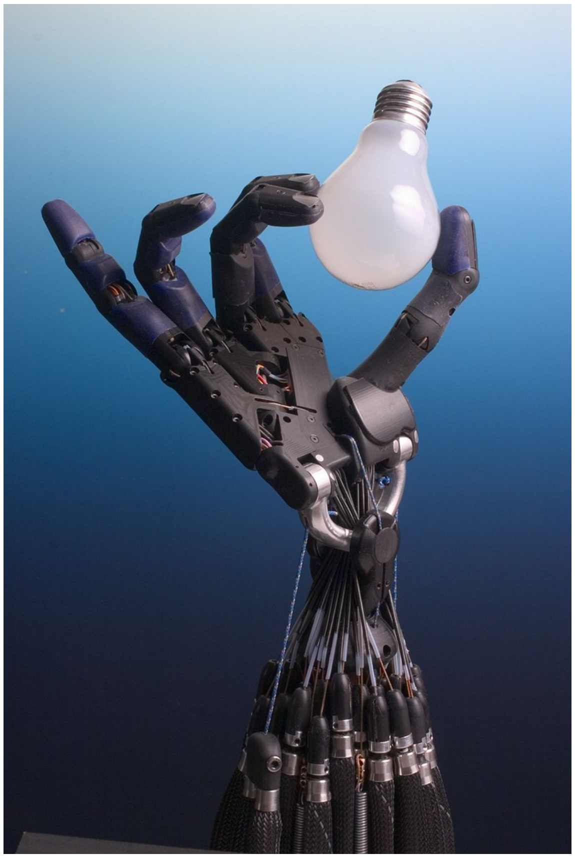
01/12/2018



d

Robotics assignment 3

By
Abhiraj Bishnoi
Götz-Gustav Dietrich
Wenbo Wu
Ilias Amri

A - Theory

1.a $m\ddot{x} + b\dot{x} + kx = f \quad \textcircled{1}; \quad m = 2 \quad b = 12 \quad k = 18$

$f = 0$

$\Leftrightarrow 0 = 2\ddot{x} + 12\dot{x} + 18x \quad | \, \mathcal{L}$

$\Leftrightarrow 0 = 2s^2 + 12s + 18 \quad | \div 2$

$\Leftrightarrow 0 = s^2 + 6s + 9$

with: $f = s^2 + 2\xi_n\omega_n s + \omega_n^2$ $\quad \Rightarrow \omega_n^2 = 9 \quad | \sqrt{\cdot}$

$\Leftrightarrow \underline{\underline{\omega_n = 3}}$

$\Rightarrow 2\xi_n\omega_n = 6$

$\Leftrightarrow 6\xi_n = 6$

$\Leftrightarrow \underline{\underline{\xi_n = 1}}$

With $\xi_n = 1$ the system is critically damped.

1.b $f = -k_p(x - x_d) - k_v\dot{x} \quad | \, x_d = 0$

$\Leftrightarrow f = -k_p x - k_v\dot{x} \quad \textcircled{2}$

$\textcircled{1}, \textcircled{2} \Rightarrow -k_p x - k_v\dot{x} = 2\ddot{x} + 12\dot{x} + 18x \quad | +k_v\dot{x} + k_p x$

$\Leftrightarrow 0 = 2\ddot{x} + (12 + k_v)\dot{x} + (18 + k_p)x \quad | \div 2 \quad \text{with } k_{CLS} = 32$

$$\Leftrightarrow 0 = \ddot{x} + \frac{12 + k_v}{2}\dot{x} + 16x$$

$\Rightarrow k_{CLS} = 32 = 18 + k_p$ $\quad \Rightarrow \omega_n = 16 \quad | \sqrt{\cdot}$

$\Leftrightarrow \underline{\underline{k_p = 14}}$ $\quad \Leftrightarrow \omega_n = 4$

$$\Rightarrow 2\xi_n\omega_n = 8\xi_n = \frac{12 + k_v}{2} \quad | \div 8$$

$$\Leftrightarrow \xi_n = \frac{12 + k_v}{16} \overset{!}{=} 1 \qquad | \cdot 16 \mid -12$$

$$\Leftrightarrow \underline{\underline{k_v = 4}}$$

$$\Rightarrow \underline{\underline{f = -14x - 4\dot{x}}}$$

1.c $f = m\ddot{x} + b\dot{x} + kx = \alpha f' + \beta$

with: $m = 2, \quad b\dot{x} = 30 \cdot sign(\dot{x}), \quad k = 18, \quad k_{CLS} = 32$

$$f = 2\ddot{x} + 30 \cdot sign(\dot{x}) + 18x$$

$$\Rightarrow \underline{\underline{\alpha = 2}}, \quad \underline{\underline{\beta = 30 \cdot sign(\dot{x})}}$$

$$f' = \ddot{x}_d - k_v'(\dot{x} - \dot{x}_d) - k_p'(x - x_d)$$

$$\Rightarrow f = 2(\ddot{x}_d - k_v'(\dot{x} - \dot{x}_d) - k_p'(x - x_d)) + 30 \cdot sign(\dot{x}) + 18x \qquad | \; e = x - x_d$$

$$\Leftrightarrow f = 2(\ddot{x}_d - k_v'\dot{e} - k_p'e) + 30 \cdot sign(\dot{x}) + 18x$$

$$\Rightarrow f' = \ddot{x}_d - k_v'\dot{e} - k_p'e = \ddot{x} \qquad | \; \ddot{e} = \ddot{x} - \ddot{x}_d$$

$$\Leftrightarrow 0 = -\ddot{e} - k_v'\dot{e} - k_p'e \qquad | \cdot(-1)$$

$$\Leftrightarrow 0 = \ddot{e} + k_v'\dot{e} + k_p'e$$

$$\Rightarrow \alpha k_p' = k_{CLS} \qquad \Leftrightarrow 2k_p' = 32 \qquad | \div 2$$

$$\Leftrightarrow \underline{\underline{k_p' = 16}}$$

$$k_v' = 2\sqrt{k_p'}$$

$$\Leftrightarrow k_v' = 2\sqrt{16} = \underline{\underline{8}}$$

1.d $f_{dist} = \ddot{e} + k_v\dot{e} + k_pe = 4 \qquad |$ when steady: $\ddot{e} = \dot{e} = 0$

$$\Rightarrow f_{dist} = k_pe$$

$$\Leftrightarrow 4 = 16e \qquad \mid \div 16$$

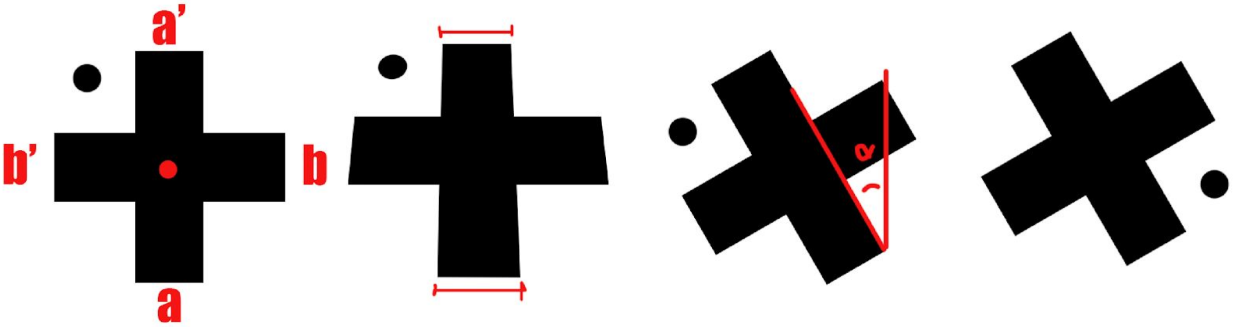
$$\Leftrightarrow \underline{\underline{e = \frac{1}{4}}}$$

B - Visual servoing

1.

Image features.

After a lot of reflection we settled on on this figure :



By knowing every variable we can get the rotation, the inclination and the position of the diagram :

- The ratio between a and a', b and b' can help us determine if it's inclined and at which angle.
- The overall size of the figure gives us the distance to the camera (like for the circle).
- The middle point gives us the x and y positions.
- The angle alpha helps us know the rotation.
- And the point on upper left corner is there to differentiate the two-position illustrated on the right. By comparing its position to the middle point.

There is only one problem. If you incline the figure you still get an angle alpha (but still no rotation)that has to be corrected numerically.

These features were chosen because it’s easy to extract them. we only have lines, and a circle.

We can control all of the degrees of freedom we this figure (6 in our case).

The image jacobian should be a 6 by 7 matrix.

1)

Triangle:

Triangle is easy to extract, because it formed by 3 straight lines, we can use Hough transform to extract these 3 lines.

We can use the location of 3 edges of the triangle as parameters. So there are 6 parameters:

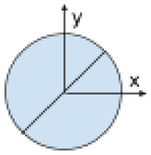
[a1,d1,a2,d2,a3,d3]   (ai,di) are angles and distance of edges.

Task needs 6 DoFs of the operational space(3 for rotation, 3 for translation), and we have 6 parameters, so it should be well formed and the image jacobian matrix should be 6 by 6.

But the triangle is not easy to track, although we can use Hough Transform to extract these 3 lines, it’s hard to find a point of the feature in operational space. When we use a circle as feature, we can track the position of circle’s center. But in triangle it’s not so easy to find such a point, so it’s difficult to get a relationship between parameters of triangle and the position and orientation in operational space.

2)

Circle with a line:



This feature is easy to extract, we can use circle hough transform and line hough transform, then we can get the position of the circle’s center , the radius of the circle and the angle of the line.

So we get 4 parameters:

$[x, y, r, \theta]$

With these 4 parameters we can track the position of the circle, and the orientation around z axis.

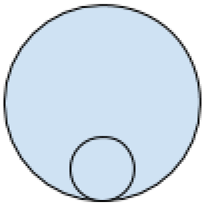
When we only have a circle it's impossible to detect the rotation of the feature around z axis, because when the feature rotate around z axis x,y,r won't change. With a line we can detect it.

So with this feature we can only control 4 DoFs of the operational space.

There are 4 parameters and 4 DoFs, so the image jacobian matrix should be 4 by 4.

3）

a circle contains another circle



In this feature is the radius of the outside circle much bigger than the inside one, so that when we rotate the feature the outside circle looks still bigger than the inside one in image , then we can use circle hough transform and always know which information belongs to which circle.

This feature is not hard to extract, we only need to use circle hough transform.

Then we get 6 parameters:

[x1,y1,r1,x2,y2,r2]

Theoretically we have 6 parameters so we can solve the problem with 6 DoFs. The image jacobian matrix should be 6 by 6 and the resulting system of equation $\dot{s} = J_I \dot{p_c}$ should be well-formed.

2.a

In this first implementation part we used a HoughCircles function offered by OpenCV and checked all circles found for their radius to ensure that we only hand over the biggest circle.

To make our implementation tolerant to adverse camera images we add a GaussianBlur function also provided by OpenCV to smooth edges in pixelated images. We also checked the detection thresholds and set the upper threshold for the internal canny edge detection lower than in the example to 150 and the threshold for the center detection to 45.

2.b

Then since the openCV frame is centered at the upper left corner, we have to translate it to the right by half of the image size (on both axis).

3.a

We have from the geometry of the camera that the relationship between the focal length (f) and the circle diameter in the image (D') is the same as the distance between camera and the observed circle (z) and the real diameter (D):

$$\frac{f}{D'} = \frac{Z}{D}$$

This relationship can than be solved for the distance (z):

$$\Rightarrow z = f\frac{D}{D'}$$

Where z is the depth, f the focal length, D the real diameter of the circle, and D' the diameter of the image circle.

3.b

To get the focal length we can solve the equation :

We know the size of the real life circle (D), the size of the image circle (D' = 2 times radius of image circle), and can measure the distance of the circle to the camera (z).

$$f = z\frac{D'}{D}$$

We take a circle of diameter equal to the height of the image plane. By moving the circle around back and forth perpendicular to the plane of the camera lens we change the size of the image of the circle in the image plane. At some point, we have the circle in the image plane perfectly cover the entire image plane with the diameter of the circle exactly equal to the height of the image plane. By the mathematics of the formation of the image on the image plane, we know that once we have reached this point the distance of the real circle from the camera lens is equal to the focal length of the camera lens.

$$f = z\frac{D'}{D}$$

Mathematically, when we empirically reach the condition D = D' = Height of the image plane, f = z = focal length of the camera.

3.c

With geometry we can get:

$$\begin{array}{ccc} \frac{S_u}{f} = \frac{x}{z} & & S_u = f \cdot \frac{x}{z} \\ \frac{S_v}{f} = \frac{y}{z} & \Rightarrow & S_v = f \cdot \frac{y}{z} \\ \frac{S_D}{f} = \frac{D}{z} & & S_D = f \cdot \frac{D}{z} \end{array}$$

$$\Downarrow \frac{d}{dt}$$

$$\begin{aligned} \dot{S_u} &= \frac{f}{z}\dot{x} - f\frac{x}{z^2}\dot{z} \\ \dot{S_v} &= \frac{f}{z}\dot{y} - f\frac{y}{z^2}\dot{z} \\ \dot{S_D} &= -f\frac{D}{z^2}\dot{z} \end{aligned}$$

Because $S_u = f \cdot \frac{x}{z}, \ S_v = f \cdot \frac{y}{z}$:

$$\dot{S_u} = \frac{f}{z}\dot{x} - \frac{S_u}{z}\dot{z}$$
$$\dot{S_v} = \frac{f}{z}\dot{y} - \frac{S_v}{z}\dot{z}$$
$$\dot{S_D} = -f\frac{D}{z^2}\dot{z}$$

Then we can get image jacobian with:

$$\begin{pmatrix}\dot{S_u}\\ \dot{S_v}\\ \dot{S_D}\end{pmatrix} = \begin{pmatrix}\frac{f}{z} & 0 & -\frac{S_u}{z}\\ 0 & \frac{f}{z} & -\frac{S_v}{z}\\ 0 & 0 & -f\frac{D}{z^2}\end{pmatrix}\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{pmatrix}$$

Because what we move is camera not image, the velocities should be negative:

$$\begin{pmatrix}\dot{S_u}\\ \dot{S_v}\\ \dot{S_D}\end{pmatrix} = \begin{pmatrix}-\frac{f}{z} & 0 & \frac{S_u}{z}\\ 0 & -\frac{f}{z} & \frac{S_v}{z}\\ 0 & 0 & f\frac{D}{z^2}\end{pmatrix}\begin{pmatrix}-\dot{x}\\ -\dot{y}\\ -\dot{z}\end{pmatrix}$$

so the image jacobian is:

$$\begin{pmatrix}-\frac{f}{z} & 0 & \frac{S_u}{z}\\ 0 & -\frac{f}{z} & \frac{S_v}{z}\\ 0 & 0 & f\frac{D}{z^2}\end{pmatrix}$$

4.a

Velocities have no position, so the translation of the frame has no influence on Velocities, but rotation does.

transformVelocityFromCFToEEF:

We can get camera frame by rotating end-effector frame around z axis 90 degree.

So here we use the rotation matrix:

$$R = \begin{pmatrix}cos(\pi/2) & -sin(\pi/2) & 0\\ sin(\pi/2) & cos(\pi/2) & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$
\begin{aligned}
Vector_{eef} &= R * Vector_{cf} \\
Vector_{eef}[0] &= -Vector_{cf}[1] \\
Vector_{eef}[1] &= \quad Vector_{cf}[0] \\
Vector_{eef}[2] &= \quad Vector_{cf}[2]
\end{aligned}
$$

transformVelocityFromEEFToBF:

We have the orientation quaternion(q0, q1, q2, q3) of the end-effector in base frame, and it can be converted to a rotation matrix:

$$
Tr = \begin{pmatrix} 1-2q_2^2-2q_3^2 & 2q_1q_2-2q_3q_0 & 2q_1q_3+2q_2q_0 \\ 2q_1q_2+2q_3q_0 & 1-2q_1^2-2q_3^2 & 2q_2q_3-2q_1q_0 \\ 2q_1q_3-2q_2q_0 & 2q_2q_3+2q_1q_0 & 1-2q_1^2-2q_2^2 \end{pmatrix}
$$

$$
Vector_{bf} = Tr * Vector_{eef}
$$

5.a

To ensure that the robot stays inside a sphere of 0.85 meters around the base frame we check the length of the end-effector vector in the base frame. If the new vector, after adding the next step to its actual position, is shorter than 0.85 meters it is still inside the sphere and is allowed to move the next step. If it the new vector is longer it stays in the actual position.

To avoid singularities we check if the rank of the Image Jacobian is full (in our case the full rank of a 3x3 matrix is 3). If it is not, the next step would lead to a singularity when we just use an inverse to calculate the next step. Therefore, we use a pseudo inverse to approximate a way around the singularity in this case.

Student name	(A)	(B1)	B2.a	B2.b	B3.a	B3.b	B3.c	B4	B5
Abhiraj Bishnoi		x	x	x	x	x		x	x
Götz-Gustav Dietrich	x	x	x		x		x		x
Wenbo Wu	x	x		x	x		x	x	
Ilias Amri		x	x	x		x		x	x