# Lab 8 Exercises

## 1 Image Based Visual Servoing for UR10

### 1.1 Definitions

Using "Image Based Visual Servoing" to control the camera mounted on the end-effector such that the camera sees a square target in the image with its corners located in:

```
p1Star = [662 362];  p2Star=[362 362];  p3Star=[362 662]; p4Star = [662 662];
```

Given that the Cartesian coordinates of the target's corners are:

```
P1 = [1.8, -0.25, 1.25];   P2 = [1.8, 0.25, 1.25];   P3 = [1.8, 0.25, 0.75];   P4 = [1.8, -0.25, 0.75];
```

and the initial configuration of the UR10 is given by:

```
q0 = [pi/2; -pi/3; -pi/3; -pi/6; 0; 0];
```

The camera mounted on the UR10 is a perspective Camera (CentralCamera) with focal length of 0.08, pixel size of 10e-5 and a resolution of 1024 x1024 with the centre point exactly in the middle of image plane, which gets images at 25fps. In Chapter 11 (Section 11.1) of the textbook you can find out all about the perspective model and how to use the Vision Toolbox for the purposes required in this Lab Exercise (see LabStarter of this week).

Locate the camera at the position of the end effector `Tc0` (tip you need to find the location of the end-effector first)

```
cam.T = Tc0;
```

Define the visual servoing gain (0< lambda < 1) and a value for the depth. Given than the points are roughly at 1.8m from the camera use `depth = 1.8;`

### 1.2 Initialise the simulation (Display in 3D world)

#### 1.2.1 Display the UR10

#### 1.2.2 Display Camera in 3D (at the location of the end-effector). The pose of the camera is given by

```
cam.plot_camera('Tcam',Tc0);
```

#### 1.2.3 Plot 3D points

### 1.3 Initialise the simulation (Display in 2D image) – Note that the Vision Toolbox already does this plotting for you, if all the 3D points are in the correct 3D location and your camera is correctly located on the end-effector your image view should look like Fig 1

#### 1.3.1 Display image view. Using the Vision Toolbox, plot the target in the Image View

```
cam.plot(pStar, '*');
```

#### 1.3.2 Display current view using the projection of the 3D points into the image at the current camera position `Tc0`

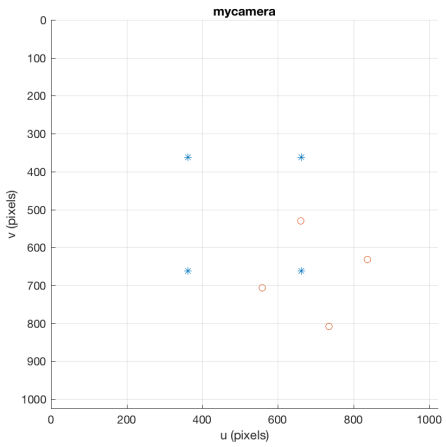


*Figure 1. Image View*

### 1.4 **Loop for max 200 steps:**

1.4.1 Compute the view of the camera by projecting the 3D points into the image plane

```
uv = cam.plot(P);
```

1.4.2 Calculate the error in the image `e`

1.4.3 Compute the current Image Jacobian using the Vision Toolbox function

```
J = cam.visjac_p(uv, depth);
```

1.4.4 Compute the desired velocity of the camera given the error and the Image Jacobian (this in your image based visual servoing lecture notes or in Chapter 15 Section 15.2 of your textbook). Note that this is the desired velocity of the end-effector

```
v =  lambda * pinv(J) * e
```

1.4.5 Calculate the joint velocities of the UR10 (tip you need to compute the robot's Jacobian with respect the end effector-frame). If needed limit these angular velocities to 180°/sec max per joint

1.4.6 Calculate the joint displacements (tip `DeltaT = 1/fps`)

1.4.7 Apply the displacement to the UR10 and update camera location

1.4.8 Display the UR10 and camera in the current position

1.4.9 When the desired image location has been reached or the maximum number of steps have been reach, finish the loop (tip if the max number of steps have been reached before the points are in the target position, augment "lambda" value)

### 1.5 **Plot results**

1.5.1 Plot points trajectory in the image, camera position, orientation and velocities and joint angles and velocities