Introduction to Vision and Robotics 2020-21

Lab 2

Chamfer link matching

October 2020

This lab will have you continuing with vision challenges, specifically with Chamfer matching for the robot's links. Use instructions from previous lab to install the package.

1 Chamfer matching

1.1 Purpose

In the last session the task was to find the positions of the coloured circles, which then allowed you to obtain the joint states of the robot. This lab will have you implement Chamfer matching to find the links in the world, and from here obtain the joint states.

1.2 Method

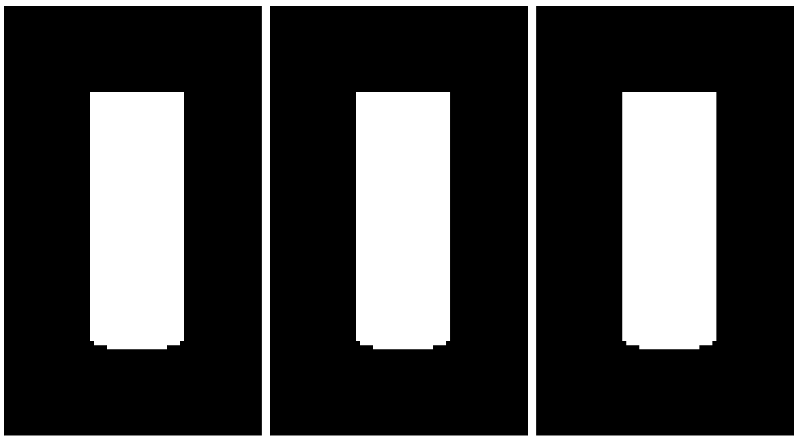
Chamfer matching is the process of moving a template through various configurations on the screen and minimizing the error between this template and the desired object. In this scenario the desired objects are the links. The templates are provided and can be loaded in the **image_processing.py** file under the names: **self.link1, self.link2, self.link3**. To load the template images follow these steps:

1. Uncomment the following line in the image_processing.py file

   ```
   #  self.link1 = cv2.inRange(cv2.imread('link1.png', 1),
   (200, 200, 200), (255, 255, 255))
   #  self.link2 = cv2.inRange(cv2.imread('link2.png', 1),
   (200, 200, 200), (255, 255, 255))
   #  self.link3 = cv2.inRange(cv2.imread('link3.png', 1),
   (200, 200, 200), (255, 255, 255))
   ```

2. copy the three image files in **ivr_lab** folder named, link1.png, link2.png, and link3.png into the **catkin_ws** folder.

Now, to estimate the joints' angles, we will rotate the templates by various angles, and perform chamfer matching to see if any configuration of the rotated template matches the links. Whichever configuration gives



(a) Link 1 (b) Link 2 (c) Link 3

Figure 1: Templates

us the minimal error we can use this as our best estimate of knowing where the link is. Using this method we can retrieve the link rotations and consequently the joint state of the robot.

A rough outline of the algorithm is:

1. Isolate the desired link.
2. Rotate the template with various angles and find the configuration of a template that matches the isolated link. This way you can estimate the rotation of each link.
3. Use the rotation of each link to calculate the joint angle between the links (i.e. converting the angles to reflect the coordinate frame of the respective link).

The following subsections have more details for each of the above steps.

1.3 Isolating the desired link

The links as you can see have different colours from the background, they are black. Therefore we can use the same thresholding technique as we did for the coloured circles in the last session here to isolate the links from the rest of the image.

After we have isolated the links, we can use the information from the coloured circles to isolate each link. Find the position of circles at the beginning and end of each link, use the positions to estimate the position of centre of the link. Next, you can pick a window of fixed size around the link and crop it.

Hint: Make sure the selected window has the same size as the template for that link, this will make chamfer matching algorithm easier in the next steps. You can get the size of each template using the command **.shape**, i.e., the dimensions of link 1 are **self.link1.shape**.

1.4 Using a template to find the estimate of link angle

This step is the key for Chamfer matching. The initial step is to generate the configurations for your search space by rotating the template to various angles for instance from 0 to 360 degrees. Then for each configuration perform chamfer matching.

1. Transform your template into a configuration (rotate it by a specific angle).
2. In the source binary image – i.e. the image where you want to find the link—calculate the distance from every pixel to the nearest 'on' pixel. This is known as a distance transform.
3. Do element wise multiplication of the above distance matrix and the transformed template, and sum the elements (i.e. the distances). This is the error for chamfer matching.

Once you have performed this on every configuration use the configuration that returns the smallest error. You may have noticed that this can be time consuming, and will need to be careful on how many configurations to generate. Too few will result in inaccurate estimates, too many will make your system very slow

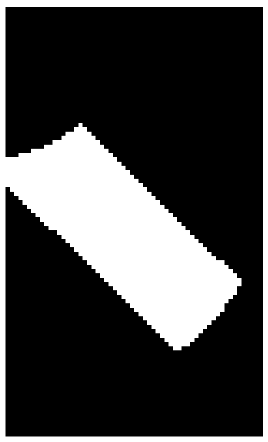
1.5 Hints:

Generating configurations in your search space Typically the search space would include the various positions of the link in the image as well as the rotations. However, this leads to a huge search space. To help you narrow this down it would be suggested to use some other information in the image. Two suggestions are:

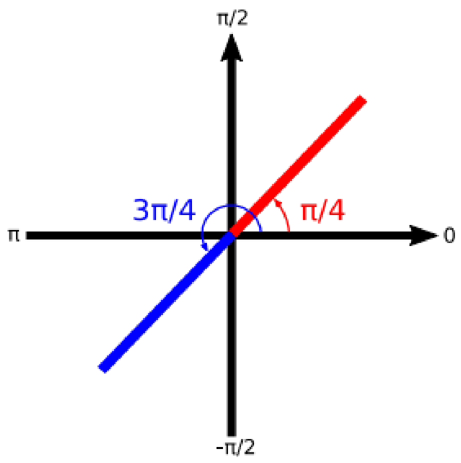
1. As mentioned before, detect the centre of each link and crop the binary image of the robot to match the size of the provided templates.
   You can use the techniques of the previous labs to find the centre. From here you can then use the relevant indexes with the shape of the template, obtained by: **self.link{NUM}.shape**, to subselect around this centre point. An example of this isolated link image looks like Fig. 2a.
2. Detect the quadrant the link is pointing in to limit the number of rotation values to use with the template. You can use the previous lab session to detect the circles, which tells you where the end of the links are. This will therefore say if they are pointing in one of our defined quadrants; upper left, upper right, lower left, or lower right. If it was upper left, for example, then we know that we should only try rotation values between 0 and $\pi/2$, as seen in Fig. 2b. You can then use the previous angles to determine the connecting links quadrants. You will still have to decide on your step size for the number of rotations to test.

Transform your template into that configuration For the transformations (i.e., rotating the template by a specific angle) you may wish to look at OpenCV Geometric Image Transformations `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html`. Note that this command rotates the images clockwise but the positive direction for robot rotation is counter clockwise. So if a link is in upper left quadrant and you want to rotate the image using OpenCV command you should try angles between π and $3\pi/2$ instead of $\pi/2$ and π. Use the **cv2.imshow** command to plot the rotated template and make sure you are rotating the template in the proper direction.

Distance transform After you have isolated a link like in Fig. 2a., you can calculate the distance of every pixel in this binary image of the link to the nearest 'on' pixel in the same binary image. This is known as a distance transform. You can look at the method of **cv2.distanceTransform** in the OpenCV library. `https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html?`. This function however returns distance to the closest 0 pixel, so you will need to invert your source binary image first (change zeros to ones and vice versa, you can use **cv2.bitwise_not()** command).



(a) Link 1 isolated



(b) Link in quadrant 1 and 3

Figure 2: Link in quadrant 1 and 3.

Minimizing the error Once you have the distance transformed image and each of the configured rotated templates, you can then calculate the sum of the distances between the link image and the template. This can be done by taking the sum of all the distance transform image pixels that overlap with one of the 'on' pixels in the transformed template image. Simply put, performing an element wise multiplication, and then taking the sum of the output will get you total distance. From this you should be able to find the correct configuration as the one with the minimal total distance. You should try this out with pen and paper first with a toy example (a 3×3 matrix for example) so you fully understand what is to be expected.

Use the rotation of each link to calculate the joint angle between the links Finally you can use these rotations to calculate the joint state of the links with respect to the previous links. This should be able to be done similarly to the last lab. You will output the centre coordinate and estimated joint angle of the link, and then return the entire joint state of the robot.

1.6 Outcome

The outcome of this part of the lab will be the same as the previous lab; the arm should move towards a desired joint configuration and you should be able to measure those joints values. Your results should be the same or more accurate than the previous method.