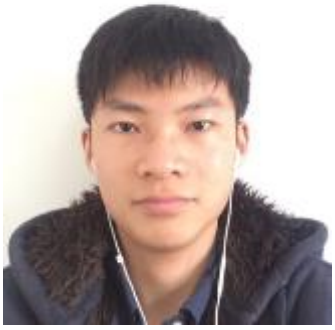
Assignment 3: Ransac, Distance Transform and Chamfer Matching



Name : Suozhi Qi

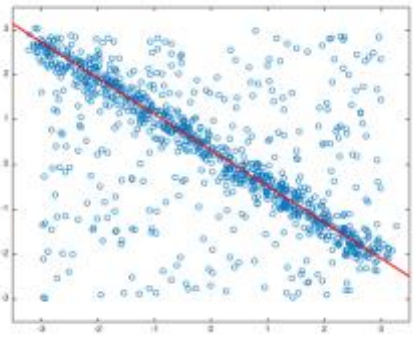
Dept: Computer Science

I Understanding RANSAC

1.1 Ransac for line fitting

1. A: Two. Because we need at least 2 points to determine a line.
2. A: I choose the threshold t to be 0.3. Though I found that choosing it as 0.2, 0.4, 0.5 all seems to work similarly.
3. A: I first set the inlierratio to be 0.5 in my matlab file, so the outlierratio is 0.5 in this case. In our lecture, we know $N = \log(1 - p)/\log(1 - (1 - e)^s)$. Here, $p = 0.9999$, $e = 0.5$, $s = 2$. And thus we got N to be 33.

Below is the plot I have got:



1.2 Ransac for Affine Fitting

1. A: I chose s to be three because the hint of the instructions asks us to choose 3 points. Also I think three is the minimum number to determine a 2-D space.
2. A: I choose the threshold to be 0.7 after many tests.
   First, I set the 6X6 matrix to be:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ x_3 & y_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_3 & y_3 & 1 \end{bmatrix}$$

Then I set the 6X1 transformed matrix to be:

$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix}$$

Then using matlab I got x to be:

$$\begin{bmatrix} 0.7880 & 0.1205 & 10.0879 \\ -0.1386 & 0.6907 & 169.2165 \end{bmatrix}$$

3. A: Sample count was calculated to be between 23 to 33. Please refer to my matlab code. Below are two images: left is the transformed image provided, and right is the transformed image using my newly generated parameter matrix x.


II Distance transform and chamfer matching

2.1 Distance Transform

1. A: The image is shown below:

2. A: Please refer to the matlab file for the code. Here is the image:

4. A:
   Chessboard distance transform:

   Cityblock distance transform:

Euclidean distance transform:

Computation of the chessboard image uses $\max(|x1 - x2|, |y1 - y2|)$. This means that two different points, one is some distance horizontally from the features, and the other is the same distance vertically from the features, will have the same distance according to this method of computation. For a particular feature point, all points that have the same distance to it will form a square whose two edges are parallel to the horizontal line and the other two edges are parallel to the vertical line. So we can clearly see some symmetry on the 45 degree line which acts as a "mirror" for the points on the horizontal line and the points on the vertical line.

Computation of the cityblock image uses $|x1 - x2| + |y1 - y2|$. This means that for a particular feature point (edge point), all points that have the same distance to it will form a square which is 45 degree from the horizontal line. That is why it is called as a "block". So the vertical lines across the feature points should act like a "mirror" for the points by their sides. This explains why we see symmetry along the vertical line in the second image (cityblock image).

Computation of the Euclidean image uses $\text{sqrt}((x1 - x2)^2 + (y1 - y2)^2)$. This means that for a particular feature point, all points that have the same distance to it will form a "circle". This explains why we see some "rounded" structure in the last image (Euclidean image).

In general, for images with many "round" features, Euclidean method should work the best. But if the image has a lot of square features, I would consider cityblock or chessboard method.

2.2 Chamfer matching

1. A: The question did not specifically tells us which distance transform image to use. So I just used my favorite distance transform image "Euclidean image" for this purpose. The min chamfer distance I got is 1.0618. Please refer to my matlab code for details in implementation.

2. A: As to the calculation of distance transform, we do not need to do the traditional exhaustive search. Instead, we can first locate those edge points that have a "1" instead of zero on it, set those as 0 in our distance transform image, and then do a bread-first search from these points. This, instead will give a O(mn) time complexity since we only search each point one time without repeating.

   As to the exhaustive search on the Chamfer matching between the template image and our distance transform image, my thought for an alternative way is that we can first try to locate some points with distinct features on the template image. This could be done in a similar method as SIFT. Then instead of trying to match all points on the template with our transform image, we just need to match those distinct feature points and calculate whether we can get a minimum Chamfer distance in this case. We can get our matching position using this method as well.

3. A: Below is the superimposed image. I just remove the blue and green pixels on the position where the template should be. So the edge is shown in red.

III Fast directional chamfer matching (Grad credits)

This paper proposed a modified chamfer matching algorithm with higher accuracy and lower computational time, which allows better object recognition and shape match in images. For decades, people put efforts on modeling object recognition with as few exemplars as possible. It

leads to the study of shape matching because it gives simplicity yet high invariances. Many algorithms are devised focusing on how to match shapes under cluttered background. However, most modern designs fail to satisfy speed requirement because of high complexity. Thus, the authors focused on revising the classic chamfer matching model and comparing their results with other modified chamfer matching designs.

Chamfer matching is an application in finding alignment within two edge maps. It does not work well when background clutter exists. Optimization includes adding edge orientation mismatch cost into the chamfer distance, which is called oriented chamfer matching (OCM). The largest drawback of such method is the sensitivity to orientation channels and their boundaries. Thus, in this paper, the authors proposed another cost function named directional chamfer matching (DCM) to solve the problem. DCM score is based on the chamfer distance to points in $\mathbb{R}^3$, which gives more robust matching against clutter environment than OCM.

Another major improvement achieved in this paper is to reduce the computational time from linear to sublinear. Sublinear time is achieved in three stages by first give a linear representation of the template edges, then describe a 3D distance transform representation and lastly present a distance transform integral image representation. A variant of RANSAC algorithm is applied to compute the first stage. It initiates by defining a subset of points with directions. Lines are then formed based on points that satisfy a line equation and give a continuous structure. Finally, only the line with largest support remains. After iteration of these steps, only points with certain structure survive, which efficiently rules out the noise. In the second stage, in order to meet the requirement of the DCM cost function, a 3D distance transform representation (DT3) is used to calculate the matching cost within linear time. The three dimensions are the x, y location on the plane and the quantized edge orientations, respectively. By using DT3v, the DCM score can be computed in linear time. In the last stage, the DCM score function is reformulated and for each orientation channel i, O(m) complexity becomes only an upper bound. Empirically, the authors showed that the computational time is improved to sublinear.

After explaining the modified chamfer matching model, authors conducted their model to three examples, object detection and localization, human post estimation and 3D post estimation. Higher or equal accuracy and 45x faster processing time are proved when compared to other algorithms.