Name: Abhiram Durgaraju
CSE 803 Computer Vision Homework 2

Problem 1

1. My program visionhw2_q1_8con.m reads one input image
2. My program thresholds the objects to 1 and background to 0. For each image, the threshold is different, and the thresholding is done manually, not automatically. I chose a threshold of 60 for 'hw2-2B.jpg', a threshold of 140 for 'hw2-3A.jpg', and a threshold of 240 for my chosen image 'shapes.jpg'.
3. Before writing the connected component algorithm, I had to make sure all 3 images were similar. In order to do that, for 'hw2-3A.jpg', I had to swap 0's and 1's to bring objects to the foreground after thresholding. For 'shapes.jpg', the image was initially a rgb, so I had to use rgb2gray to convert the image to grayscale first before thresholding. These lines are commented out in the program near the beginning, but are necessary to uncomment for my program to function for each specific file.

   I used the 2nd approach we learned in class, via raster scanning, but for 8 connectivity instead of 4, to construct my algorithm for connected component labeling. I used an equivalency matrix to then relabel all of the mislabeled connected components by reiterating through all of the labels an extra time. The variable 'output matrix' in my program displays area, centroid, the three moments, minimum inertia, and maximum inertia for each label.

4.

   Below are my results for 'hw2-2B.jpg':

	Blobs (there are 4)			
Label	1	2	3	4
Area	46.0000	12.0000	8.0000	11.0000
Centroid (r)	9.0000	6.5000	9.5000	18.0000
Centroid (c)	7.5000	5.0000	7.5000	7.0000
Second-row moment	23.5652	1.2500	5.2500	0
Second-column moment	20.6848	0.6667	5.2500	10.0000
Second-mixed moment	0	0	-5.2500	0
Maximum Inertia (deg)	-90.0000	-90.0000	-45.0000	0
Minimum Inertia (deg)	0	0	-135.0000	-90.0000

Here is proof of my MATLAB for image 2A:

```
>> visionhw2_q1_8con
>> output_matrix

output_matrix =

    1.0000    2.0000    3.0000    4.0000
   46.0000   12.0000    8.0000   11.0000
    9.0000    6.5000    9.5000   18.0000
    7.5000    5.0000    7.5000    7.0000
   23.5652    1.2500    5.2500         0
   20.6848    0.6667    5.2500   10.0000
         0         0   -5.2500         0
  -90.0000  -90.0000  -45.0000         0
         0         0 -135.0000  -90.0000
```

Name: Abhiram Durgaraju

All of the values calculated in my program should be correct for the threshold chosen. Area was calculated by simply counting the number of pixels of each label. Centroid was calculated by averaging the row and column coordinate of each pixel.

The maximum and minimum inertias were calculated with the parameters given in the book on page 92 for cases where division by 0 would cause a problem. For others, the standard formula was used. Here were the edge cases as described by the textbook:

1. $\mu_{rc} = 0$ and $\mu_{rr} > \mu_{cc}$
   The major axis is oriented at an angle of $-90°$ counterclockwise from the column axis and has a length of $4\mu_{rr}^{1/2}$. The minor axis is oriented at an angle of $0°$ counterclockwise from the column axis and has a length of $4\mu_{cc}^{1/2}$.

2. $\mu_{rc} = 0$ and $\mu_{rr} \leq \mu_{cc}$
   The major axis is oriented at an angle of $0°$ counterclockwise from the column axis and has a length of $4\mu_{cc}^{1/2}$. The minor axis is oriented at an angle of $-90°$ counterclockwise from the column axis and has a length of $4\mu_{rr}^{1/2}$.

3. $\mu_{rc} \neq 0$ and $\mu_{rr} \leq \mu_{cc}$
   The major axis is oriented at an angle of

$$\tan^{-1}\left[-2\mu_{rc}\text{Ø}\mu_{rr} - \mu_{cc} + \left((\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2\right)^{1/2}\right]$$

The standard formula was used in all other cases, as listed below, according to the textbook:

$$\begin{aligned}
\tan 2\hat{\alpha} &= \frac{2 \sum (r - \bar{r})(c - \bar{c})}{\sum (r - \bar{r})(r - \bar{r}) - \sum (c - \bar{c})(c - \bar{c})} \\
&= \frac{\frac{1}{A}\, 2 \sum (r - \bar{r})(c - \bar{c})}{\frac{1}{A} \sum (r - \bar{r})(r - \bar{r}) - \frac{1}{A} \sum (c - \bar{c})(c - \bar{c})} \\
&= \frac{2\ \mu_{rc}}{\mu_{rr} - \mu_{cc}}
\end{aligned}$$

Axis of minimum inertia can be thought of as the axis which intersects the most pixels and therefore has the fewest pixels rotating around the axis. The opposite is true for maximum inertia. Using the formula above, my program calculated these correctly too I believe.

Name: Abhiram Durgaraju
CSE 803 Computer Vision Homework 2

Below is my results for 'hw-3A.jpg':

	Blobs (there are 20)										
Label	1	2	5	6	9	10	12	14	16	20	21
Area	109.0000	112.0000	131.0000	145.0000	112.0000	121.0000	110.0000	140.0000	145	112	108
Centroid (r)	37.5688	38.875	40.458	41.7448	51.0893	52.2727	54.5	56.3571	57.3862	76.6339	75.4815
Centroid (c)	90.6147	75.875	56.1985	41.3724	126.5714	111.9835	84.3545	57.4286	42.6966	78.9732	93.7407
Second-row moment	11.3278	12.6808	11.4238	12.4245	9.5992	10.1983	11.7045	11.9153	12.4026	12.8749	11.2682
Second-column moment	7.0992	6.6987	9.9759	11.2958	8.8878	9.57	7.0107	11.0306	11.439	6.4904	6.9328
Second-mixed moment	0.8889	0.6897	-0.0527	0.3709	0.1276	-0.4583	0.9045	-0.0316	-0.2	-0.1169	0.5137
Maximum Inertia (deg)	11.4015	6.4926	-2.0831	16.6559	9.8635	-27.7838	10.5387	-2.0452	-11.2742	-1.049	6.6662
Minimum Inertia (deg)	-78.5985	-83.5074	-92.0831	-73.3441	-80.1365	-117.7838	-79.4613	-92.0452	-101.2742	-91.049	-83.3338

	Blobs (there are 20)								
Label	24	25	29	31	34	36	37	39	41
Area	144	148	135	129	123	117	116	134	136
Centroid (r)	78.1667	79.3784	80.6148	81.8682	89.0244	90	92.3276	95.2836	94.3529
Centroid (c)	59.1528	44.3311	26.5704	11.7132	129.6179	115	87.6552	45.7463	60.5
Second-row moment	11.7222	12.7622	10.859	10.8276	9.6986	9.4872	12.0996	11.9345	10.9637
Second-column moment	11.9905	11.3296	11.5784	10.8092	10.415	9.4872	7.3639	10.1297	10.9853
Second-mixed moment	0.2245	-0.1523	1.2049	-0.2238	0.05	0	0.5699	0.3481	0
Maximum Inertia (deg)	-29.5707	-6.0015	-36.6895	-43.824	-3.971	0	6.7658	10.546	0
Minimum Inertia (deg)	-119.5707	-96.0015	-126.6895	-133.824	-93.971	-90	-83.2342	-79.454	-90

Name: Abhiram Durgaraju
CSE 803 Computer Vision Homework 2

Below is my proof from MATLAB for image 3A:

```
>> visionhw2_q1_8con
>> output_matrix

output_matrix =

  Columns 1 through 15

    1.0000    2.0000    5.0000    6.0000    9.0000   10.0000   12.0000   14.0000   16.0000   20.0000   21.0000   24.0000   25.0000   29.0000   31.0000
  109.0000  112.0000  131.0000  145.0000  112.0000  121.0000  110.0000  140.0000  145.0000  112.0000  108.0000  144.0000  148.0000  135.0000  129.0000
   37.5688   38.8750   40.4580   41.7448   51.0893   52.2727   54.5000   56.3571   57.3862   76.6339   75.4815   78.1667   79.3784   80.6148   81.8682
   90.6147   75.8750   56.1985   41.3724  126.5714  111.9835   84.3545   57.4286   42.6966   78.9732   93.7407   59.1528   44.3311   26.5704   11.7132
   11.3278   12.6808   11.4238   12.4245    9.5992   10.1983   11.7045   11.9153   12.4026   12.8749   11.2682   11.7222   12.7622   10.8590   10.8276
    7.0992    6.6987    9.9759   11.2958    8.8878    9.5700    7.0107   11.0306   11.4390    6.4904    6.9328   11.9905   11.3296   11.5784   10.8092
    0.8889    0.6897   -0.0527    0.3709    0.1276   -0.4583    0.9045   -0.0316   -0.2000   -0.1169    0.5137    0.2245   -0.1523    1.2049   -0.2238
   11.4015    6.4926   -2.0831   16.6559    9.8635  -27.7838   10.5387   -2.0452  -11.2742   -1.0490    6.6662  -29.5707   -6.0015  -36.6895  -43.8240
  -78.5985  -83.5074  -92.0831  -73.3441  -80.1365 -117.7838  -79.4613  -92.0452 -101.2742  -91.0490  -83.3338 -119.5707  -96.0015 -126.6895 -133.8240

  Columns 16 through 20

   34.0000   36.0000   37.0000   39.0000   41.0000
  123.0000  117.0000  116.0000  134.0000  136.0000
   89.0244   90.0000   92.3276   95.2836   94.3529
  129.6179  115.0000   87.6552   45.7463   60.5000
    9.6986    9.4872   12.0996   11.9345   10.9637
   10.4150    9.4872    7.3639   10.1297   10.9853
    0.0500         0    0.5699    0.3481         0
   -3.9710         0    6.7658   10.5460         0
  -93.9710  -90.0000  -83.2342  -79.4540  -90.0000
```

Below is results for my own chosen image ‘shapes.jpg’:

	Blobs (there are 6)					
Label	1	2	3	4	5	7
Area	2067	865	859	2352	2016	1430
Centroid (r)	35	42.5	42.5	124.7	124.5	125.0
Centroid (c)	40	124.9	198.4	39.9	124.5	211.1
Second-row moment	126.7	72.3	71.7	190.5	173.3	81.3
Second-column moment	234.0	96.4	127.1	184.8	173.3	170.8
Second-mixed moment	0	0.1	47.7	0.2	0	11.5
Maximum Inertia (deg)	0	-0.2	-29.9	2.4	0	-7.2
Minimum Inertia (deg)	-90.0	-90.2	-119.9	-87.6	-90.0	-97.2

Below is my proof from MATLAB for image 'shapes.jpg':

```
>> visionhw2_q1_8con
>> output_matrix

output_matrix =

   1.0e+03 *

    0.0010    0.0020    0.0030    0.0040    0.0050    0.0070
    2.0670    0.8650    0.8590    2.3520    2.0160    1.4300
    0.0350    0.0425    0.0425    0.1247    0.1245    0.1250
    0.0400    0.1249    0.1984    0.0399    0.1245    0.2111
    0.1267    0.0723    0.0717    0.1905    0.1733    0.0813
    0.2340    0.0964    0.1271    0.1848    0.1733    0.1708
         0    0.0001    0.0477    0.0002         0    0.0115
         0   -0.0002   -0.0299    0.0024         0   -0.0072
   -0.0900   -0.0902   -0.1199   -0.0876   -0.0900   -0.0972
```

5. For this, I choose blob #2, blob #3, and blob #4 from 'hw2-2B.jpg'.

This is blob #2 before any changes:

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Leftmost 1 has a (r,c) of (5,4). We know from the first table in this report that the centroid of blob#2 is (6.5,5). Therefore, we can first calculate the mean radial distance according to the textbook (perimeter pixels are shown in green in above table):

$$\mu_R = \frac{1}{K}\sum_{k=0}^{K-1} \|(r_k, c_k) - (\bar{r}, \bar{c})\|$$

Distance is calculated by:

$$\sqrt{((r_1 - r_2)^2 + (c_1 - c_2)^2)}$$

Mean radial distance = (Sqrt((5-6.5)^2 + (4-5)^2) + Sqrt((5-6.5)^2 + (5-5)^2) + ….)/10 =

(2(1.5) + 4(sqrt(3.25)) + 4(sqrt(1.25)))/10 = 14.683/10 = 1.468

Standard deviation of the radial distance is calculated according to the textbook:

$$\sigma_R = \left(\frac{1}{K}\sum_{k=0}^{K-1}\left[\|(r_k, c_k) - (\bar{r}, \bar{c})\| - \mu_R\right]^2\right)^{1/2}$$

= Sqrt((Sqrt((5-6.5)^2 + (4-5)^2) – 1.468)^2 + (Sqrt((5-6.5)^2 + (5-5)^2) – 1.468)^2 + ….)/10)

= Sqrt((2(0.00102) + 4(0.11202) + 4(0.12247))/10) = Sqrt(0.94/10) = 0.3065

Circularity is therefore defined in the textbook by:

$$C_2 = \frac{\mu_R}{\sigma_R} \qquad (3.12)$$

Therefore, **blob #2 circularity = 1.468/0.3065 = 4.788**

For blob#3, here is the unchanged image:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Top-right labeled pixel has a (r,c) of (6,11). We know from the first table of this report that blob#3 has a centroid of (9.5, 7.5). Therefore, mean radial distance is:

(Sqrt((6-9.5)^2 + (11-7.5)^2) + Sqrt((7-9.5)^2 + (10-7.5)^2) + Sqrt((8-9.5)^2 + (9-7.5)^2) ….)/8 =

(2*4.95 + 2*3.535 + 2*2.123 + 2*0.707)/8) = 22.63/8 = 2.828

The standard deviation of mean radial distance is therefore:

= Sqrt((Sqrt((6-9.5)^2 + (11-7.5)^2) – 2.828)^2 + (Sqrt((7-9.5)^2 + (10-7.5)^2) – 2.828)^2 + ….)/8)

=sqrt((4*4.5 + 4*0.5)/8) = sqrt(2.5) = 1.581

Therefore, **blob #3 circularity = 2.828/1.581= 1.788**

For, blob#4, here is the unchanged image:

0	0	0	0	0	0	0	0	0	0	0	0	0
0	4	4	4	4	4	4	4	4	4	4	4	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Top left labeled pixel has a (r,c) of (18,2). We know from the first table of this report that blob #4 has a centroid of (18, 7). Therefore, mean radial distance is:

(Sqrt((18-18)^2 + (2-7)^2) + Sqrt((18-18)^2 + (3-7)^2) + Sqrt((18-18)^2 + (4-7)^2) ….)/11 =

(5+ 4+ 3 + 2 + 1 + 0 + 1 + 2 +3 +4 +5)/11 = 30/11 = 2.727

The standard deviation of mean radial distance is therefore:

= Sqrt((Sqrt((18-18)^2 + (2-7)^2) – 2.727)^2 + (Sqrt((18-18)^2 + (3-7)^2) – 2.727)^2+ ….)/8)

= Sqrt((2*(5.16 + 1.619 + 0.074 + 0.528 + 2.983)+7.438)/11) = sqrt(28.166/11) = 1.600

Therefore, **blob #4 circularity = 2.727/1.600 = 1.704**

6. For my two blobs, I chose blob #2 and blob #3 from 'hw2-2B.jpg'. Since we had to use morphological operators, I chose to use dilation to make the object larger. Then, I subtracted the original binary image from this new image to get the boundary pixels. I then estimated the circumference of the blobs according to the textbook. For every horizontal and vertical pair of boundary pixels, I added 1 to the circumference. For every pair of diagonal pixels without any vertical or horizontal pair of pixels, I added 1.4 or √2 to the circumference

This is blob #2 before any changes:

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Name: Abhiram Durgaraju

This is blob #2 after dilation with structural element

1
1
1

0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

These are the boundary pixels after the original image is subtracted from the dilated one:

0	1	1	1	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	1	1	1	0

For blob #2, as seen above, there are 4 ordered pairs of vertical and horizontal pixels, therefore, the circumference is 4. This underestimates the true circumference because the chosen blob is so small, the effect of dilation using our structural element is very minimal. If we just use the perimeter pixels, without morphological operators for this blob, the circumference is 10. I chose this blob to illustrate this pitfall.

This is blob #3 before any changes:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Name: Abhiram Durgaraju

This is blob #3 after dilation with structural element

1
1
1

0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	1	1	1	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0

This is blob #3 after subtracting the original image from the dilated one:

0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1	0	0
0	0	0	0	1	0	1	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	1	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0

As seen above, there are 14 ordered diagonal pairs after subtraction. Therefore, the circumference is $14\sqrt{2}$ or 19.79. Of course, this vastly overestimate the circumference with our chosen structural element. The actual circumference, as we can tell from the original image, has 7 diagonal pairs, and the circumference is therefore $7\sqrt{2}$ or 9.89.

Problem 2:

Below is the connected component of image ‘hw2-2A.jpg’, generated by my program in MATLAB:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0
0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0
0	0	1	0	0	2	0	3	0	0	0	0	4	4	4	4	0	2	0	0
0	0	1	0	0	2	0	0	3	0	0	0	4	4	4	4	0	2	0	0
0	0	1	0	0	2	0	0	0	3	0	0	4	4	4	4	0	2	0	0
0	0	1	0	0	2	0	0	0	0	3	0	0	0	0	0	0	2	0	0
0	0	1	0	0	2	0	0	0	0	0	3	0	0	0	0	0	2	0	0
0	0	1	0	0	2	0	0	0	0	0	0	3	0	0	0	0	2	0	0
0	0	1	0	0	2	0	0	0	0	0	0	0	3	0	0	0	2	0	0
0	0	1	0	0	2	0	0	0	0	0	0	0	0	3	0	0	2	0	0
0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	2	0	0
0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

It’s data is shown below, in the same format as previously:

1.0000	2.0000	3.0000	4.0000
11.0000	46.0000	8.0000	12.0000
7.0000	7.5000	7.5000	5.0000
3.0000	12.0000	11.5000	14.5000
10.0000	20.6848	5.2500	0.6667
0	23.5652	5.2500	1.2500
0	0	5.2500	0
-90.0000	0	45.0000	0
0	-90.0000	-45.0000	-90.0000

Below is the connected component of image ‘hw2-2B.jpg’, generated by my program in MATLAB:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	2	2	2	0	0	0	0	0	0	1	0	0
0	1	0	2	2	2	0	0	0	0	3	0	1	0	0
0	1	0	2	2	2	0	0	0	3	0	0	1	0	0
0	1	0	2	2	2	0	0	3	0	0	0	1	0	0
0	1	0	0	0	0	0	3	0	0	0	0	1	0	0
0	1	0	0	0	0	3	0	0	0	0	0	1	0	0
0	1	0	0	0	3	0	0	0	0	0	0	1	0	0
0	1	0	0	3	0	0	0	0	0	0	0	1	0	0
0	1	0	3	0	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	4	4	4	4	4	4	4	4	4	4	4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

As see above there is a correspondence between labels [1,2,3,4] in 2A to labels [4,1,3,2]. I picked 2 of these blobs which correspond and did calculated rotation based on this using MATLAB. In image 2A, I picked, blobs #4 and #1. In image 2B therefore, I picked blobs #2 and #4 respectively. From question 1, we know the centroids are the following:

Image2B	Centroid(r)	Centroid(c)
Blob#2 (lets call J)	6.5000	5.0000
Blob#1 (lets call K)	9.0000	7.5000

Image2A	Centroid(r)	Centroid(c)
Blob#4 (lets call L)	5.0000	14.5000
Blob#2 (lets call M)	7.5000	12.0000

The direction of vector from J to K is given by arctan((7.5-5)/(9-6.5)) = 0.7853 radians or 45. The direction of vector from L to M is given by arctan((12-14.5)/(7.5-5)) = -0.7853 radians or -45 degrees. The rotation is thus (-0.7853 – 0.7853)) = -1.5707 radians or **-90 degrees** counterclockwise. This is the expected rotation, with no error

The translation that maps to these centroids can be calculated for J to L using a matrix where θ is 90 deg:

$$\begin{bmatrix} 5 \\ 14.5 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & u_0 \\ \sin\theta & \cos\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6.5 \\ 5 \\ 1 \end{bmatrix}$$

Solving these systems of equations using a simple calculator, I got the following transformation:

$$u_0 = -6.5\cos(-90) + 5\sin(-90) + 5 = \mathbf{0}$$

$$v_0 = 14.5 - 6.5\sin(-90) - 5\cos(-90) = \mathbf{21}$$

The translation that maps to these centroids can be calculated for K to M using a matrix:

$$\begin{bmatrix} 7.5 \\ 12 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & u_0 \\ \sin\theta & \cos\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 \\ 7.5 \\ 1 \end{bmatrix}$$

Solving these systems of equation using a simple calculator, I got the same transformation:

$$u_0 = 0$$

$$v_0 = 21$$

Either set of transformation values yield same results, and are used to project the 4 blobs for this problem. Below is a centroid mapping for image 2A from 2B:

Image 2B	
	Centroid (r, c)
Blob#1	(9, 7.5)
Blob#2	(6.5, 5)
Blob#3	(9.5, 7.5)
Blob#4	(18, 7)

Image 2A	
	Centroid(r,c)
Blob#2	(7.5, 12)
Blob#4	(5, 14.5)
Blob#3	(7.5, 11.5)
Blob#1	(7, 3)

Name: Abhiram Durgaraju
CSE 803 Computer Vision Homework 2

Mapped Blob #2 in 2A -> $\begin{bmatrix}\mathbf{7.5}\\ \mathbf{12}\\ \mathbf{1}\end{bmatrix} = \begin{bmatrix}\cos(-90) & -\sin(-90) & 0\\ \sin(-90) & \cos(-90) & 21\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}9\\ 7.5\\ 1\end{bmatrix}$ <- Blob#1 in 2B

Mapped Blob #4 in 2A -> $\begin{bmatrix}\mathbf{5}\\ \mathbf{14.5}\\ \mathbf{1}\end{bmatrix} = \begin{bmatrix}\cos(-90) & -\sin(-90) & 0\\ \sin(-90) & \cos(-90) & 21\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}6.5\\ 5\\ 1\end{bmatrix}$ <- Blob#2 in 2B

Mapped Blob #3 in 2A -> $\begin{bmatrix}\mathbf{7.5}\\ \mathbf{11.5}\\ \mathbf{1}\end{bmatrix} = \begin{bmatrix}\cos(-90) & -\sin(-90) & 0\\ \sin(-90) & \cos(-90) & 21\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}9.5\\ 7.5\\ 1\end{bmatrix}$ <- Blob#3 in 2B

Mapped Blob #1 in 2A -> $\begin{bmatrix}\mathbf{7}\\ \mathbf{3}\\ \mathbf{1}\end{bmatrix} = \begin{bmatrix}\cos(-90) & -\sin(-90) & 0\\ \sin(-90) & \cos(-90) & 21\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}18\\ 7\\ 1\end{bmatrix}$ <- Blob#4 in 2B

As seen on this page, the mapped blobs have identical centroid values to the original ones from image 2A.

Here is a summary of the mapped centroids, and the error associated with each. The error was calculated using Euclidean distance with formula given below:

$$\sqrt{((r_1 - r_2)^2 + (c_1 - c_2)^2)}$$

Since our mapped blobs were identical to the centroids of the original image 2A, the error is 0 for all 4 blobs. The results are as expected, given out accurate rotation of -90 degrees counterclockwise, our translation too will be accurate given the objects are identical.

Mapped Centroids from Image 2B to 2A using $u_0 = 0$ and $v_0 = 21$		
	Centroid(r,c)	Error (pixel units)
Mapped Blob#2	(9, 17)	0
Mapped Blob#4	(5, 14.5)	0
Mapped Blob#3	(2.5, 17.5)	0
Mapped Blob#1	(3, 26)	0