

Problem Set 7

CMSC 426

Assigned Tuesday, November 22, Due Thursday, December 8

1. 2D Motion with a Matrix

- a. **6 points** For each of the following 3x3 matrices, indicate whether it represents a 2D rotation. Explain your answer.

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{4} & 0 \\ -1 & \frac{1}{1} & 0 \\ \frac{4}{4} & \frac{2}{2} & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{1}{2} & 0 \\ -\sqrt{2} & \sqrt{3} & 0 \\ \frac{2}{2} & \frac{2}{2} & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\sqrt{3}}{3} & -\frac{\sqrt{6}}{3} & 0 \\ \frac{\sqrt{6}}{3} & \frac{\sqrt{3}}{3} & 0 \\ \frac{3}{3} & \frac{3}{3} & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

- b. **5 points** Give a matrix that will represent a 45 degree clockwise rotation of 2D points.
- c. **5 points** Give a matrix that will rotate points clockwise by 90 degrees about the point (1,2). That is, the effect should be as if you held the point (1,2) fixed and rotated everything about that point by 90 degrees.

2. 3D Motion with a Matrix

- a. **6 points** For each of the following 4x4 matrices, indicate whether it represents a 3D rotation. Explain your answer.

$$\begin{pmatrix} 0 & \frac{2\sqrt{2}}{3} & -\frac{1}{3} & 0 \\ -1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{2\sqrt{2}}{3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -\frac{1}{2} & \frac{1}{3} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{2}{3} & -\frac{1}{2} & 0 \\ \frac{2}{2} & \frac{3}{3} & -\frac{2}{2} & 0 \\ \frac{2}{0} & -\frac{2}{3} & -\frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- b. **6 points** Give matrices that will have the following effects:
- Translate 3D points in the direction $(2, -1, 3)$.
 - Translate 3D points in the direction $(1, 0, -1)$ and then scale them by a factor of 2.
 - Scale the points by a factor of 2 and then translate them in the direction $(1, 0, -1)$.

- c. **5 points** Give a rotation matrix that will rotate the y axis so that it points in the same direction as $(1, 1, 1)$, while rotating the x axis so that it points in the direction $(-1, 0, 1)$.
- d. **5 points** Explain why it is not possible to rotate the y axis so that it points in the same direction as $(1, 1, 1)$, while rotating the x axis so that it points in the direction $(-1, 1, 1)$.

3. Structure-from-Motion

- a. **12 points** Suppose we take two images of a scene. In the first image, we find four points with coordinates:

$$p_1^1 = (3,6) \quad p_2^1 = (5,7) \quad p_3^1 = (5,4) \quad p_4^1 = (4,8)$$

In the second image, we find the same points, now with coordinates:

$$p_1^2 = (2,4) \quad p_2^2 = (5,7) \quad p_3^2 = (2,4) \quad p_4^2 = (5,1)$$

We use the superscript to indicate the image number, and the subscript to indicate the point number. Suppose we know that the 3D coordinates of the four points are: $P_1=(0, 0, 0)$, $P_2=(1, 0, 0)$, $P_3=(0, 1, 0)$, $P_4=(0, 0, 1)$. Determine what the camera motion was for these two images. You can describe each camera motion using a 2×4 matrix. This matrix will combine together rotation, translation, scaling, and projection to 2D.

- b. **Challenge problem (5 points):** Do the matrices that you found really represent a rigid 3D motion? Explain how you would determine this.
- c. **5 points** Suppose there is a fifth scene point, which produces the image coordinates:

$$p_5^1 = (9,3) \quad p_5^2 = (5,7)$$

Find the 3D coordinates of this scene point.

4. **45 points.** Corner Detection. You should implement the corner detector that we discussed in class. Recall that to do this we perform the following steps. First, for every point in the image, we form a matrix, C , which is based on the image gradients in the neighborhood of this point. You should compute image gradients in the same way that you did for edge detection, smoothing the image first. I suggest that you reuse the routines *image_gradient* and *smooth_image* from the edge detection problem set. Use a 5×5 neighborhood around each point to form C . Then use singular value decomposition, with the matlab routine *svd*, to find the singular values of C . This is summarized in the equations:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \stackrel{\text{Using SVD}}{=} R_2 \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R_1$$

The measure of cornerness for a point will be the smallest of the two singular values, λ_2 . Since you have a measure of cornerness for every pixel in the image, you should output a matrix that is the same size as the input image, and that contains the cornerness for each pixel. So your complete program will have the form: $M = \text{image_cornerness}(I, \text{sigma})$. Here I is the input image, and sigma is a value indicating how much to smooth the image before computing the image gradient. I have given you a function `display_best_corners`, which will take as input an image, and the output of your cornerness routine, and will output an image that shows where the corners are. When this routine finds a good corner, it eliminates from consideration nearby points that are slightly less good, so it doesn't exactly find the locations in C that have the highest values.

We can use this, on the swan image for example, with the calls:

```
C = image_cornerness(swanbw, 2);
imshow(display_best_corners(swanbw, C, 20));
```

This will display the 20 best corners in the swan image. A link to a picture of the result is in `swancorners.jpg`.

Let's step through a simpler example of how this how program will work. Suppose we have an image with a corner:

```
I =
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  10 10 10 10
  0  0  0  10 10 10 10
  0  0  0  10 10 10 10
  0  0  0  10 10 10 10
```

To keep things simpler, I'll do no smoothing (so I call my routine with `sigma = 0`). Computing the x and y components of the image gradient, I get:

```
Ix =
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  5  5  0  0  0
  0  0  5  5  0  0  0
  0  0  5  5  0  0  0
  0  0  5  5  0  0  0
```

Iy =

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	5	5	5	5
0	0	0	5	5	5	5
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Using these to build a C matrix for the point (4,4), we get the matrix:

C =

150	25
25	150

Why these numbers? Below, I show all the gradients, as pairs (I_x,I_y). In red are the gradients in a 5x5 neighborhood around the pixel at (4,4). If we square all the I_x values we get 150, since there are six such values. Similarly for squaring I_y. Only pixel (4,4) gives a non-zero value, 25, for I_x*I_y though.

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	0,5	0,5	0,5	0,5
0,0	0,0	5,0	5,5	0,5	0,5	0,5
0,0	0,0	5,0	5,0	0,0	0,0	0,0
0,0	0,0	5,0	5,0	0,0	0,0	0,0
0,0	0,0	5,0	5,0	0,0	0,0	0,0

We will have to do this for every point. Performing svd on the matrix C, we get:

svd(C)

ans =

175.0000
125.0000

So the cornerness of this point is 125, which is pretty high. Looking at the cornerness of all the points, we get:

0	0	0	0	0	0	0	
0	25.0000	39.6447	44.0983	45.9431	21.4986		0
0	39.6447	75.0000	89.6447	94.0983	45.9431		0
0	44.0983	89.6447	125.0000	139.6447	70.1854		0
0	45.9431	94.0983	139.6447	175.0000	94.0983		0
0	21.4986	45.9431	70.1854	94.0983	75.0000		0
0	0	0	0	0	0	0	

Notice that the point with the highest cornerness is located at (5,5), even though intuitively this does not seem to be exactly where the corner is. This is because a 5x5 region centered at (5,5) has the greatest diversity of gradients, since it has all the non-zero gradients we got in the neighborhood around (4,4), plus some new ones.