Homework 1 solution template
CMPSCI 370 Spring 2019, UMass Amherst
Name: Subhransu Maji

1 Matrix manipulation

1. Create a $m \times n$ array of all zeros.

```
matlab: zeros(m, n)
numpy: np.zeros((m, n))
```

2. Create a random $m \times n$ array.

```
matlab: rand(m, n)
numpy: np.random.rand(m, n)
```

3. Code to compute its length (or norm).

```
matlab: sqrt(sum(v.^2))
numpy: np.sqrt(np.sum(a*a))
```

4. Given variables u and v representing arrays of size $n \times 1$, write down code to compute their

   (a) dot product

   ```
   matlab: u'*v
   numpy: np.sum(u * v)
   ```

   (b) angle

   ```
   matlab: acos(u'*v/(norm(u)*norm(v)))
   numpy: np.arccos(np.sum(u*v)/(np.linalg.norm(u)*np.linalg.norm(v)))
   ```

   (c) distance

```
matlab: sqrt((u - v)'* (u - v))
numpy: np.sqrt(np.sum((u-v)*(u-v)))
```

5. Given an array $a \in \mathbb{R}^{m \times n}$ write code to reshape it to a vector of size $nm \times 1$.

```
matlab: reshape(a, [], 1) or a(:)
numpy: np.reshape(a, (-1, 1))
```

2 Image formation

1. Illustration of the object and the image formed in the pinhole camera.

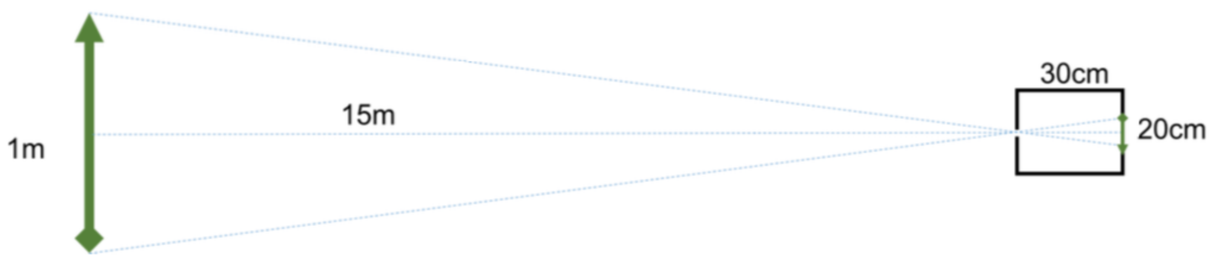


Figure 1: Pinhole camera.

2. Calculations for the size of the object.

$$\frac{1\text{m}}{15\text{m}} = \frac{x}{30\text{cm}}$$
$$x = 2\text{cm}$$

3. Calculations for the distance.

$$\frac{1\text{m}}{d} = \frac{20\text{cm}}{30\text{cm}}$$
$$d = 1.5\text{m}$$

4. Time taken for the camera with a lens.

$$10 \times \pi \times 1^2 = t \times \pi \times 10^2$$
$$t = 0.1 \text{ milliseconds}$$

3 Aligning Prokudin-Gorskii images

1. Outputs of `evalAlignment` on the toy images. (The answer is okay as long as the prediction is exactly the inverse of ground truth)

```
Evaluating alignment ..
 1 balloon.jpeg
   gt shift: (-5, 3) (10, 2)
 pred shift: ( 5,-3) (-10,-2)
 2 cat.jpg
   gt shift: (13, 8) (-7, 8)
 pred shift: (-13,-8) ( 7,-8)
 3 ip.jpg
   gt shift: (-4,-13) ( 2,-14)
 pred shift: ( 4,13) (-2,14)
 4 puppy.jpg
   gt shift: ( 1,13) ( 9,-11)
 pred shift: (-1,-13) (-9,11)
 5 squirrel.jpg
   gt shift: ( 2,-15) (-1,-5)
 pred shift: (-2,15) ( 1, 5)
 6 pencils.jpg
   gt shift: (-10,-6) ( 9, 1)
 pred shift: (10, 6) (-9,-1)
 7 house.png
   gt shift: (-10,-7) ( 3, 5)
 pred shift: (10, 7) (-3,-5)
 8 light.png
   gt shift: ( 6,-2) ( 8,-13)
 pred shift: (-6, 2) (-8,13)
 9 sails.png
   gt shift: (-8,-11) (13,10)
 pred shift: ( 8,11) (-13,-10)
10 tree.jpeg
   gt shift: ( 1,-13) (15,-2)
 pred shift: (-1,13) (-15, 2)
```

2. Output of `alignChannels.m` that shows the computed shifts as seen below.

```
1 00125v.jpg shift: G (-4, 1) B (-10, 2)
2 00153v.jpg shift: G (-8,-2) B (-15,-3)
3 00398v.jpg shift: G (-6, 1) B (-11, -4)
4 00149v.jpg shift: G (-5, 0) B (-9,-1)
5 00351v.jpg shift: G (-9, 0) B (-13, 1)
6 01112v.jpg shift: G (-6,-1) B (-7,-2)
```

3. A figure that shows all the aligned color images. Only include the images from the Prokudin-Gorskii dataset in the original resolution. For example, do not take low-resolution screenshots of the outputs; Instead save them using appropriate commands in Matlab and Python.

Figure 2: Aligned color images.

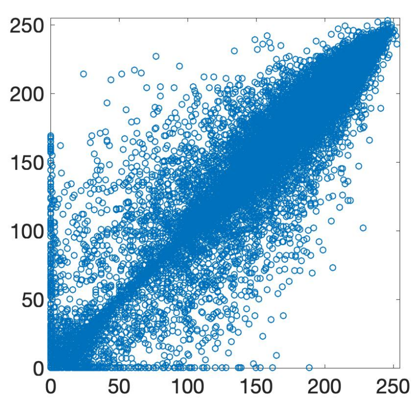
4 Color image demosaicing

1. Errors of the nearest neighbor interpolation algorithm.

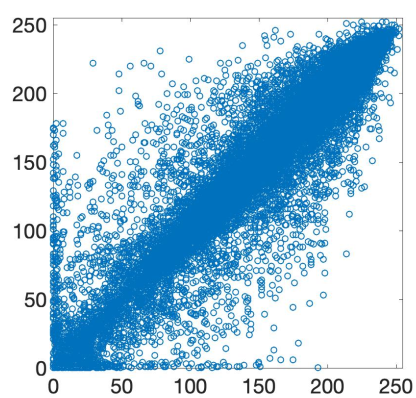
```
>> evalDemosaicing
--------------------------------------------------
#   image           baseline       nn
--------------------------------------------------
1   balloon.jpeg   0.179239   0.018572
2   cat.jpg        0.099966   0.021616
3   ip.jpg         0.231587    0.023371
4   puppy.jpg       0.094093    0.013670
5   squirrel.jpg    0.121964    0.037806
6   pencils.jpg     0.183101    0.019147
7   house.png       0.117667    0.026267
8   light.png       0.097868    0.026179
9   sails.png      0.074946   0.020231
10   tree.jpeg      0.167812    0.024825
--------------------------------------------------
      average       0.136824    0.023168
--------------------------------------------------
```

2. A figure (e.g., Figure 3) that shows the images obtained after interpolation.
3. The three plots for the `puppy.jpg` image.

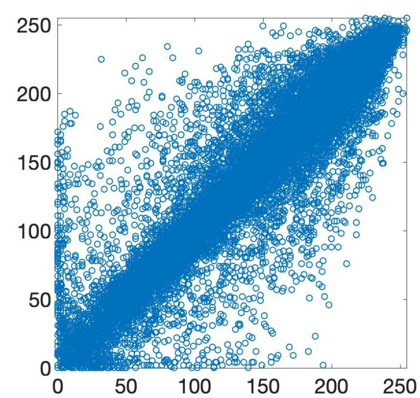
Figure 3: Demosaiced images using nearest neighbor interpolation.



(a) red



(b) green



(c) blue

Figure 4: The value of pixels against the values of their neighbor.