Homework 1 solution template

CMPSCI 370 Spring 2019, UMass Amherst

Name: Subhransu Maji

Here is a template that your solutions should roughly follow. Include outputs as figures, and code should be included in the end.

1 Matrix manipulation

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

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

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

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

   (a) dot product

   (b) angle

(c) distance

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

2 Image formation

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

2. Calculations for the size of the object.

3. Calculations for the distance.

4. Time taken for the camera with a lens.

3 Aligning Prokudin-Gorskii images

1. Outputs of `evalAlignment` on the toy images.
2. Output of `alignChannels.m` that shows the computed shifts as seen below.

```
>> alignProkudin
 1 00125v.jpg shift: G ( 0, 0) B ( 0, 0)
 2 00153v.jpg shift: G ( 0, 0) B ( 0, 0)
 3 00398v.jpg shift: G ( 0, 0) B ( 0, 0)
 4 00149v.jpg shift: G ( 0, 0) B ( 0, 0)
 5 00351v.jpg shift: G ( 0, 0) B ( 0, 0)
 6 01112v.jpg shift: G ( 0, 0) B ( 0, 0)
```

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 1: Aligned color images.

4 Color image demosaicing

1. Errors of the nearest neighbor interpolation algorithm.

```
>> evalDemosaicing
-------------------------------------------
#   image             baseline    nn
-------------------------------------------
1   balloon.jpeg      0.179239   0.179239
2   cat.jpg           0.099966   0.099966
3   ip.jpg            0.231587   0.231587
4   puppy.jpg         0.094093   0.013670
5   pencils.jpg       0.181449   0.181449
6   candy.jpeg        0.206359   0.206359
7   house.png         0.117667   0.117667
8   light.png         0.097868   0.097868
9   sails.png         0.074946   0.074946
10   tree.jpeg        0.167812   0.167812
-------------------------------------------
   average            0.139150   0.139150
-------------------------------------------
```

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

5 Solution code

Include the source code for your solutions as seen below (only the files you implemented are necessary). In latex the command `verbatiminput{alignChannels.m}` allows you to include the code verbatim as seen below. Regardless of how you do this the main requirement is that the included code is readable (use proper formatting, variable names, etc.) A screenshot of your code works too provided you include a link to source files.

5..1 alignChannels.m

```
function [im, predShift] = alignChannels(im, maxShift)
% Implement this
im = im;
predShift = zeros(2,2);
```

5..2 mosaicImage.m

```
function mosim = mosaicImage(im)
% MOSAICIMAGE computes the mosaic of an image.
%   MOSIM = MOSAICIMAGE(IM) computes the response of the image under a
%   Bayer filter. Given an image IM = NxMx3, the output is a NxM image
%   where the R,G,B channels are sampled according to RGRG on the top left.
%
% This code is part of:
%
%   CMPSCI 370: Computer Vision, Fall 2014
%   University of Massachusetts, Amherst
%   Instructor: Subhransu Maji
```

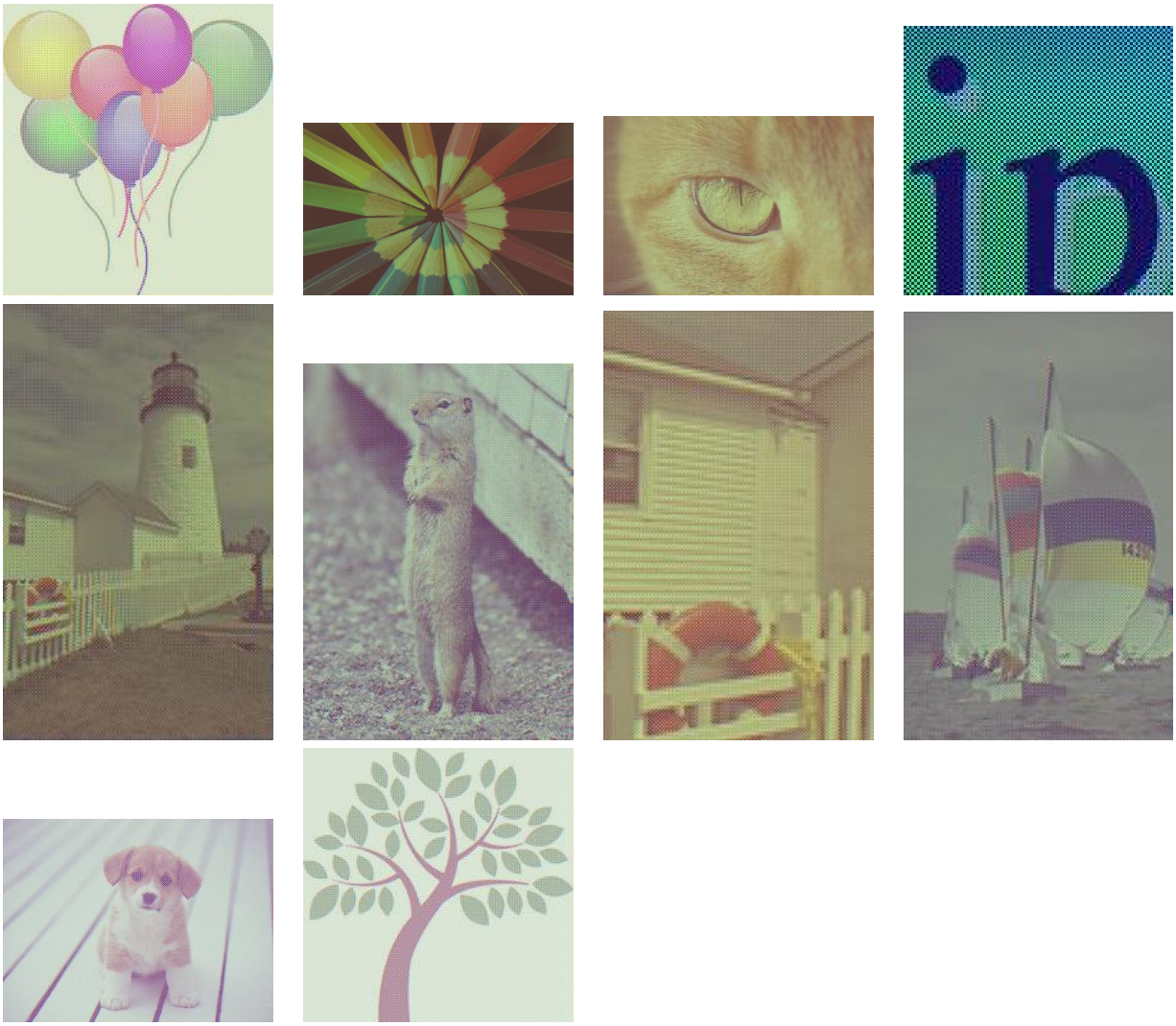
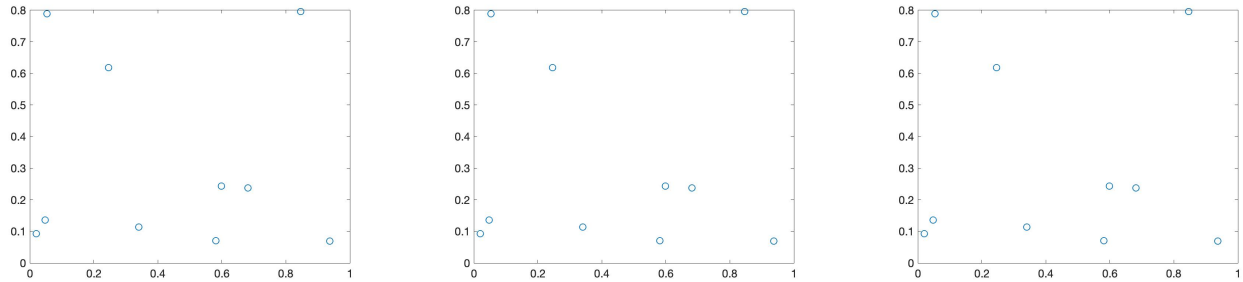


Figure 2: Demosaiced images using nearest neighbor interpolation.



(a) red (b) green (c) blue

Figure 3: The value of pixels against the values of their neighbor.

```
%
%    Homework 1: Color images
```

```
[imageHeight, imageWidth, numChanels] = size(im);
assert(numChanels == 3); % Check that it is a color image
mosim = im;
```

5..3 demosaicImage.m

```
function output = demosaicImage(im, method)
% DEMOSAICIMAGE computes the color image from mosaiced input
%   OUTPUT = DEMOSAICIMAGE(IM, METHOD) computes a demosaiced OUTPUT from
%   the input IM. Various interpolation METHOD can be used such as
%   'baseline', 'nn'
%
% Author: Subhransu Maji
% Copyright 2014

switch lower(method)
    case 'baseline'
        output = demosaicBaseline(im);
    case 'nn'
        output = demosaicNN(im);         % Implement this
end

%--------------------------------------------------------------------------
%                           Baseline demosacing algorithm.
%                           The algorithm replaces missing values with the
%                           mean of each color channel.
%--------------------------------------------------------------------------
function mosim = demosaicBaseline(im)
mosim = repmat(im, [1 1 3]); % Create an image by stacking the input
[imageHeight, imageWidth] = size(im);

% Red channel (odd rows and columns);
redValues = im(1:2:imageHeight, 1:2:imageWidth);
meanValue = mean(mean(redValues));
mosim(:,:,1) = meanValue;
mosim(1:2:imageHeight, 1:2:imageWidth,1) = im(1:2:imageHeight, 1:2:imageWidth);

% Blue channel (even rows and colums);
blueValues = im(2:2:imageHeight, 2:2:imageWidth);
meanValue = mean(mean(blueValues));
mosim(:,:,3) = meanValue;
mosim(2:2:imageHeight, 2:2:imageWidth,3) = im(2:2:imageHeight, 2:2:imageWidth);

% Green channel (remaining places)
% We will first create a mask for the green pixels (+1 green, -1 not green)
mask = ones(imageHeight, imageWidth);
mask(1:2:imageHeight, 1:2:imageWidth) = -1;
mask(2:2:imageHeight, 2:2:imageWidth) = -1;
greenValues = mosim(mask > 0);
meanValue = mean(greenValues);
% For the green pixels we copy the value
greenChannel = im;
greenChannel(mask < 0) = meanValue;
mosim(:,:,2) = greenChannel;
```

```
function mosim = demosaicNN(im)
mosim = demosaicBaseline(im); %replace this with your implementation
```