CS682: Computer Vision (Spring 2018)
Assignment 1

Bhavika Tekwani (btekwani@gmu.edu)

1. I installed OpenCV with the Python 2 version on my computer. I ran into an issue with installation though - if you're using the Anaconda 3 distribution with Ubuntu 14.04, you may not be able to run VideoCapture in opencv-python. I resolved this by switching to Python 2 and building OpenCV from source following the instructions in the documentation for Linux distributions. Basically, pip and conda install commands may not install the version of opencv-python that is compatible with Linux.
2. **Convert to grayscale:**
   I used the *cvtColor* method in OpenCV to change the image I had captured to grayscale.
   The flag used for this is COLOR_RBG2GRAY. As mentioned in the homework specification, I used an image size of 1 MP (1200 pixels in width, 900 in height).
   This grayscale image is saved as *bhavika_grey.png*.
3. **Transformations**: I have performed 5 transformations: averaging, switched colorspaces, perspective transformation, median blurring and blending.

   Averaging: I used a (15, 15) filter and the *blur* method from OpenCV to convert both - the original RGB image and it's grayscale version to a blurred image. The output is shown in averaging_rgb.png and averaging_grey.png.

   Switching colorspace: Since OpenCV provides a number of colorspace flags that can be used to convert the image from one to another, I experimented with a few of them. I finally used one called COLOR_BGR2Luv to give the images a pinkish tint & shading. Similar to when we converted the image to greyscale, I used the *cvtColor* method.

   Perspective transformation: The picture I took of myself is a frontal view. I wanted to transform the perspective to what I could image seeing on a movie screen from a side angle. To do this, I used the perspective transformation. First, I selected points on the original image that I wanted transformed. Then I selected points for the transformed image (4 points to pick a boundary). Using the *getPerspectiveTransform* method in OpenCV, I calculate the 3*3 matrix of a perspective transform and apply it to the image using the *warpPerspective* method.

   Median Blurring: The median blurring filter smoothens the image even more than the averaging filter. I used an aperture size of 15 in the *medianBlur* method from OpenCV. Using median blurring on both RGB and grayscale images, I can see that some portions that might be considered edges seem to look smoothed over and more pronounced.

   Blending: I used the *addWeighted* method from OpenCV to calculate the weighted sum of two arrays. I took another picture of myself and blended the two in a 70%-30% weighted scheme. The result is represented as follows where we decide the value of *alpha* as 0.3.

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$$

4. I have been observing the popularity of Mask R-CNN for object instance segmentation in the deep learning and vision community.
   Original paper: https://arxiv.org/abs/1703.06870
   Original implementation: https://github.com/facebookresearch/Detectron
   Alternative implementation: https://github.com/matterport/Mask_RCNN

   Mask R-CNN is built upon the Feature Pyramid Network (FPN) and ResNet-101. It works by first proposing anchor boxes and then refining bounding boxes to identify the object in a particular box. Then the R-CNN generates masks to fit the shape of the identified object and overlays those masks onto the objects.

   Mask R-CNN is known to achieve state of the art results on the COCO 2016 - Common Objects in Context dataset.

5. Code and output hosted on bhavikatekwani.me/vision.html