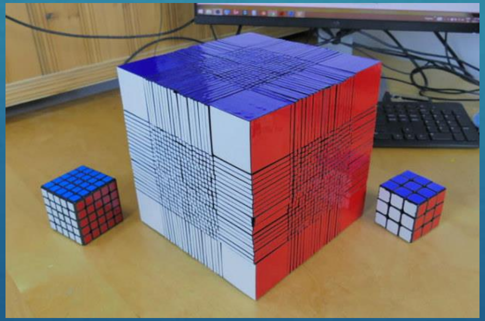
Filters and Transformations: Graphics Programming Assignment 01

AKARSH KUMAR

Images Used

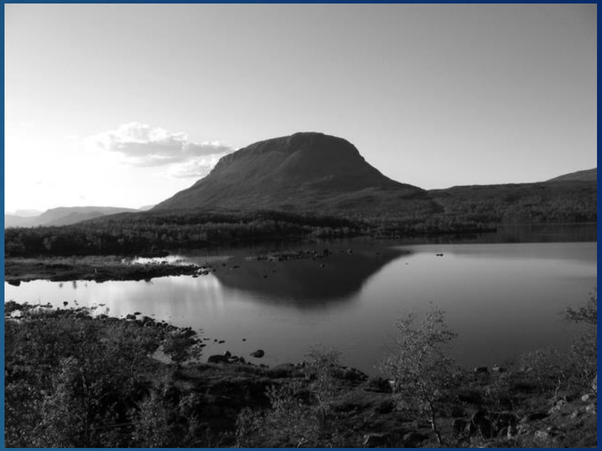
- The following two images were used for altering and transformations:

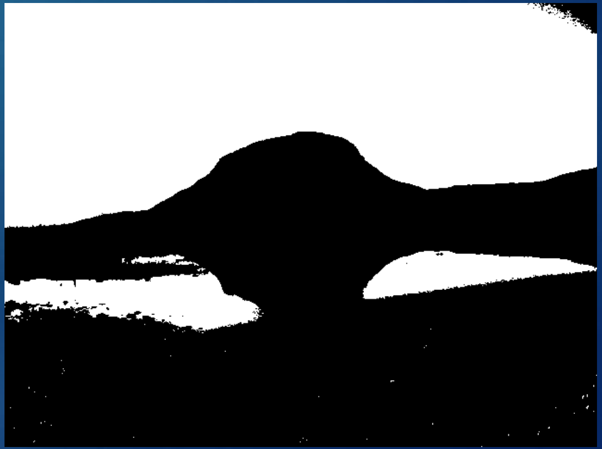
- This image was used for perspective transform:



Greyscale and Black and White

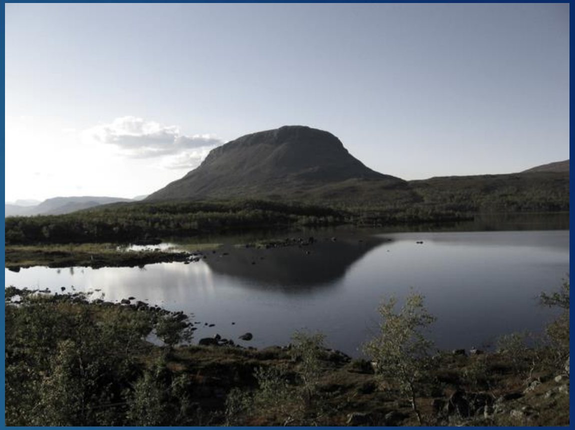
- A function to convert an image to greyscale was made by taking 20% of the blue channel, 70% of the green channel, 10% of the red channel.
- Code:
  - newimg = 0.2*img[:,:,2]+0.7*img[:,:,1]+0.1*img[:,:,0]
- The blackWhite function to convert it to pure black and white established a threshold of 128. Any value of a pixel higher than the threshold was reset to 255 while everything below or equal to the threshold was set to zero.
- Code:
  - newimg[newimg<=128] = 0
  - newimg[newimg>128] = 255

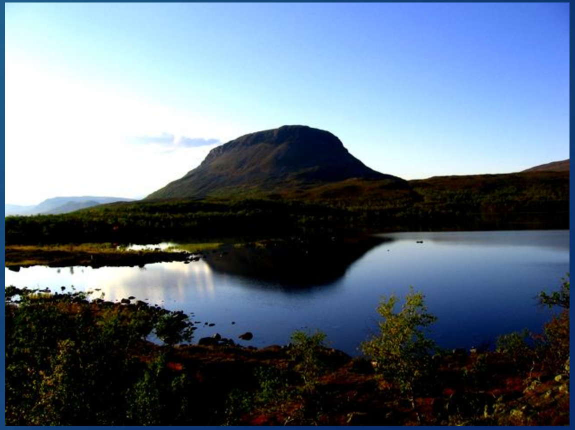




Desaturate and Contrast

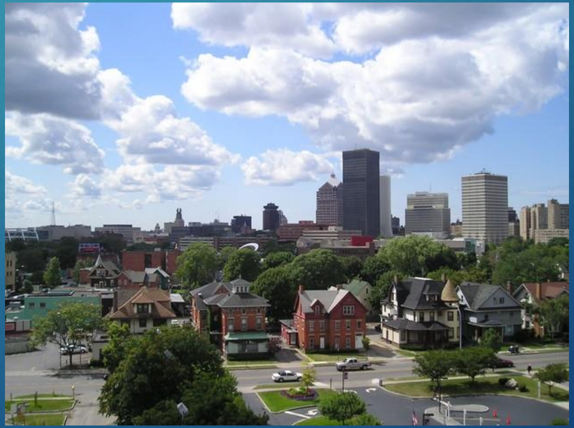
- To desaturate an image, the weighted average of the image's color channels and the greyscale value of them was taken using the percentage of desaturation.
- Code:
  - newimg[:,:,0] = newimg[:,:,0]*(1-percent)+greyimg[:,:]*percent
  - newimg[:,:,1] = newimg[:,:,1]*(1-percent)+greyimg[:,:]*percent
  - newimg[:,:,2] = newimg[:,:,2]*(1-percent)+greyimg[:,:]*percent
- Contrast was done by scaling all color channels of the image by a given factor from the baseline 128 value.
- Code:
  - newimg[:,:,:] = (newimg[:,:,:]-128)*factor+128
  - newimg[newimg>255] = 255
  - newimg[newimg<0] = 0

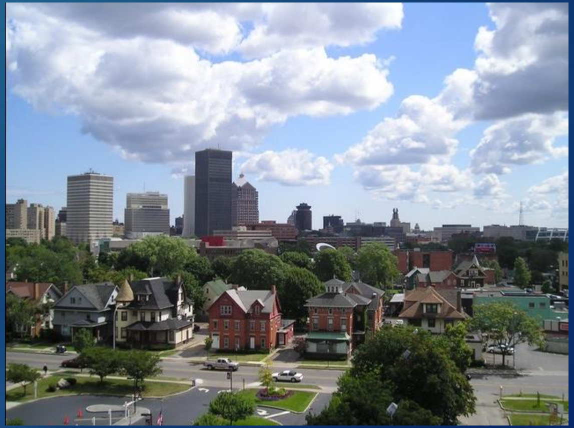




Mirror Transformation Matrix

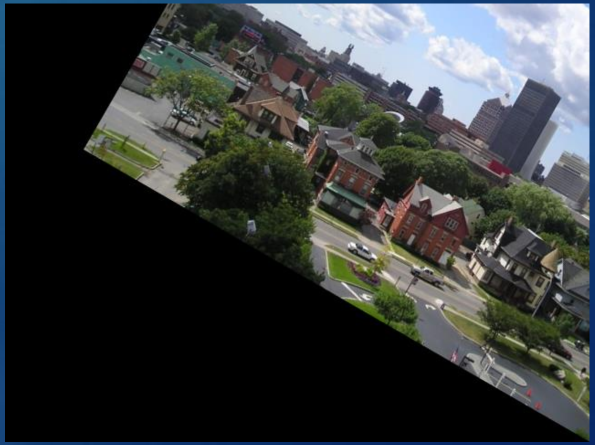
- The following matrix was used to mirror the image across the y-axis:
- $$\begin{matrix} -1 & 0 & img\_width \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$
- The first -1 scaled the x coordinates of all points by -1 and flipped them across the y axis.
- The img_width translated this new flipped image to the appropriate spot to make it look like it was flipped across the center.





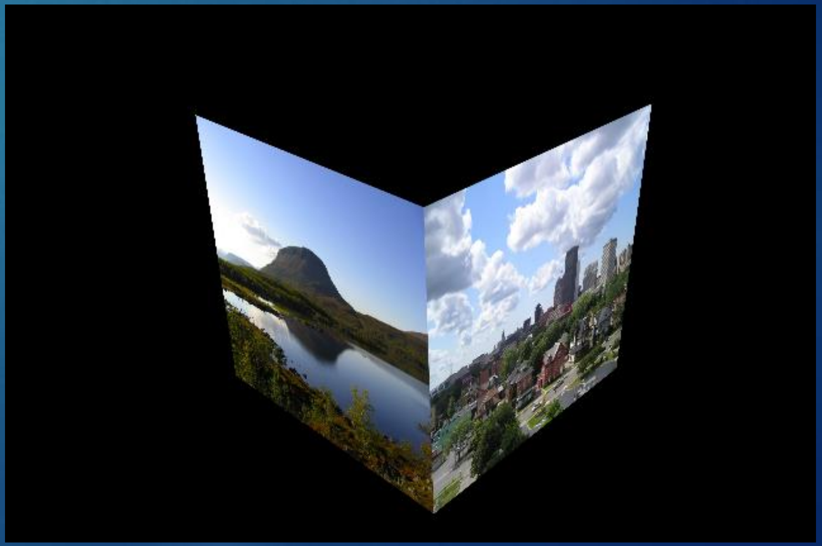
Rotation Transformation Matrix

- To rotate the image about the bottom right corner, a matrix was first created to translate the bottom right corner to the origin of the grid
- Then, it was rotated clockwise using a rotation matrix
- Finally, it was translated back to where the bottom right corner would be where it originally was
- Code:
  - matrix1 = math.array([[1,0,-img_width],[0,1,-img_height],[0,0,1]],math.float32)
  - matrix2 = math.array([[costheta,sintheta,0],[-sintheta,costheta,0],[0,0,1]],math.float32)
  - matrix3 = math.array([[1,0,img_width],[0,1,img_height],[0,0,1]],math.float32)
  - M = matrix3.dot(matrix2).dot(matrix1)
  - img2 = cv2.warpPerspective(img2,M,(x,y))



Perspective Transformation Matrix

- The points of the vertices of the cube in image3 were found using a program online that located pixel coordinates.
- The points of image1 and image2 were then transformed to the cube points using the following code:
  - Mtrans1 = cv2.getPerspectiveTransform(ptsimg1,pts1)
  - Mtrans2 = cv2.getPerspectiveTransform(ptsimg2,pts2)
  - img1trans = cv2.warpPerspective(img1,Mtrans1,(x,y))
  - img2trans = cv2.warpPerspective(img2,Mtrans2,(x,y))



Perspective Transformation Matrix Continued

- The transformed image was then placed on top of image3 using boolean masking built into python
- A mask was created to show where the transformed image is not black, then the mask was used to copy and paste everything from the transformed image into image3 only where the mask validated so
- Code:
  - mask = trans[:,:,:] !=0
  - newimg[mask] = trans[mask]