# Comp Photography Final Project

Deliang Yin

Fall 2017

deliang.yin@gatech.edu

# Image Colorization

**Project Description**: This project is proposed to replicate the colorization algorithm presented in the paper *Colorization using Optimization* by A. Levin, D. Lischinski, and Y. Weiss.

This project is to colorize the black-and-white images marked with color scribbles. In addition, I extended the scope to colorize black-and-white video clips.

## The Goal of Your Project

- Original project scope:

  Given a grayscale image marked with some color scribbles, the presented algorithm is able to colorize the image. The colorized image is then compared to the original color image.

- What motivated you to do this project?

  In my childhood, I saw some funny black-and-white movies, such as the ones performed by Charles Spencer Chaplin. At that time, I had a dream to colorize them using the modern colors, and this left me a room for imagination. After learning computational photography, I am finally equipped necessary skills to fulfill my child dream.

## Scope Changes

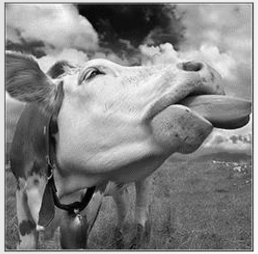
- Did you run into issues that required you to change project scope from your proposal?

  No, I haven't met issues during replication. However, when I finished colorizing still black-and-white images, I still got time remaining before the deadline.  As such, I extended the scope to colorize black-and-white video clips.

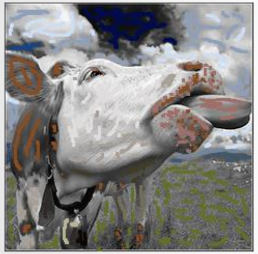
# Showcase

## To colorize still image

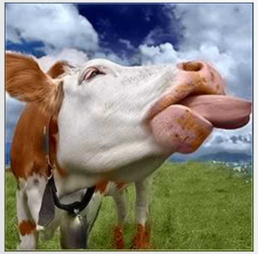
**Input 1**:
black and
white
image



**Input 2**:
marked Image

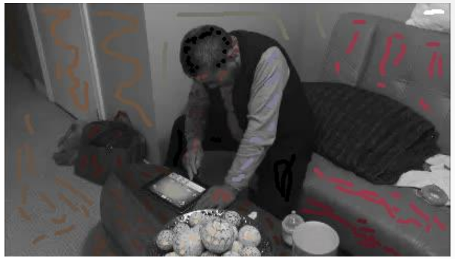


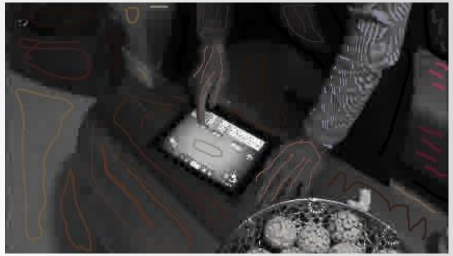
**Output**:
colorized Image



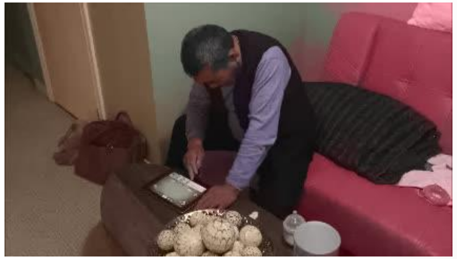
## To colorize black-and-white video clips

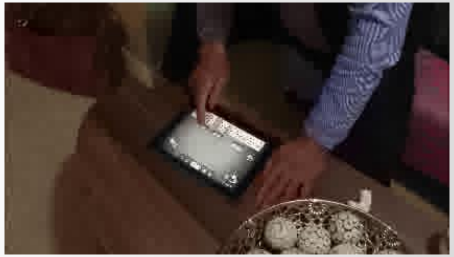
**Input**:
marked image





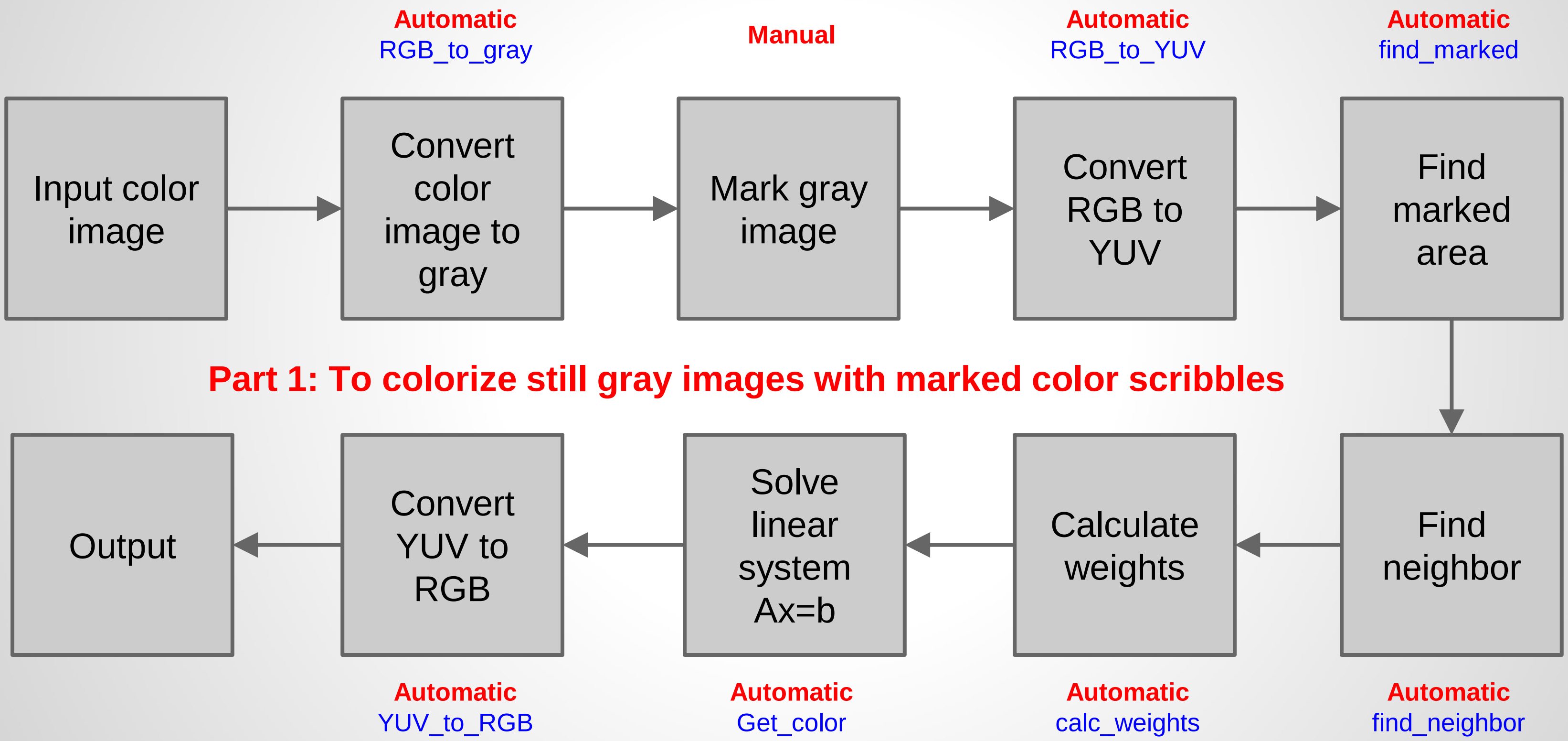
**Output**:
colorized
image





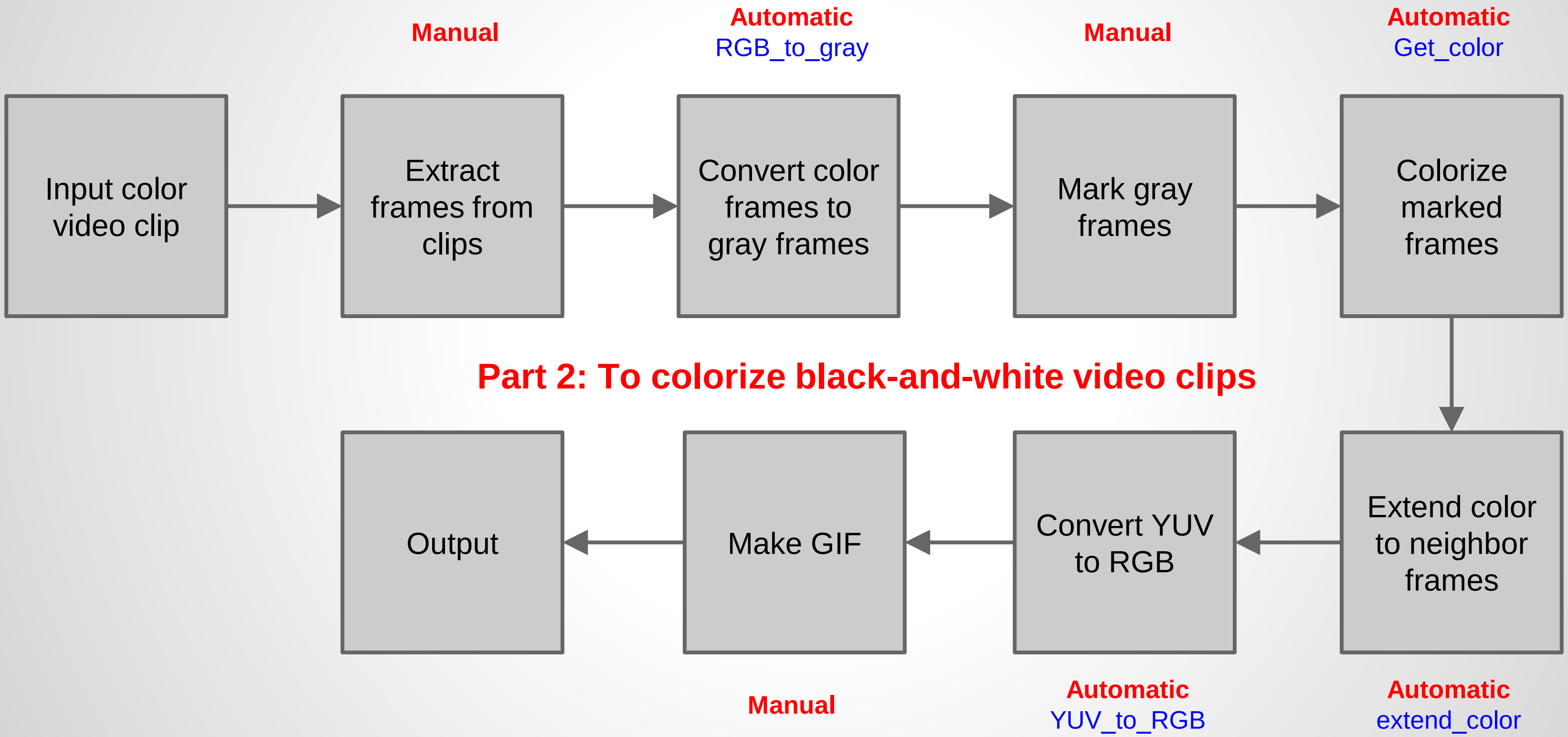
**Compared to
original image**

# Project Pipeline: Part 1



**Part 1: To colorize still gray images with marked color scribbles**

# Project Pipeline: Part 2



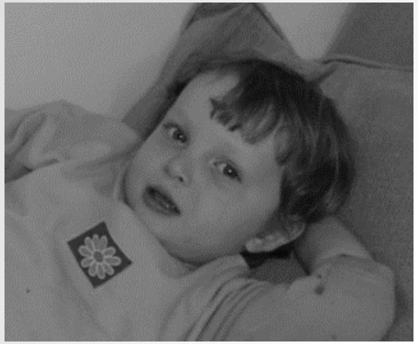
## Demonstration: Result Sets

- First I tested my code based on the set of images presented in the paper. After successfully replicated the results, I tested my codes based on four sets of my own images, downloaded online.

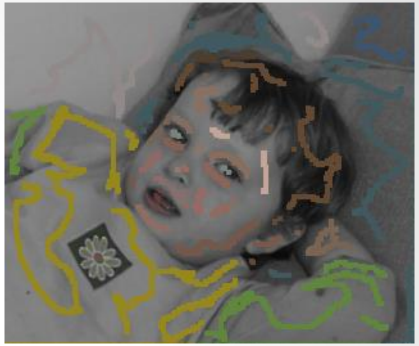
- Dropbox links to test images:

  https://www.dropbox.com/sh/b38gnbpgjp737h8/AAD-mae1x2mGW0dUNZd6Hqjoa?dl=0

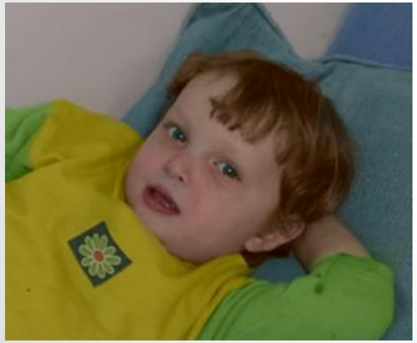
# Demonstration:  Result Sets



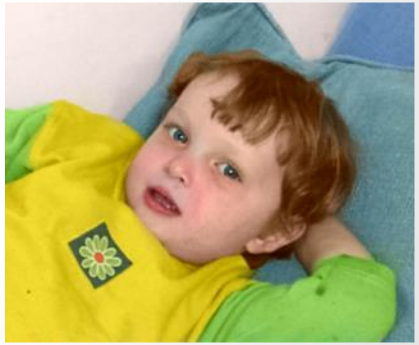
**Input 1**: black and white image



**Input 2**: marked image

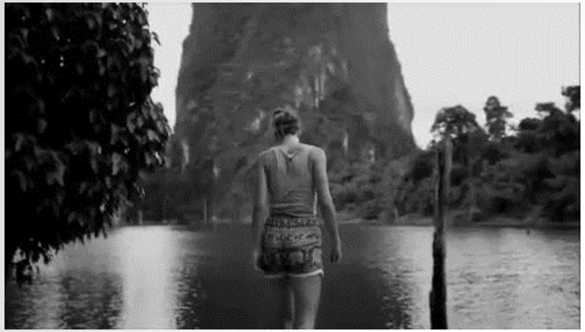


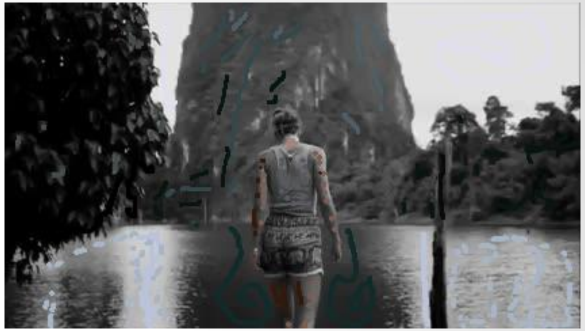
**Output**: colorized image

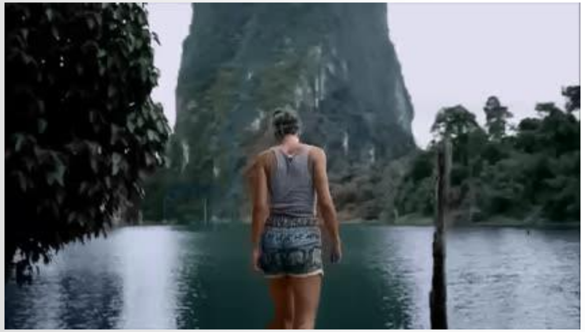


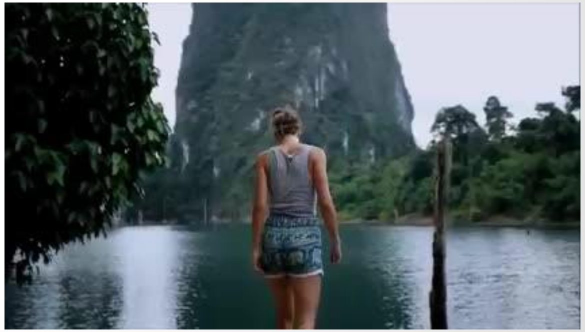
**Compared to original image**

# Demonstration:  Result Sets (cont'd)

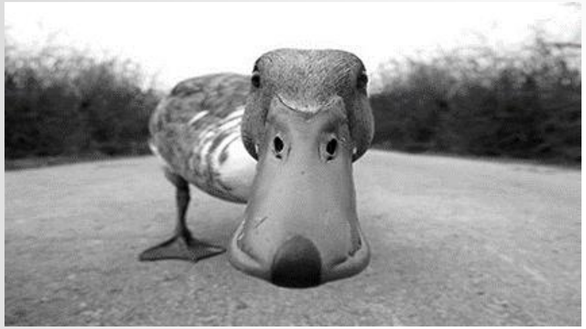

**Input 1**: black and white image

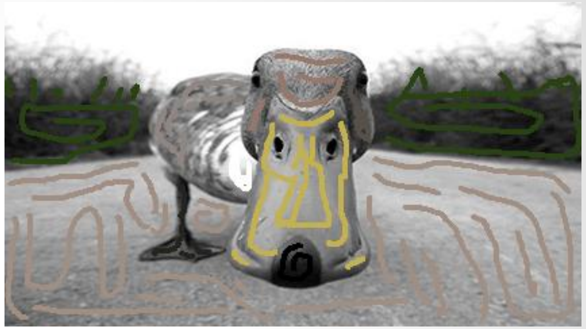

**Input 2**: marked image

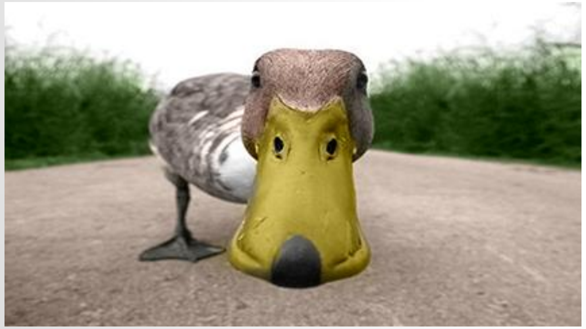

**Output**: colorized image

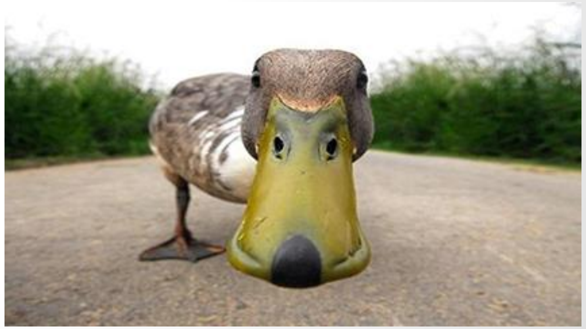

**Compared to original image**

# Demonstration: Result Sets (cont'd)

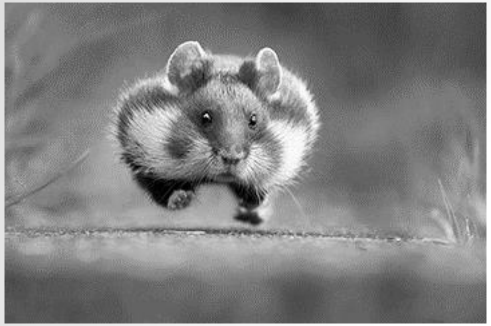

**Input 1**: black and white image


**Input 2**: marked image

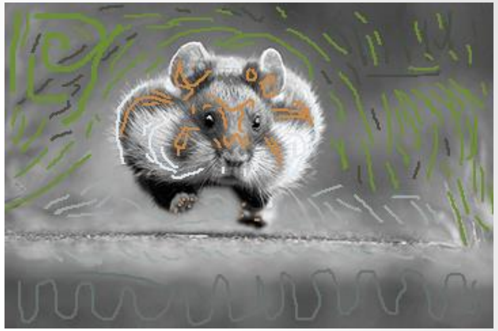

**Output**: colorized image


**Compared to original image**

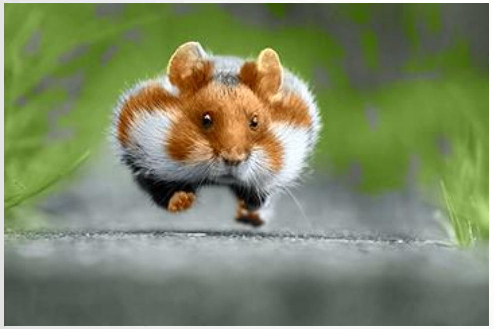
# Demonstration:  Result Sets (cont'd)



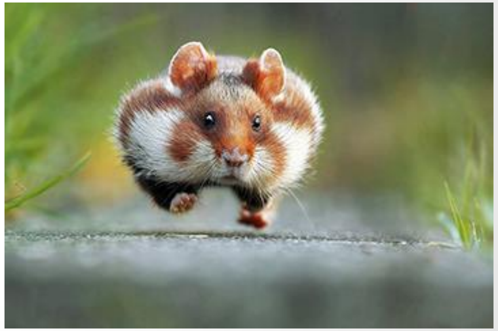
**Input 1**: black and white image



**Input 2**: marked image

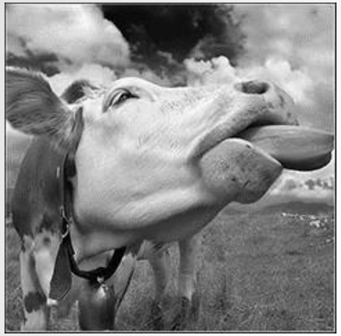


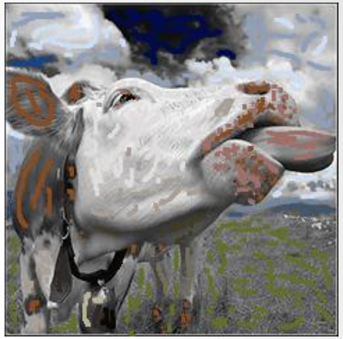
**Output**: colorized image

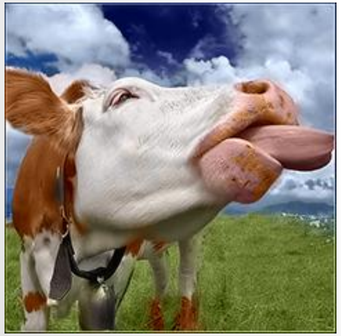


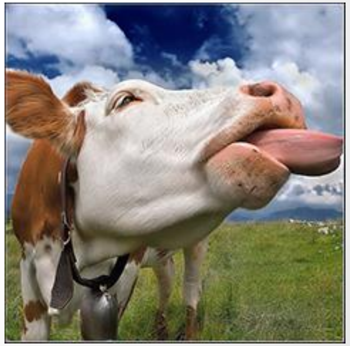
**Compared to original image**

# Demonstration:  Result Sets (cont'd)


**Input 1**: black and white image


**Input 2**: marked image


**Output**: colorized image


**Compared to original image**

## Project Development – Image Colorization

**Purpose**: to colorize still black-and-white image

**Step 1**: Take a colored image, and convert it to black-and-white

1. Convert the colored image into YUV format
2. Keep only intensity channel Y, and ignore the color channels U and V
3. Convert the Y channel back to RGB channel
4. Save to black-and-white image

Note: see function *RGB_to_gray* for coding details

# Project Development – Image Colorization

**Step 2**: Manually mark the black-and-white image using color scribbles

I used MS paint to mark the black-and-white image. At the same time, I open the color image in Photoshop to get the color information (i.e. R, G, B values). With these RGB values, I used the paint to mark the gray image with color scribbles.

# Project Development – Image Colorization

Step 3: To convert the black and white images from $RGB$ format to $YUV$ format, where $Y$ stands for intensities, and $U$ and $V$ contain relevant color information.

$$\begin{aligned} Y &= \phantom{-}0.299 \times R + 0.587 \times G + 0.114 \times B \\ U &= -0.147 \times R - 0.289 \times G + 0.436 \times B \\ V &= \phantom{-}0.615 \times R - 0.515 \times G - 0.100 \times B \end{aligned}$$

Note: the coefficients above are based on paper by Jack 2001 (see Source slide)

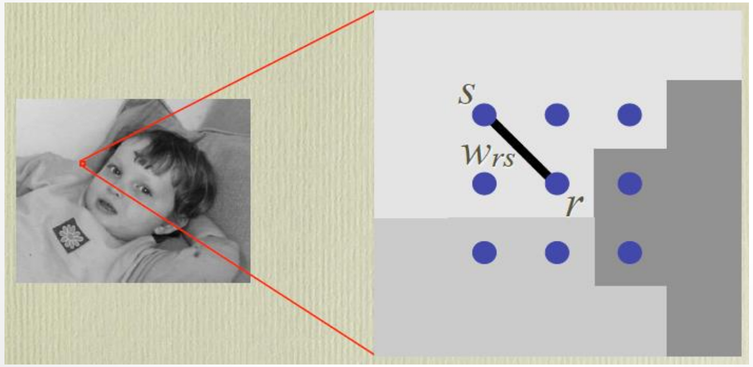
Note: See function *RGB_to_YUV* for coding details

## Project Development – Image Colorization

Step 4: To find the matrix U such that J(U) is minimized:

$$J(U) = \sum_{\forall r \in \text{Image}} \left( U(\boldsymbol{r}) - \sum_{\boldsymbol{s} \in N(\boldsymbol{r})} \omega_{\boldsymbol{rs}} \times U(\boldsymbol{s}) \right)^2$$

Subject to constraints: $U[r_m] = u_m$, where $r_m = [i_m, j_m]$ are all marked pixels and $u_m$ are pre-specified marked colors.



Note: the above figure is from one of A. Levin's presentations

# Project Development – Image Colorization

In order to solve the above constrained minimization problem, I reshaped the color matrix U from

$$M \times N \text{ dimension matrix } U[i, j]$$

to

$$P \times 1 \text{ dimension vertical vector } U[j \times N + i]$$

with $P = M \times N$, and M and N are number of rows and columns of input image

The above problem is equivalent to find $U$, such that

- For each colored pixel $r_m = [i_m, j_m]$, set the U value using the given color:$U[j_m \times N + i_m] = u_m$, which can be written in the vector form:

$$[0, \ldots, 1, \ldots, 0]_{1\times P}\boldsymbol{U}_{P\times 1} \ = \ u_m$$

# Project Development – Image Colorization

- For each uncolored pixel $r_m$, set the U value to be the weighted average value of neighbor pixels.

$$\begin{aligned} U[i_m, j_m] &- \omega_{[i_m-1,j_m-1],[i_m,j_m]} \times U[i_m-1, j_m-1] \\ &-\omega_{[i_m-1,j_m],[i_m,j_m]} \times U[i_m-1, j_m] - \omega_{[i_m-1,j_m+1],[i_m,j_m]} \times U[i_m-1, j_m+1] \\ &-\omega_{[i_m,j_m-1],[i_m,j_m]} \times U[i_m, j_m-1] - \omega_{[i_m,j_m+1],[i_m,j_m]} \times U[i_m, j_m+1] \\ &-\omega_{[i_m+1,j_m-1],[i_m,j_m]} \times U[i_m+1, j_m-1] - \omega_{[i_m+1,j_m],[i_m,j_m]} \times U[i_m+1, j_m] \\ &-\omega_{[i_m+1,j_m+1],[i_m,j_m]} \times U[i_m+1, j_m+1] = 0 \end{aligned}$$

Where the weight function $\omega_{\boldsymbol{rs}}$, based on equation (2) in the paper, is defined as

$$\omega_{\boldsymbol{rs}} \propto \exp\left(-\frac{(Y[\boldsymbol{r}] - Y[\boldsymbol{s}])^2}{2\sigma_{\boldsymbol{r}}^2}\right)$$

With $\sigma_r^2$ being the variance of the intensity values in the window around pixel **r** with size 1

# Project Development – Image Colorization

Combining the above two cases, the constrained problem is equivalent to solve the following linear system

$$AU = b$$

Where A is a $P \times P$ sparse matrix with

- If $r_m = [i_m, j_m]$ is colored, then

$$A[j_m \times N + i_m, l] = \begin{cases} 1 & \text{if } l = j_m \times N + i_m \\ 0 & \text{elsewhere} \end{cases}$$

- If $r_m = [i_m, j_m]$ is not colored, then

$$A[j_m \times N + i_m, l] = \begin{cases} 1 & \text{if } l = j_m \times N + i_m \\ -\omega_{[i_m-1, j_m-1],[i_m, j_m]} & \text{if } l = (j_m - 1) \times N + (i_m - 1) \\ -\omega_{[i_m-1, j_m],[i_m, j_m]} & \text{if } l = j_m \times N + (i_m - 1) \\ -\omega_{[i_m-1, j_m+1],[i_m, j_m]} & \text{if } l = (j_m + 1) \times N + (i_m - 1) \\ -\omega_{[i_m, j_m-1],[i_m, j_m]} & \text{if } l = (j_m - 1) \times N + i_m \\ -\omega_{[i_m, j_m+1],[i_m, j_m]} & \text{if } l = (j_m + 1) \times N + (i_m - 1) \\ -\omega_{[i_m+1, j_m-1],[i_m, j_m]} & \text{if } l = (j_m - 1) \times N + (i_m - 1) \\ -\omega_{[i_m+1, j_m],[i_m, j_m]} & \text{if } l = (j_m - 1) \times N + (i_m - 1) \\ -\omega_{[i_m+1, j_m+1],[i_m, j_m]} & \text{if } l = (j_m - 1) \times N + (i_m - 1) \\ 0 & \text{elsewhere} \end{cases}$$

## Project Development – Image Colorization

and

$$b[j_m \times N + i_m] = \begin{cases} u_m & \text{if the pixel } r_m = [i_m, j_m] \text{ is colored} \\ 0 & \text{if the pixel } r_m = [i_m, j_m] \text{ is not colored} \end{cases}$$

Note: see function *get_color* for coding details

# Project Development – Image Colorization

Step 5: Solving the linear system (2) gives us the color channel $U$ in the vector form

$$U = A^{-1}b$$

then convert the $P \times 1$ vector into a $M \times N$ matrix form.

Note:

- The $V$ color channel can be calculated in the same way as U color channel
- The Y intensity channel is calculated to be the black and white intensities
- See function *get_color* for coding details

## Project Development – Image Colorization

Step 6: Convert the colorized image from YUV format to RGB format

$$\begin{aligned} R &= Y && + 1.140V \\ G &= Y - 0.395U && - 0.581V \\ B &= Y + 2.032U && \end{aligned}$$

Note: see function *YUV_to_RGB* for coding details

## Project Development – Video Colorization

**Purpose**: to extend colors to grey frames in video clips

**Step 1**: Manually extract frames of images from video

I manually extract image frames from the source video based online converter filezigzag. Frame for every XXX

## Project Development – Video Colorization

**Step 2:** Convert the colored frames into black-and-white images

In this step, I first convert the image from RGB format to YUV format, then only keep intensity information (Y channel), and ignore the color information (U and V channel). Finally convert back to RGB format to form a gray image.

Note: See function *RGB_to_gray* for coding details

## Project Development – Video Colorization

**Step 3:** Manually mark selected black-and-white images using color scribbles

I used MS paint as a tool to mark the gray images. In order to find the right color to mark, I open the color image in Photoshop, and find the R, G, B values of the color I’d like to mark. In my example, I chose to mark 1 frame for every 10 image frames.

# Project Development – Video Colorization

**Step 4:** Colorized the marked black and white images

It follows the same procedures as colorizing the still black-and-white images. We may propose different colors from those in the source video, if necessary.

Note: See function *get_color* for coding details

# Project Development – Image Colorization

**Step 5:** Colorize the marked black and white frames based on colored frame. For example, colorize frame 1 to frame 9 based on the colorized frame 0

1. Convert both colored image (e.g. frame 0) and black and white image (e.g. frame 1) from RGB format to YUV format
2. For each pixel $\boldsymbol{r} = [i, j]$ in the black and white image, loop through each pixel $\boldsymbol{s} = [i^{'}, j^{'}]$ on the colored image, such that where the Y channel of black and white frame and colored frame is minimized. In particular, difference between $\mathrm{YUV}[i, j, 0]$ and $\mathrm{YUV'}[i', j', 0]$ is minimized.
3. Set $\mathrm{YUV}[i, j, 1] = \mathrm{YUV}^{'}[i^{'}, j^{'}, 1]$ and $\mathrm{YUV}[i, j, 2] = \mathrm{YUV}^{'}[i^{'}, j^{'}, 2]$
4. Convert the colorized frame (e.g. frame 1) from YUV format back to RGB format
5. Perform 1 to 4 for frame 1 to frame 9

Note: See function *extend_color* for coding details

## Project Development – Image Colorization

**Step 6:** Manually convert the colorized frames into GIF format

I used the free online tool GIFMAKER to convert the image frames to GIF

## Project Development – Image Colorization

What would you do differently?

1. If had chance to start from beginning, I would choose images that are simple in colors. That is, the colors are more uniform across pixels, so that I may spend less time on marking the black-and-white images.
2. I would test my algorithm on more than 1 video clips. But this takes time to mark frames, especially I would like to make sure that the same area is colorized in the same color across frames.

# Project Development – Image Colorization

Any challenges or issues you met?

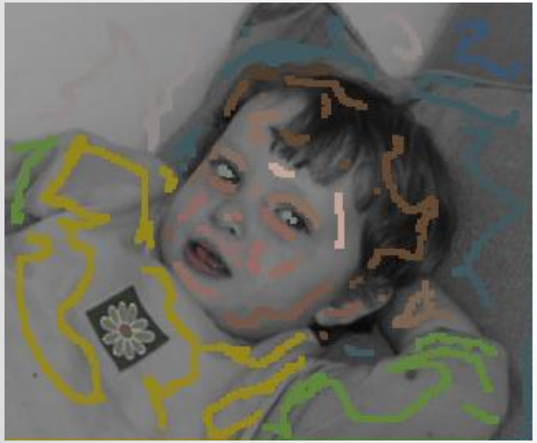
1. Not sure how to define the neighbor of a pixel. I choose the distance to be 1 pixel, and hence there are 8 pixels in the neighbor for a specified pixel
2. Not sure how to solve the constrained least square problem. It took me a very long time to figure it out, by writing down the formulas and do some derivation. Reading authors' matlab codes also helps me understand.
3. Not sure the conversion formula between YUV and RGB format. I read the reference paper by Jack, and took the formula there.

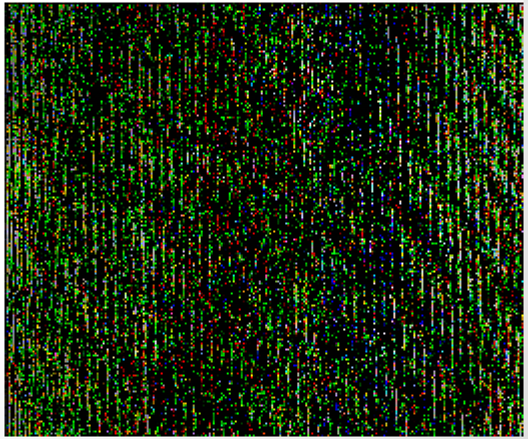
The authors of the paper shared their matlab codes about how to colorize still images using matlab, which provides a very good source to start with. Without it, I may not be able to complete my final project.

## Failed Image 1

**Output**: I am using the images in the paper as my initial test case to replicate the algorithm in the paper. The output is just noises with no clear image.

**Reason**: I forgot to put 1 to in the diagonal items in the weight matrix A. Note: the colorized image is derived by solving the linear system: Ax = b

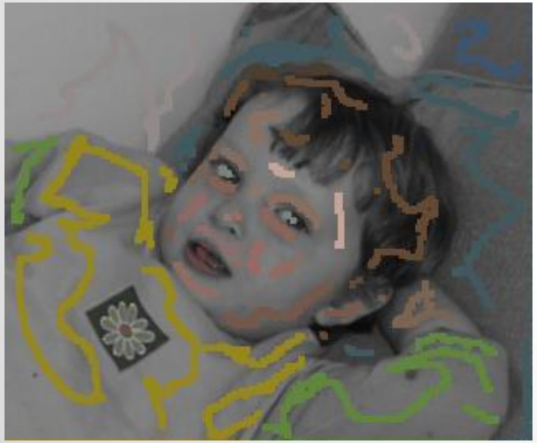


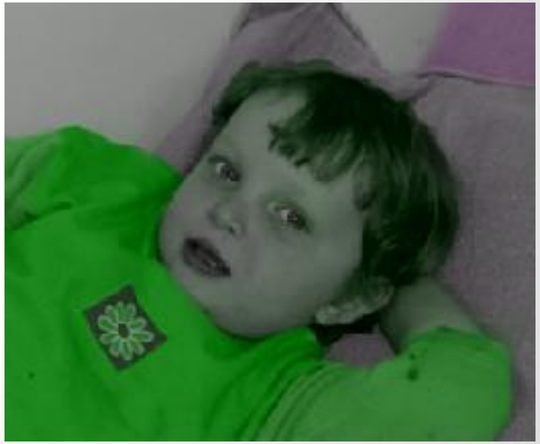


## Failed Image 2

**Output**: I am using the images in the paper as my initial test case to replicate the algorithm in the paper. The output tends to more green.

**Reason**: When I convert YUV format to RGB format, I made a mistake in the code that all R, G, B set to be R.

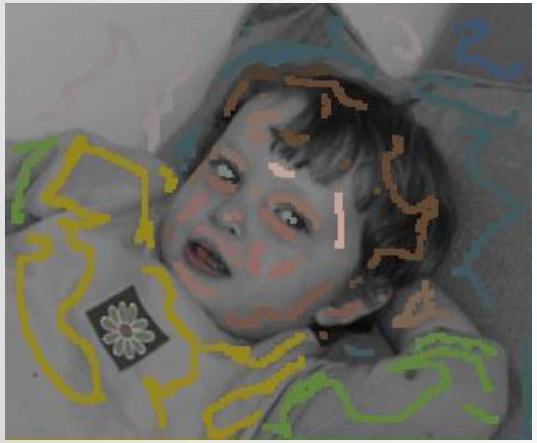




## Failed Image 3

**Output**: I am using the images in the paper as my initial test case to replicate the algorithm in the paper. The output is all black ...

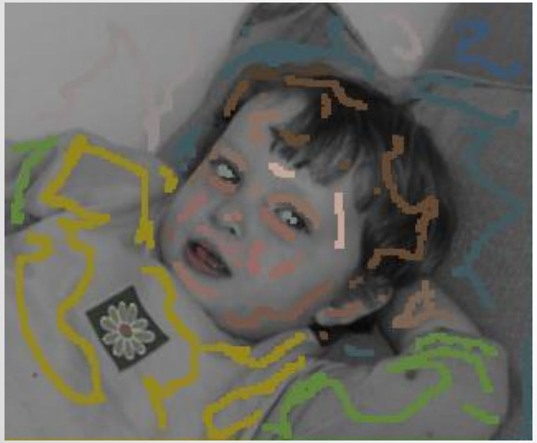
**Reason**: When I convert RGB format to YUV format, I forgot to divide by 255.0.

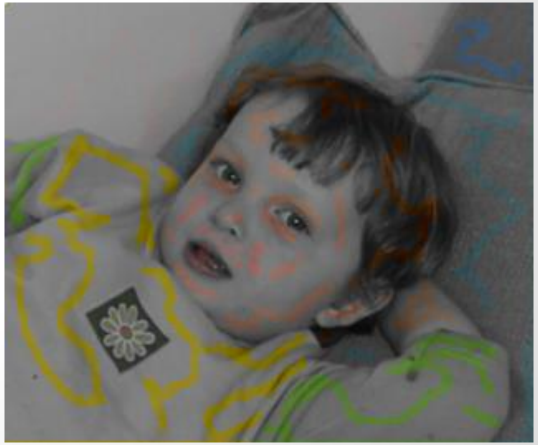


# Failed Image 4

**Output**: I am using the images in the paper as my initial test case to replicate the algorithm in the paper. The output looks the same as the marked image

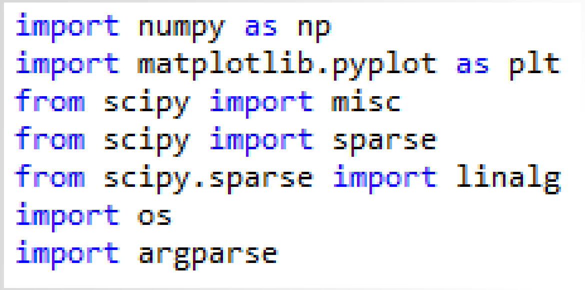
**Reason**: I forgot to update the row index, so that the weight matrix is not correct.





# Computation: Code Functional Description

## Part 1: To colorize still Image

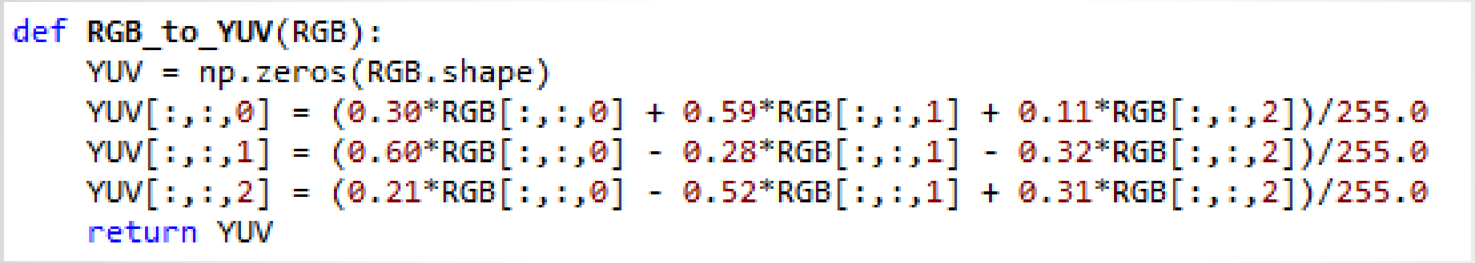


```
import numpy as np
import matplotlib.pyplot as plt
from scipy import misc
from scipy import sparse
from scipy.sparse import linalg
import os
import argparse
```

Above is the list of libraries I used in my final project

# Computation: Code Functional Description

## Part 1: To colorize still Image

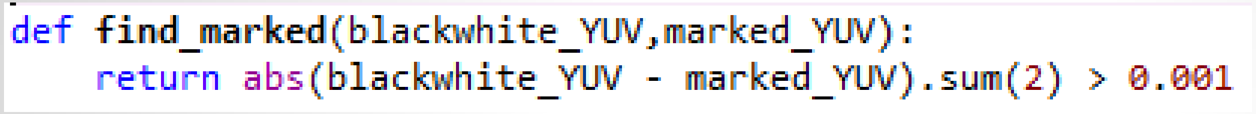


```
def RGB_to_YUV(RGB):
    YUV = np.zeros(RGB.shape)
    YUV[:,:,0] = (0.30*RGB[:,:,0] + 0.59*RGB[:,:,1] + 0.11*RGB[:,:,2])/255.0
    YUV[:,:,1] = (0.60*RGB[:,:,0] - 0.28*RGB[:,:,1] - 0.32*RGB[:,:,2])/255.0
    YUV[:,:,2] = (0.21*RGB[:,:,0] - 0.52*RGB[:,:,1] + 0.31*RGB[:,:,2])/255.0
    return YUV
```

Step 1: As suggested in the paper, I first convert the image in the RGB format to YUV format, where Y is the intensity channel, and U and V channel include color information.

# Computation: Code Functional Description

**Part 1: To colorize still Image**

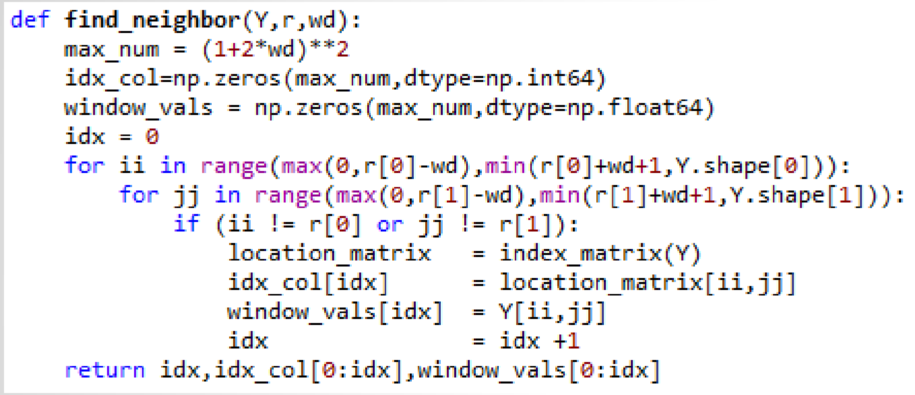


```
def find_marked(blackwhite_YUV,marked_YUV):
    return abs(blackwhite_YUV - marked_YUV).sum(2) > 0.001
```

Step 2: find the marked area with color scribbles. This is where there are differences in Y, U or V channel.

# Computation: Code Functional Description

## Part 1: To colorize still Image



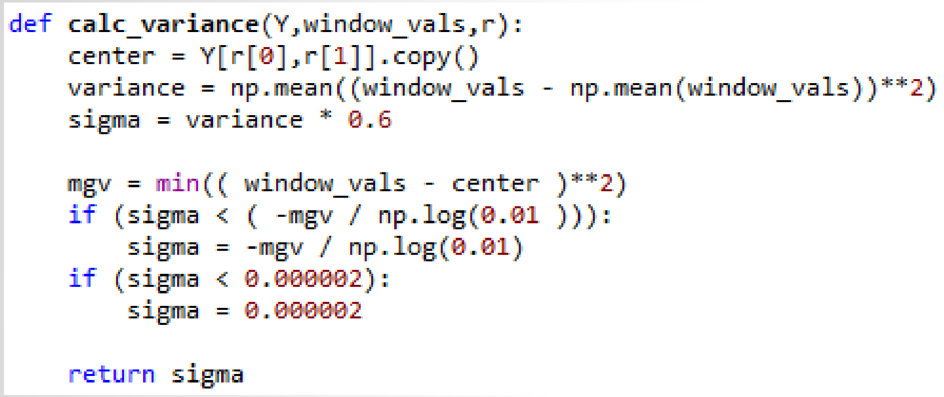
```python
def find_neighbor(Y,r,wd):
    max_num = (1+2*wd)**2
    idx_col=np.zeros(max_num,dtype=np.int64)
    window_vals = np.zeros(max_num,dtype=np.float64)
    idx = 0
    for ii in range(max(0,r[0]-wd),min(r[0]+wd+1,Y.shape[0])):
        for jj in range(max(0,r[1]-wd),min(r[1]+wd+1,Y.shape[1])):
            if (ii != r[0] or jj != r[1]):
                location_matrix   = index_matrix(Y)
                idx_col[idx]      = location_matrix[ii,jj]
                window_vals[idx]  = Y[ii,jj]
                idx               = idx +1
    return idx,idx_col[0:idx],window_vals[0:idx]
```

Step 3: For each pixel **r**=[i,j] in the image, identify its neighbor with width 1 pixel.

| | | |
|---|---|---|
| i-1, j-1 | i-1, j | i-1, j+1 |
| i, j-1 | i,j | i, j+1 |
| i+1, i-1 | i+1, j | i+1, j+1 |

## Computation: Code Functional Description

**Part 1: To colorize still Image**

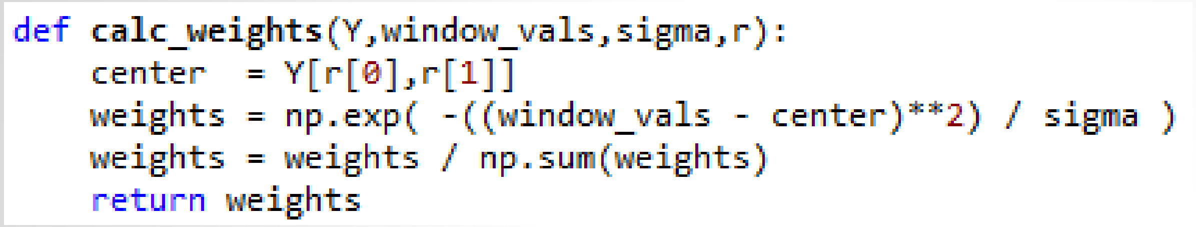


```python
def calc_variance(Y,window_vals,r):
    center = Y[r[0],r[1]].copy()
    variance = np.mean((window_vals - np.mean(window_vals))**2)
    sigma = variance * 0.6

    mgv = min(( window_vals - center )**2)
    if (sigma < ( -mgv / np.log(0.01 ))):
        sigma = -mgv / np.log(0.01)
    if (sigma < 0.000002):
        sigma = 0.000002

    return sigma
```

Step 4: Calculate variance in the neighbor of each pixel **r**=[i,j]. The variance is required to calculate the weight matrix A

# Computation: Code Functional Description

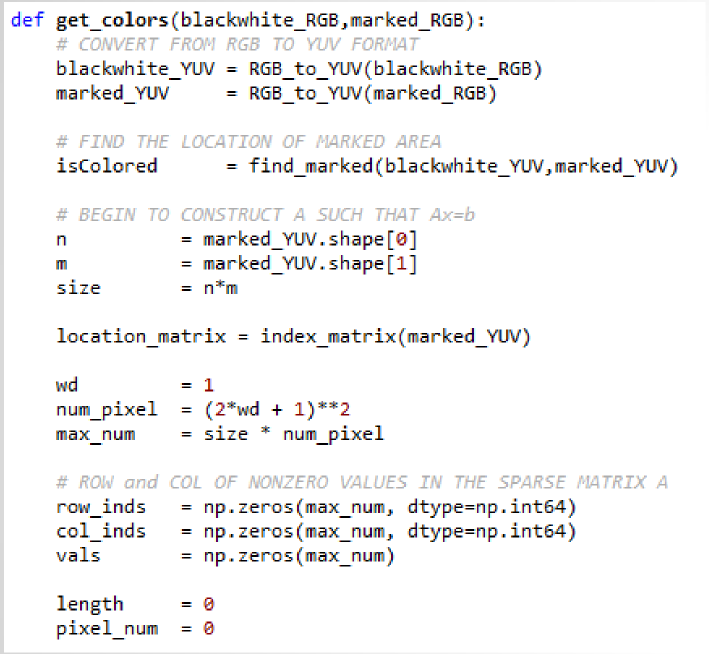
## Part 1: To colorize still Image



```
def calc_weights(Y,window_vals,sigma,r):
    center  = Y[r[0],r[1]]
    weights = np.exp( -((window_vals - center)**2) / sigma )
    weights = weights / np.sum(weights)
    return weights
```

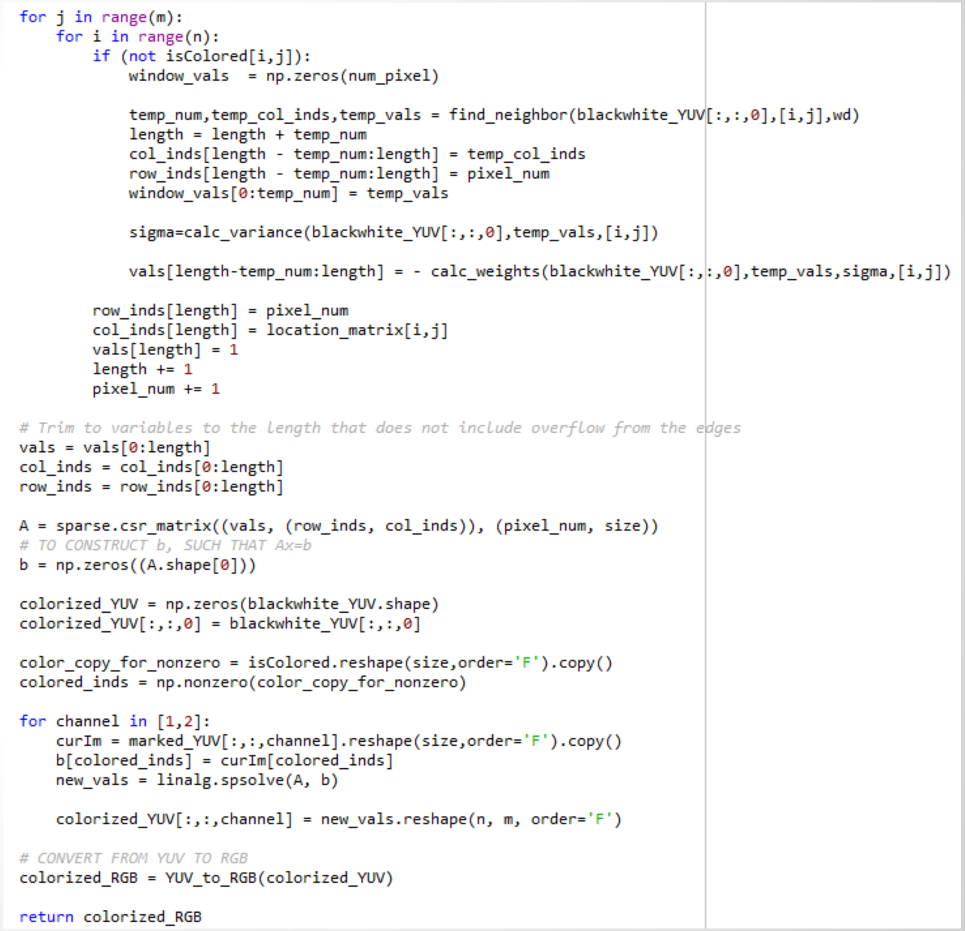
Step 5: for each pixel **r**=[i,j], calculate the weight of each pixel in its neighbor based on equation 2 in the paper. The weights will be added into the weight matrix A, to form the linear system Ax = b.

# Computation: Code Functional Description



```
def get_colors(blackwhite_RGB,marked_RGB):
    # CONVERT FROM RGB TO YUV FORMAT
    blackwhite_YUV = RGB_to_YUV(blackwhite_RGB)
    marked_YUV     = RGB_to_YUV(marked_RGB)

    # FIND THE LOCATION OF MARKED AREA
    isColored      = find_marked(blackwhite_YUV,marked_YUV)

    # BEGIN TO CONSTRUCT A SUCH THAT Ax=b
    n           = marked_YUV.shape[0]
    m           = marked_YUV.shape[1]
    size        = n*m

    location_matrix = index_matrix(marked_YUV)

    wd          = 1
    num_pixel   = (2*wd + 1)**2
    max_num     = size * num_pixel

    # ROW and COL OF NONZERO VALUES IN THE SPARSE MATRIX A
    row_inds    = np.zeros(max_num, dtype=np.int64)
    col_inds    = np.zeros(max_num, dtype=np.int64)
    vals        = np.zeros(max_num)

    length      = 0
    pixel_num   = 0
```

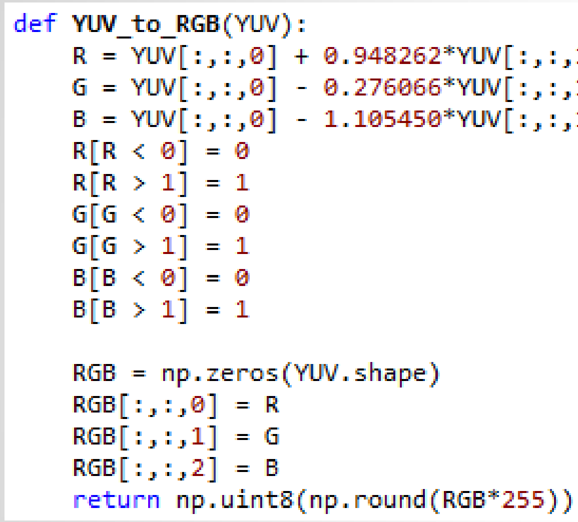
Step 6: Once derived the weight matrix A, the colorized image is calculated by solving the linear system Ax = b. The detailed algorithm is shown in the slides of development details.



```
for j in range(m):
    for i in range(n):
        if (not isColored[i,j]):
            window_vals  = np.zeros(num_pixel)

            temp_num,temp_col_inds,temp_vals = find_neighbor(blackwhite_YUV[:,:,0],[i,j],wd)
            length = length + temp_num
            col_inds[length - temp_num:length] = temp_col_inds
            row_inds[length - temp_num:length] = pixel_num
            window_vals[0:temp_num] = temp_vals

            sigma=calc_variance(blackwhite_YUV[:,:,0],temp_vals,[i,j])

            vals[length-temp_num:length] = - calc_weights(blackwhite_YUV[:,:,0],temp_vals,sigma,[i,j])

        row_inds[length] = pixel_num
        col_inds[length] = location_matrix[i,j]
        vals[length] = 1
        length += 1
        pixel_num += 1

# Trim to variables to the length that does not include overflow from the edges
vals = vals[0:length]
col_inds = col_inds[0:length]
row_inds = row_inds[0:length]

A = sparse.csr_matrix((vals, (row_inds, col_inds)), (pixel_num, size))
# TO CONSTRUCT b, SUCH THAT Ax=b
b = np.zeros((A.shape[0]))

colorized_YUV = np.zeros(blackwhite_YUV.shape)
colorized_YUV[:,:,0] = blackwhite_YUV[:,:,0]

color_copy_for_nonzero = isColored.reshape(size,order='F').copy()
colored_inds = np.nonzero(color_copy_for_nonzero)

for channel in [1,2]:
    curIm = marked_YUV[:,:,channel].reshape(size,order='F').copy()
    b[colored_inds] = curIm[colored_inds]
    new_vals = linalg.spsolve(A, b)

    colorized_YUV[:,:,channel] = new_vals.reshape(n, m, order='F')

# CONVERT FROM YUV TO RGB
colorized_RGB = YUV_to_RGB(colorized_YUV)

return colorized_RGB
```

# Computation: Code Functional Description

## Part 1: To colorize still Image

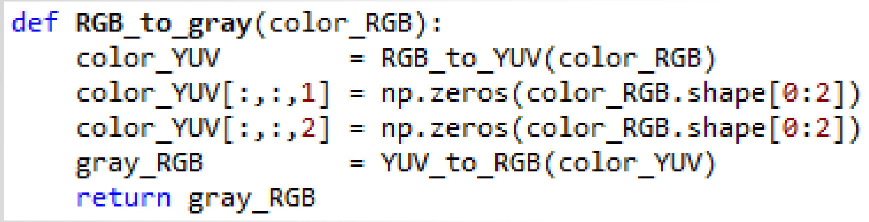


```
def YUV_to_RGB(YUV):
    R = YUV[:,:,0] + 0.948262*YUV[:,:,1] + 0.624013*YUV[:,:,2]
    G = YUV[:,:,0] - 0.276066*YUV[:,:,1] - 0.639810*YUV[:,:,2]
    B = YUV[:,:,0] - 1.105450*YUV[:,:,1] + 1.729860*YUV[:,:,2]
    R[R < 0] = 0
    R[R > 1] = 1
    G[G < 0] = 0
    G[G > 1] = 1
    B[B < 0] = 0
    B[B > 1] = 1

    RGB = np.zeros(YUV.shape)
    RGB[:,:,0] = R
    RGB[:,:,1] = G
    RGB[:,:,2] = B
    return np.uint8(np.round(RGB*255))
```

Step 7: Convert the colorized image from YUV format back to RGB format, and save the output.

## Computation: Code Functional Description

**Part 2: To colorize black and white video clips**

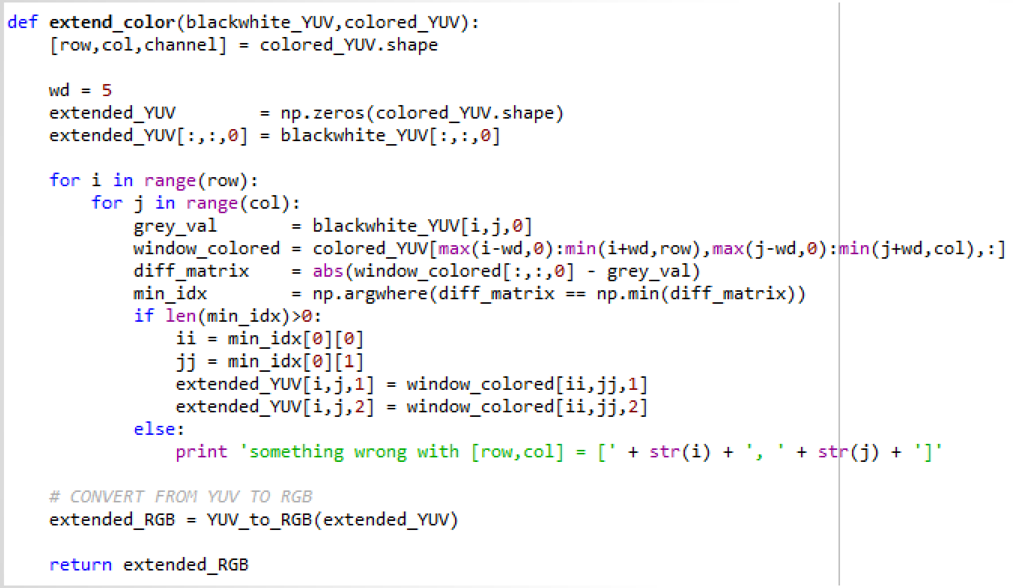


```
def RGB_to_gray(color_RGB):
    color_YUV        = RGB_to_YUV(color_RGB)
    color_YUV[:,:,1] = np.zeros(color_RGB.shape[0:2])
    color_YUV[:,:,2] = np.zeros(color_RGB.shape[0:2])
    gray_RGB         = YUV_to_RGB(color_YUV)
    return gray_RGB
```

Step 1: given the image frames extracted from video clip, I first convert the color image into black-and-white images. That is I convert the image from RGB to YUV format, but only keep the Y channel (intensity channel) and ignore the U and V channels (color channels). Then convert back to RGB format.

# Computation: Code Functional Description

## Part 2: To colorize black and white video clips



```
def extend_color(blackwhite_YUV,colored_YUV):
    [row,col,channel] = colored_YUV.shape

    wd = 5
    extended_YUV        = np.zeros(colored_YUV.shape)
    extended_YUV[:,:,0] = blackwhite_YUV[:,:,0]

    for i in range(row):
        for j in range(col):
            grey_val       = blackwhite_YUV[i,j,0]
            window_colored = colored_YUV[max(i-wd,0):min(i+wd,row),max(j-wd,0):min(j+wd,col),:]
            diff_matrix    = abs(window_colored[:,:,0] - grey_val)
            min_idx        = np.argwhere(diff_matrix == np.min(diff_matrix))
            if len(min_idx)>0:
                ii = min_idx[0][0]
                jj = min_idx[0][1]
                extended_YUV[i,j,1] = window_colored[ii,jj,1]
                extended_YUV[i,j,2] = window_colored[ii,jj,2]
            else:
                print 'something wrong with [row,col] = [' + str(i) + ', ' + str(j) + ']'

    # CONVERT FROM YUV TO RGB
    extended_RGB = YUV_to_RGB(extended_YUV)

    return extended_RGB
```

Given one gray frame and one colorized frame, for each pixel r=[i,j] in the gray frame, I compare its intensity value (Y channel) to intensity values in the neighbor of r with width 5 pixels of the colorized image. I identify the location in the colorized image s=[ii,jj], such that the different Y[r]-Y[s] is minimized in the neighbor of pixel r with width 5 pixels. Then let U[r] and V[r] values of gray image to equal to U[s] and V[s] of colorized image
gray_U[i,j] = colorized_U[ii,jj]
gray_V[i,j] = colorized_V[ii,jj]
Finally, I converted the YUV format back to RGB format

Step 2: After manually marking the black-and-white image using color scribbles, I used the *get_color* in part 1 to colorize the marked image. Then I extended the colorized image to its neighbor black-and-white images. In my test example, I marked and colorized 1 frame for every 10 frames, and extend the colorized frame to the next 9 frames. For example, I colorized the 0th image, and extend its color to frame 1, 2, ..., and 9

## Resources

- Source paper *Colorization using Optimization* by Anat Levein, Dani Lischinski, Yair Weiss, https://www.cs.huji.ac.il/~yweiss/Colorization/colorization-siggraph04.pdf

- Source matlab code
  https://www.cs.huji.ac.il/~yweiss/Colorization/colorization.zip

- Conversion from RGB to YUV and YUV to RGB
  Jack, K. 2001, Video Demystified, 3rd edition ed. Elsevier Science & Technology

- Duck image
  https://i.ytimg.com/vi/vufs2Y1pFeQ/maxresdefault.jpg

## Resources

- Cattle image https://i.pinimg.com/736x/9f/c4/c6/9fc4c6ce00de0d4ffb3f98f9c7d4e852--cow-face-funny-cows.jpg

- Rat image https://cdn.vox-cdn.com/thumbor/tSvdZRmVKzBGDXujME9MSw1VqGo=/60x0:939x586/1280x854/cdn.vox-cdn.com/uploads/chorus_image/image/48349001/WINNER-Julian-Rad.0.0.jpg

- Girl image from Youtube using screen shot https://www.youtube.com/watch?v=HgmDCkkLQT0

## Appendix: Your Code

**Code Language:**

- Python 2.7

**List of code files:**

- README.txt
- colorize_image.py
- colorize_video.py

Note:

- *README.txt* includes the instructions to run python code
- The python codes has been compressed in resources.zip and uploaded

Dropbox link

- https://www.dropbox.com/sh/b38gnbpgjp737h8/AAD-mae1x2mGW0dUNZd6Hqjoa?dl=0

## Credits or Thanks

I would like to thank the following tool providers for my final project

- Online converter from video to frames

http://www.filezigzag.com/Download/DT.aspx?T=107b8c2f-041a-4393-a14c-80fb36a684f5

- GIF maker

http://gifmaker.me/