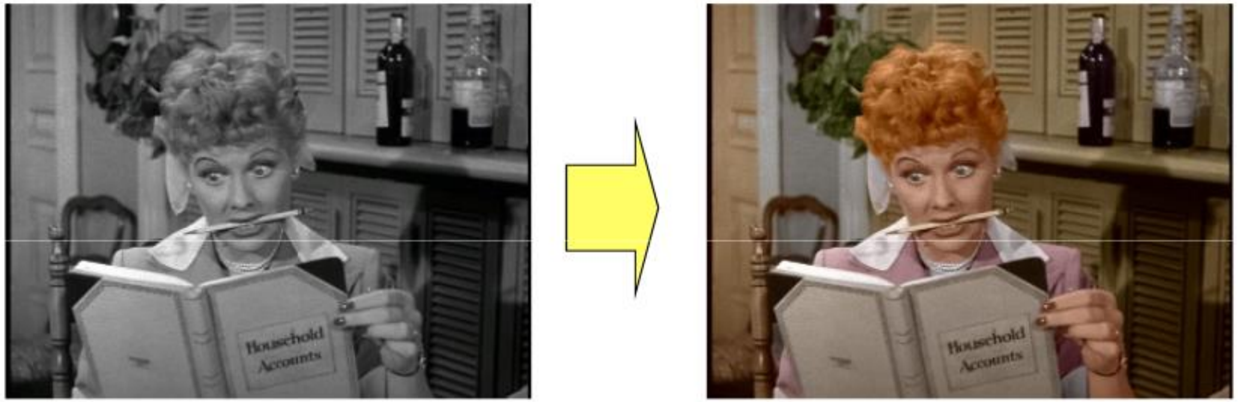
Colorization by Optimization

Sudharshan Ashwin Renganathan

Aerospace Systems Design Laboratory
School of Aerospace Engineering
ashwinsr@gatech.edu

Introduction

- Given a grayscale image with some user inputs, we want to generate a *colorized* image



Colorization: a computer-assisted process of adding color to a monochrome image or movie. (Invented by Wilson Markle, 1970)

Motivation

- Typical colorization process involves
  - Delineating *region* boundary for the image
  - Choose a color and apply to region
  - Track regions across frames in case of video

- Limitation – time consuming and labor intensive
- Motivation – How can we automate the process and minimize user input?

Colorization by Optimization

- Works under the simple principle that

> "***Nearby pixels with similar intensities should have the same color***"

- User scribbles image with color
- Code propagates color to regions appropriately
- The idea is completely proprietary to Levin et.al.[1]
- MATLAB source code available in the open internet

[1] http://www.cs.huji.ac.il/~yweiss/Colorization/

Approach

Grayscale Image → Convert to YIQ format → GS_YIQ

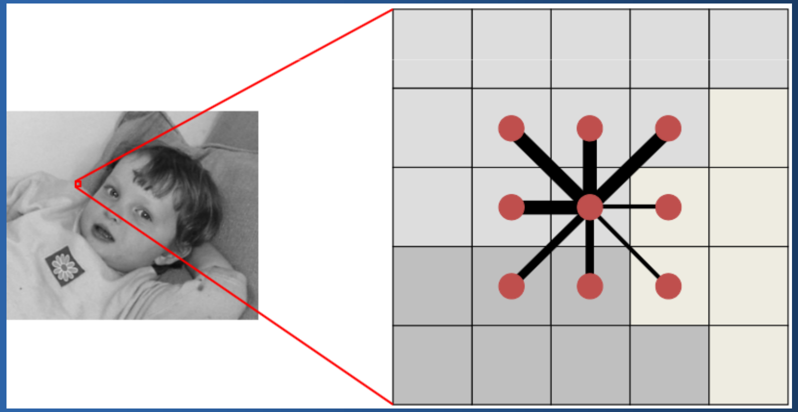
Scribbled Image → Convert to YIQ format → SC_YIQ

→ Colorize(GS_YIQ, SC_YIQ) → Final Colorized Output

Y – grayscale pixel intensity (luminance)
I, Q – color palette values (chrominance)

Approach…

- Each image has 3 channels Y, U, V
- Y-intensity; UV-Color
- Minimize difference in U,V for each pixel from neighbors
    - Neighbors weighted based on similarity in intensity (Y)
- Constraints – pixels colored by user



$$E(U) = \sum_{i=1}^{npixels} \left[U(i) - \sum_j w_j U(j) \right]^2$$

$$w_j = e^{-\left(Y(i)-Y(j)\right)^2/\sigma_i^2}$$

$$E(V) = \sum_{i=1}^{npixels} \left[V(i) - \sum_j w_j V(j) \right]^2$$

- w_j are called 'affinity functions' which are nothing but **weights**
- σ_i^2 is the variance of pixel intensity

Challenges faced

- Use of MATLAB Optimization
  - Number of variables = O(npixels)
  - For a 512 x 512 image, this is 262,144 variables
  - MATLAB reports ‘insufficient memory’ at this scale
  - Limited by number of variables
- Decipher Levin’s code to make modular changes
  - Mathematical treatment not published
  - Code is not annotated
  - Hence not easy to understand completely to make desired changes
- Problem was re-scoped
  - Re-formulate the problem on my own.
    - Understand the mathematical treatment.
  - Write my own code to implement formulation
  - Test code on some test images

Goal

- Re-formulate the optimization problem
  - Work out missing steps in the original paper
  - Solve problem analytically and present solution
- Recreate the colorization code
  - Helps understand process better
  - Helps understand the math better
  - Helps make incremental changes easier
- Test code with sample images.

Problem Formulation

$$E(U) = \sum_{i=1}^{npixels} \left[U(i) - \sum_j w_j U(j) \right]^2$$

$$E(U) = \sum_{i=1}^{npixels} U(i)^2 - 2U(i) \sum_j w_j U(j) + \left[\sum_j w_j U(j) \right]^2$$

$$E(U) = \sum_{i=1}^{npixels} U(i) \left(U(i) - 2 \sum_j w_j U(j) \right) + \left[\sum_j w_j U(j) \right]^2$$

- *Each cost function can be expressed in matrix form*
- *A and B are sparse matrices composed only of w_j's*
- *Turns out E can be minimized just using Matrix Algebra*

$$E(U) = \mathrm{U}^{\mathrm{T}}\mathrm{AU} + \mathrm{U}^{\mathrm{T}}\left(\mathrm{BB}^{\mathrm{T}}\right)\mathrm{U}$$

$$\boxed{E(U) = U^T (A + BB^T) U}$$

$$\boxed{E(V) = V^T (A + BB^T) V}$$

- *Input image is of size m x n*
- *U and V are vectors of size 1 X mn*
- *A and B are sparse square matrices of size mn x mn*

Problem Formulation

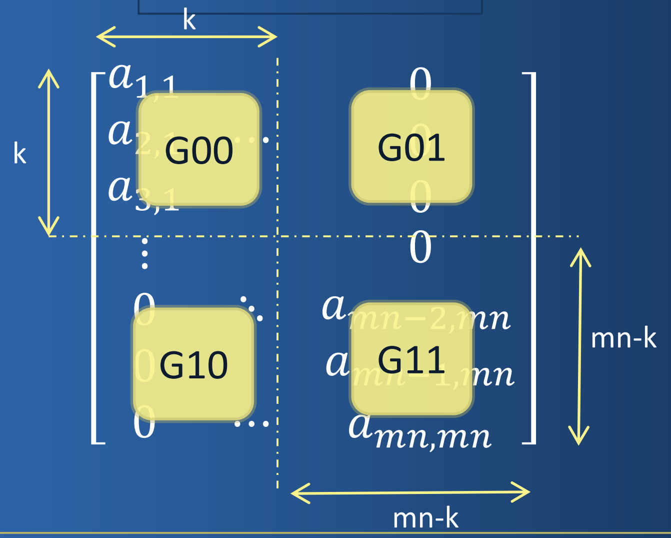
$$E(U) = U^T(A + BB^T)U$$

1

$$E(U) = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ \vdots \\ \vdots \\ U_{mn} \end{bmatrix}^T \begin{bmatrix} a_{1,1} & & 0 \\ a_{2,1} & \cdots & 0 \\ a_{3,1} & & 0 \\ \vdots & & 0 \\ 0 & \ddots & a_{mn-2,mn} \\ 0 & & a_{mn-1,mn} \\ 0 & \cdots & a_{mn,mn} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ \vdots \\ \vdots \\ U_{mn} \end{bmatrix}$$

2

$$(A + BB^T)$$



3

$$E = \begin{bmatrix} x \\ q \end{bmatrix}^T \begin{bmatrix} G00 & G01 \\ G10 & G11 \end{bmatrix} \begin{bmatrix} x \\ q \end{bmatrix}$$

4

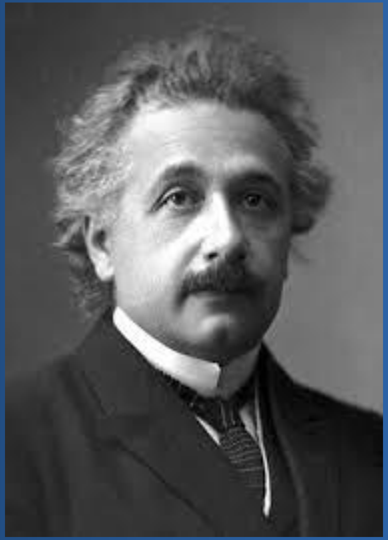
$$\frac{\partial \mathrm{E}}{\partial \mathrm{x}} = x^T(G00 + G00^T) + q(G01 + G10^T) = 0$$

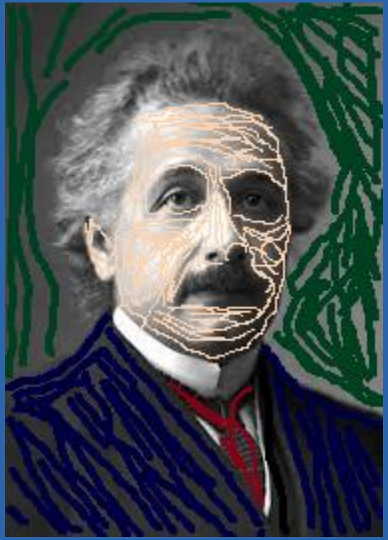
5

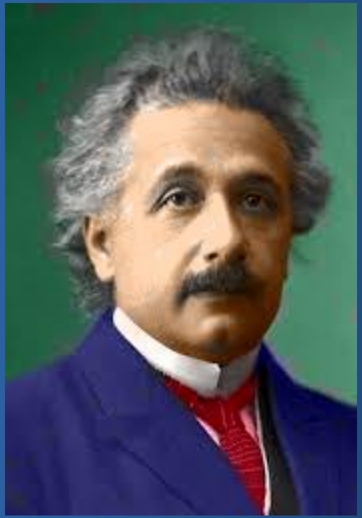
$$\boxed{x = -(G00 + G00^T)^{-1} * q(G01 + G10^T)}$$

- G00 corresponds to un-scribbled pixels
  - Unknowns
- G11 corresponds to scribbled pixels
  - Constraints
- G01 and G10 correspond to both categories
- We split U and V vectors into two vectors
  - x: unknown pixels of length 1 x k
  - q: known pixels of length 1 x mn-k

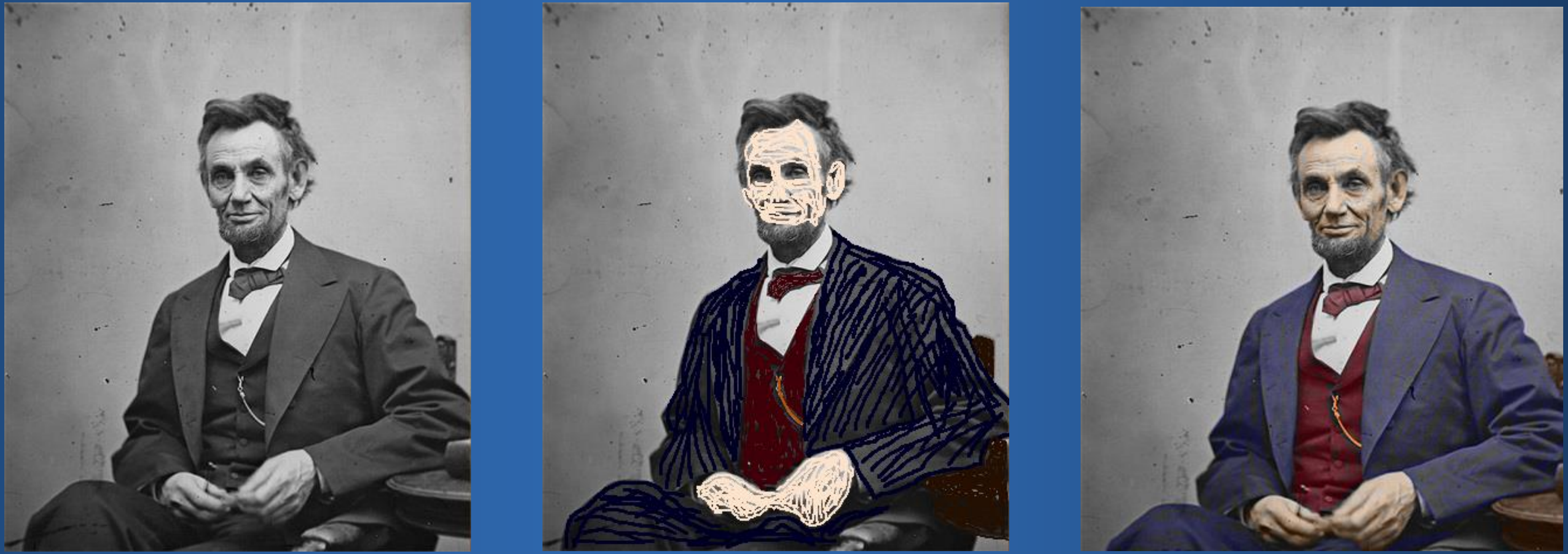
Results



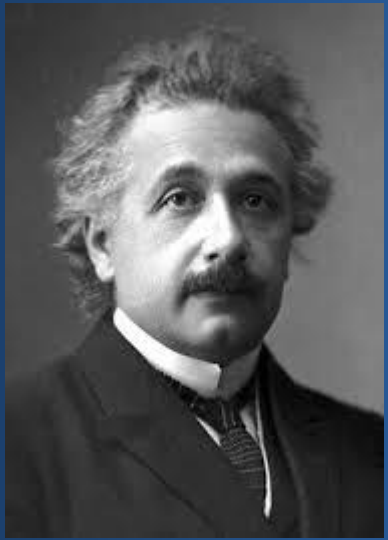
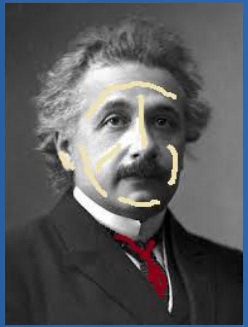
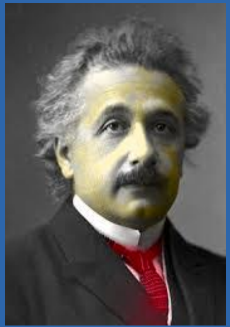
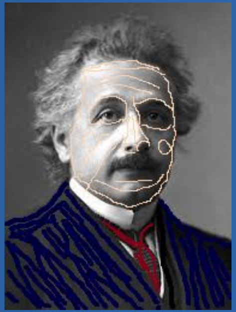
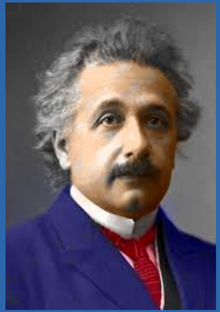
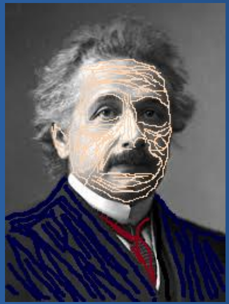
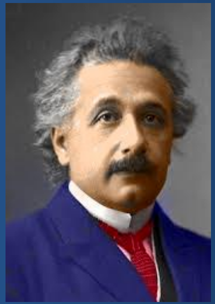
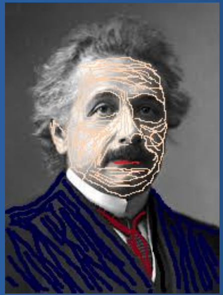
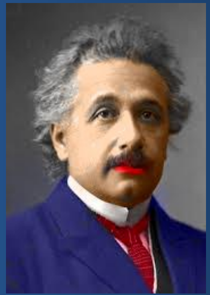




Results



Results (failed cases)

Results (Failed Cases)

Conclusion

- ✓ Colorization by Optimization problem by Levin was understood
  - ✓ Problem was formulated and solution was derived analytically
  - ✓ MATLAB code to implement formulation was developed
- Progressive coloring provides better results
- Coloring close to region boundaries produce better results
- Coloring large and very small regions not very effective
  - More than just a few scribbles required
- Approach not fully automatic
- Some initial goals did not provide expected results but was a learning experience
- Next Steps
  - Video!