



Image colorization using similar Images

Daniel Miller, Riza Hassan, Anjaney Mishra

Motive

- Get to colour WW2 images





Interactive Colourization

- Photoshop, MS Paint, GIMP, etc...

Weakness:

- Requires User to manually map color to each pixel/region. Requires a lot of effort.
- Quality dependent on the skill of user: Colourization may not be accurate enough, colour smudges.



Automatic Colourization

- Take a reference image as input and transfer colour to a target image. Generally Require the use of Neural Networks and large amounts of training sets.

Weakness:

- Requires a huge data set to train from.
- GPU, CPU and Memory consumption is high: Speed and Accuracy of the colourization is dependent of the length of training set and number of layers in the Neural Network.
- Generally, lower layers and training set give as fast program but reduce Accuracy. Getting a higher Accuracy program results in a slower program.



State of the art

State of the art colouring techniques use Convolutional Neural Networks and Deep Learning with huge datasets.



Problem

- Image Colourization is underconstrained, many colours can be assigned to gray pixels of an input image.
- The Automatic Method described need a ton of computational power.
- Manual Colouring is tedious.



Solution

Transfer Colour from a single input image to a target image. This method requires the following:

- Two similar images: One being the coloured reference and the other a grayscale target image.
- Extraction of superpixels: a group of pixels sharing similar characteristics.
- Extract discriminative Gabor, SURF features along with Intensity and standard deviation features.
- Cascade Colour Matching
- Image Space Voting

(All of these are explained later in presentation)



Based On:

Paper: Image Colorization Using Similar Images

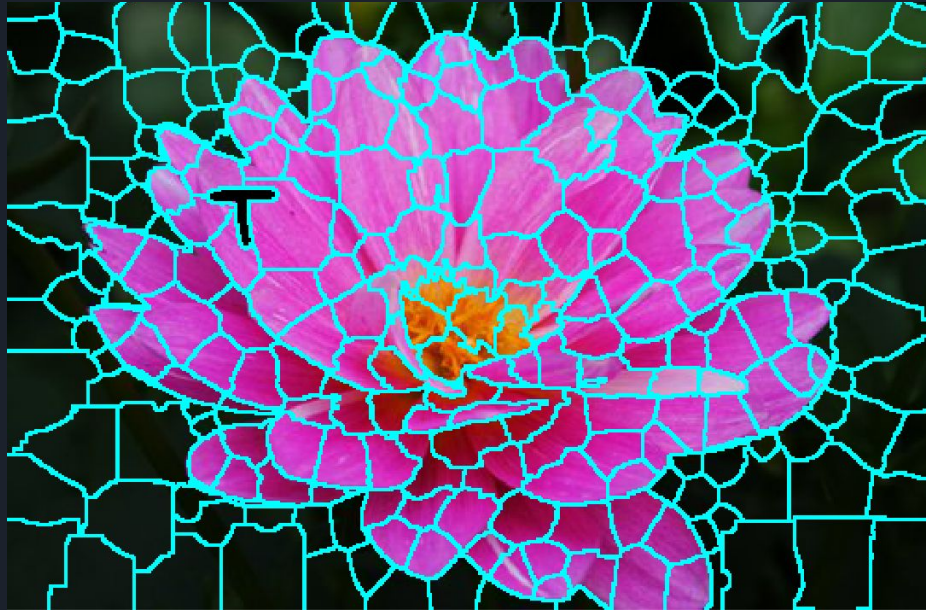
Citation: R. K. Gupta, A. Y. Chia, D. Rajan, E. S. Ng, and H. Zhiyong. Image Colorization Using Similar Images. <https://people.cs.clemson.edu/~jzwang/ustc13/mm2012/p369-gupta.pdf>



Setup

- From both images, extract 1000 (or any arbitrary) super pixels.
 - Extract 40 Gabor Features, 72 SURF, 2 Intensity and 2 Standard Deviation Features from Grayscale versions of both images
 - This gives 172 element vector per super pixel
-
- We use super pixels as they are faster to work with.

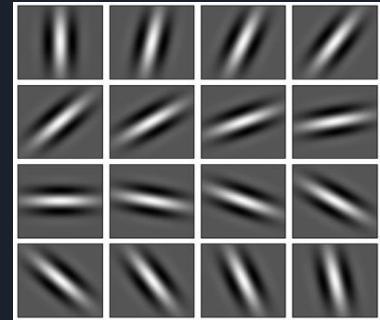
Supapixel Extraction



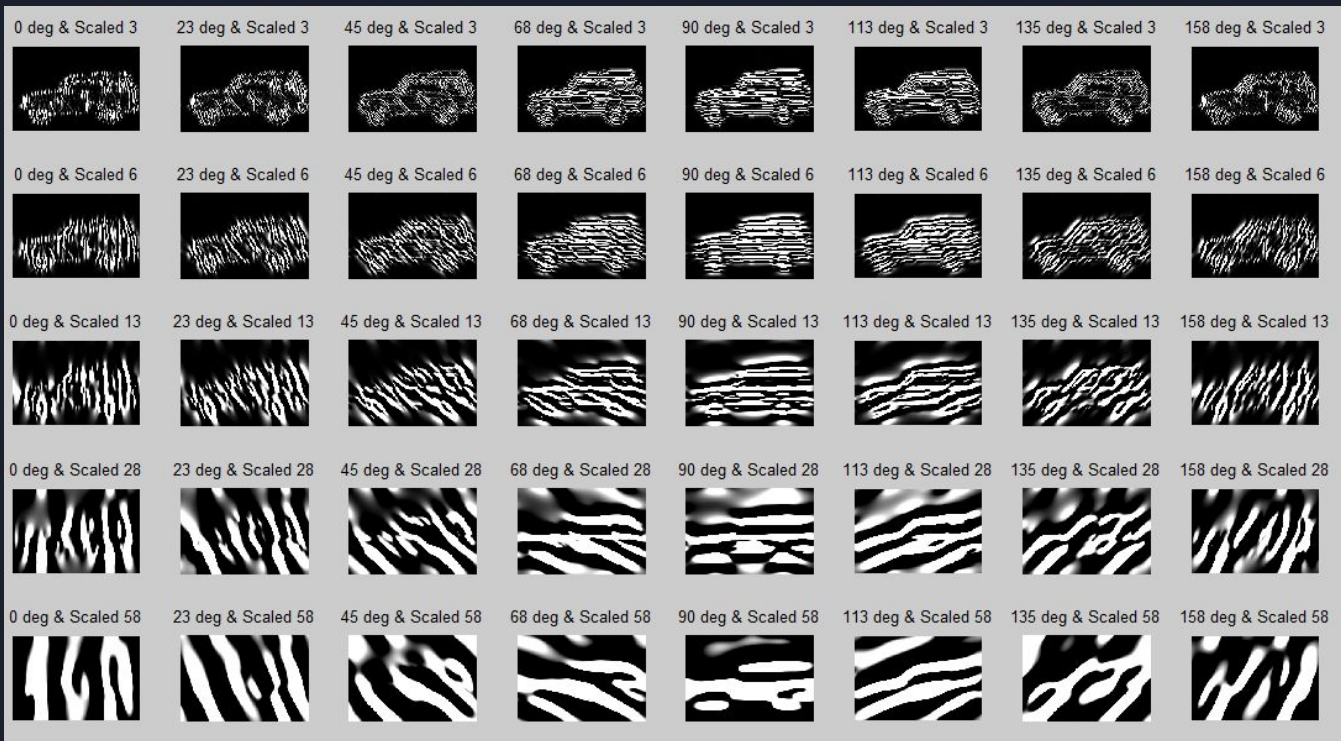
Gabor Feature:

Gabor Filters are used for Texture analysis: Look for specific frequencies in specific direction and orientation of a particular region.

In our case, apply a filter with 8 orientations (from 0 to $7\pi/8$, of increments $\pi/8$), and use 5 exponential scales ($\exp(i \times \pi)$, i in $[0,4]$). This gives us 40 gabor features per super pixel.



Gabor Features: (this is just an example)





SURF

- In Part Inspired from SIFT, but faster and more robust
- Surf uses square filters as an approximation of Gaussian, and uses a Blob Detector to find points of interest
- A Blob is a region of an image with approximately constant characteristics.

- SURF is chosen for its excellent discriminative ability and compactness.



Cascade Matching Scheme

Idea: Given a target super pixel, Converge on the right super pixel from the reference image using Gabor, SURF and other features. (Like an upside down pyramid.)



First Level

Algorithm:

Let T be a target super pixel: For each super pixel R in the reference image

- Compute Euclidean Distance between Gabor Features
- If Difference is less than error epsilon, add it to a list indexes

We took the mean gabor features for each super pixels, and computed the euclidean distance.

Our current epsilon value is 0.5, we found this by experimenting. Too large gives way to many values and too little gives no value.




Second Level

Same as previous, but with SURF features. Only difference this time is that we start with the index generated from previous step, and find half as many better matching super pixels.

The third and fourth level follow the same except with intensity and standard deviation features.

The order of features don't matter much for Gabor and SURF. They can be swapped.



Final step

We might not necessary get a single super pixel match, we might have multiple. We find the right matching super pixel using the following function:

- $\text{Arg min } F(r,t)$, where r is the set of reference super pixels and t is target
- $F(r,t) = w_1 * \text{dist}(r_g, t_g) + w_2 * \text{dist}(r_s, t_s) + w_3 * \text{dist}(r_i, t_i) + w_4 * \text{dist}(r_d, t_d)$
- Where w_i is a weight and dist is euclidean dist between Gabor, SURF, Intensity and Standard Dev feature.
- Acc to paper, $w = [0.2, 0.5, 0.2, 0.1]$, but we are experimenting with our own weights

Colour Mapping

Micro Scribbles: Takes a small area of an image and spreads it across. Kind of like interpolation

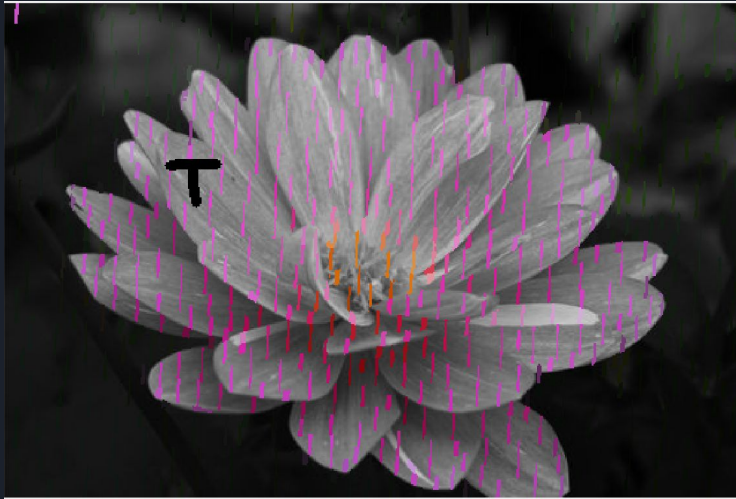


Image Space Voting

Pros

- Improving color assignments by voting for the color assignments in the image space.
- Exploiting neighboring superpixels to identify and to correct invalid color assignments.

Methods

- Cluster each image segments to have similar image properties with k-means clustering.



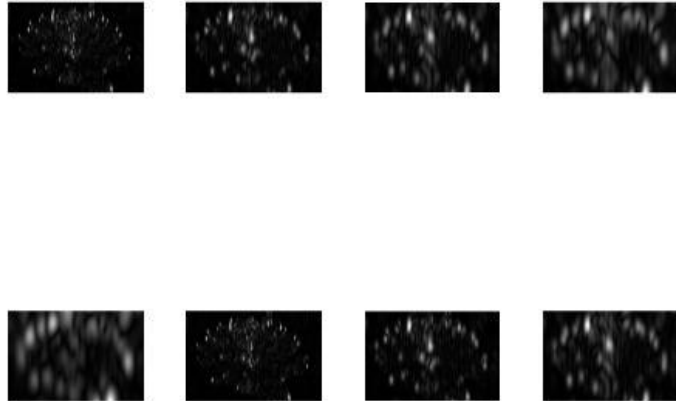


Our Progress

- Implemented Gabor Feature Mapping and Cascade level
- Currently working on SURF feature mapping

Results

Extracting Gabor Features:



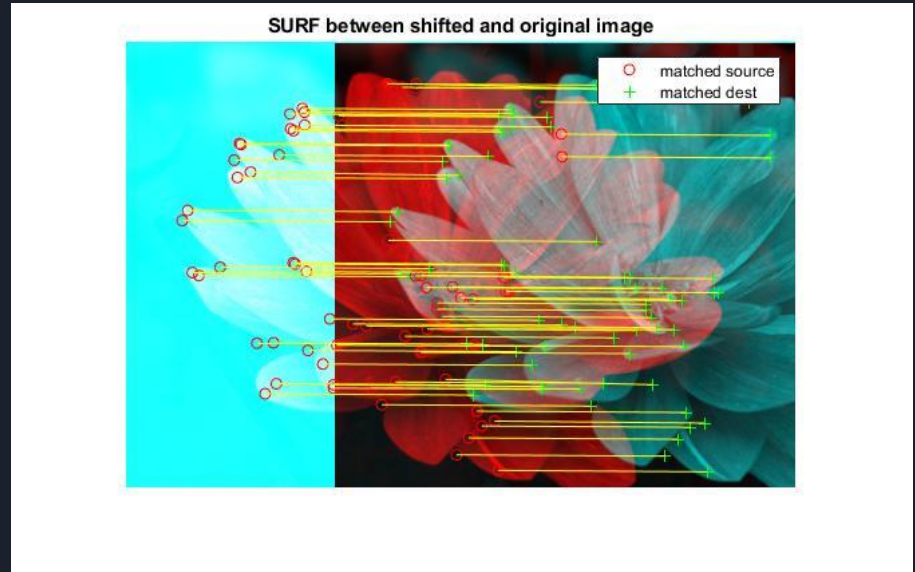
Results

SURF Mapping:

Note we did this part incorrectly by

Getting SURF for whole image,

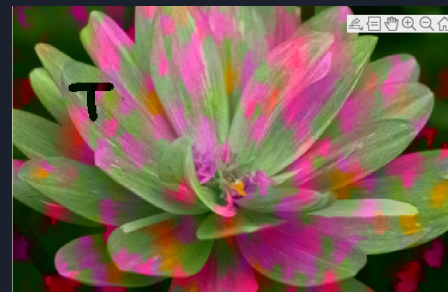
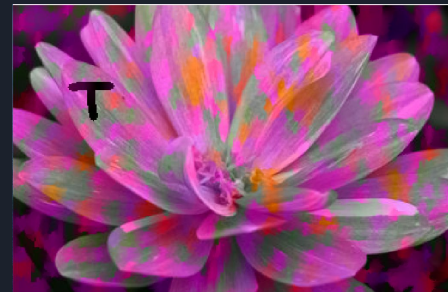
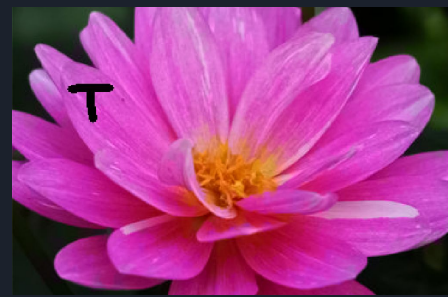
We need to get it for each superpixel



Reference

Target

Colorized Result





Reference



Target





Discussion

- One can see why using Gabor features isn't enough. Need more discriminative features to map
- The first transfer looks perfect, but that is because we colored the same image, so super pixels are exactly the same.
- In Application, one wouldn't need to color the same image using itself.
- The colour map is heavily dependent on the reference image. This is where Deep Learning works better as it uses a vast collection of references. This also relates to the underconstraint-ness of image colourization



Pros/Cons

Pros:

- A lot less data intensive than Neural Networks (i.e. doesn't need the use of implemented libraries with pre-trained data sets).
- Conceptually a lot easier to understand
- Can be done locally without needing big data sets to train
- Takes < min to run

Cons:

- Harder to implement than a Neural Network
- At technical level, a lot more math than a Neural Network
- Color is dependent on the reference image



Challenges

We Initially thought this was gonna be easy:

- We recently overcame a challenge we were facing with mapping Gabor features per super-pixel.
- Mapping colour from image to another is tricky, we cannot just transfer the rgb channels. We tried this and it basically just copied a part of image and put it in another place.
- Matlab superpixel extraction was unconventional.
- We might skip Intensity and Standard dev features as they are weaker than SURF and Gabor.
- SURF features don't quite work for super-pixels



Future Work

- Implement a better version of our colour mapper.
- Add more layers of features to get more accurate results
- Implement Image Space Voting
- Experiment with Higher number of super pixels
- SURF extraction (Currently trying to get it to work with SIFT)