

Automatic Wall Painter

Tal Mor

Abstract

We would like to create an automatic system that given an image of a room and a color, will color the room walls in that color while maintaining the original texture and shading of the walls.

The challenge is segmenting the image to identify which parts of it belongs to the walls and color them and them only.

In this project we will research the use of segmentation only for the identification of the walls. We will not experiment with Edge Detection or Shape Inference of any kind.

Introduction

In order to separate a wall segment of an image, we will have to explore ways to differentiate the wall from the rest of the image. The most trivial attribute to use would be the color. The more complex way would be to use the attributes of the texture.

In this project we will use the Laws texture energy measurements to try and separate the walls textures from the rest of the image. We will use 4 images as examples and experiment with various combinations of filters in order to find the optimum parameter s for segmenting the image into 2 parts: Wall and non-Wall.

We will use the K-means clustering method to divide the image into 2 segments. We will take into account many features of the image (textural and color) in order to cluster the image.

The code used in this project was completely written from scratch. No external code segments or functions were used but only inspired from.

Image Segmentation

There are many ways to segment an image, some of the most common ones are:

- 1) Thresholding
- 2) Graph Cuts
- 3) Clustering
- 4) Graph Merging and Splitting

The choice to use clustering was mainly since the number of segments is known in advance. It is also a very common method and was previously explored by many. Another reason was that clustering originates from initial centroids that can be picked by the user. Statistically, it should not be difficult to guess a pixel in an image where a wall exists and this should improve the performance of the algorithm.

Laws Texture Energy

We would like to differentiate one pixel from another in the image. Aside from the color, we can examine the pixel's texture (by using its neighbors).

Kenneth Ivan Laws developed a simple technique using filters and convolutions to extract texture features from an image. The information is taken from the paper "Texture Image Segmentation" [2].

We use 5 types of 2D convolution kernels:

L5 = [1 4 6 4 1] - Level
E5 = [-1 -2 0 2 1] - Edge
S5 = [-1 0 2 0 -1] - Spot
W5 = [-1 2 0 -2 1] - Wave
R5 = [1 -4 6 -4 1] - Ripple

These 5 filters are then converted to 25 3D filters by multiplying each couple together.

The generated 3D filters are then separately convolved with the image to produce 25 layers of the image, each representing a single texture attribute. For each of the resulting layers, 8 neighborhood cells are summed for each pixel to generate the texture energy measure for it.

My Experience

While working on this project, I was constantly thinking of additions and improvements that might result in better performance, trial and error was a significant part of the work.

Here are the stages I experienced when developing this application by chronological order:

- 1) Segmenting only one channel (R / G or B) without texture filters (as a grayscale image) .
- 2) Segmenting multiple channels and 1 Laws filter. (Experiencing with different filters)
- 3) Segmenting multiple channels and multiple filters. (applying the filter on a grayscale conversion of the color image)
- 4) Add color differentiation.
- 5) Move to Histogram representation for faster processing. (eliminate similar values)
- 6) Use more matrix operations than loops to improve performance.
- 7) Convert to layers and cluster separately for each filter.
- 8) Compute color differentiation using hsv rather than rgb.

Algorithm

The algorithm is divided to a number of steps:

- 1) Preprocessing
 - a. Eliminate negative and zero values in the image
 - b. Create 1-3 layers of the image from the color channels
 - c. Create 1-25 layers of the image by applying laws filters to the grayscale conversion of the image
 - d. Eliminate fractional values from the image to allow histogram processing.
 - e. Return the image with 1-28 layers after preprocessing
- 2) Clustering
 - a. Compute initial centroids.
 - b. Compute initial clusters.
 - c. Cluster each layer repeatedly until the centroids stabilize.
- 3) Post Processing
 - a. Combine the clustering of the different layers by placing a pixel in its most frequent cluster among the layers.
- 4) Wall Identification
 - a. Simple identification by statistical wall and ceiling position in an image.
- 5) Wall Color
 - a. Convert image to HSV representation.
 - b. Color the wall in the given color (by changing hue for the relevant pixels)
 - c. Convert back to RGB.

Results

We will examine the results on 4 images using 3 sets of parameters.

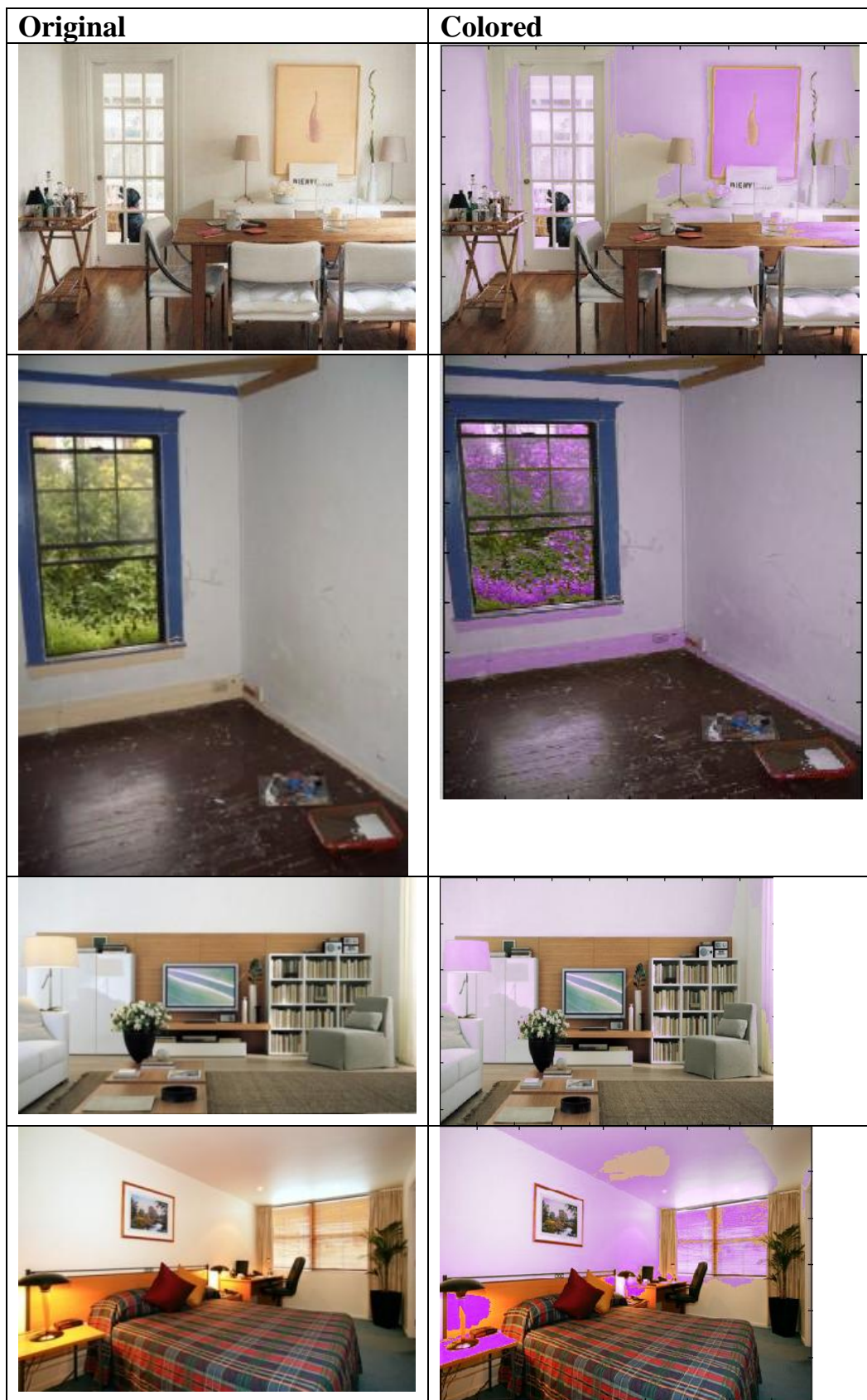
The fixed parameters we will use is:

- 1) Max number of iterations for k-means clustering is 1000.
- 2) Centroid positions will be considered alike when their difference is smaller than 0.05.
- 3) Filters will be summed by convolving a 3x3 ones matrix.
- 4) Clustering will be done with 2 clusters
- 5) Pixel will belong to a wall if it is more than 90% closer to it.



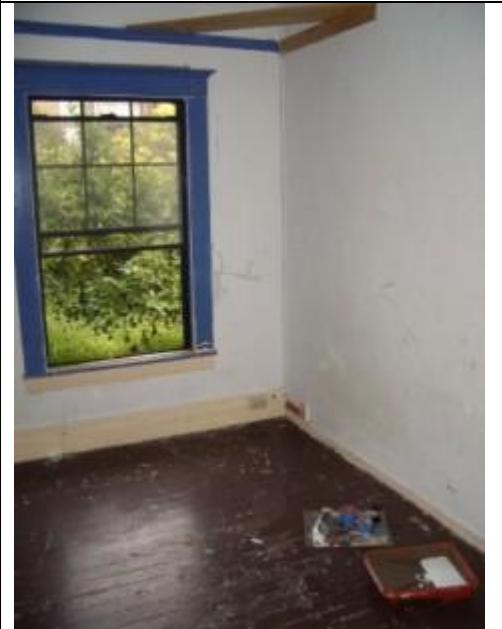





The dynamic parameters are:

- 1) Which image to process.
- 2) Which filters to use – Array of numbers between 1-25.
- 3) Which channels to use (RGB) – Array of numbers between 1-3
- 4) Which channels to use (HSV) – Array of numbers between 1-3
- 5) What color to use – provide Hue value between 0-1
- 6) What brightness and saturation to add – value between 0-1.



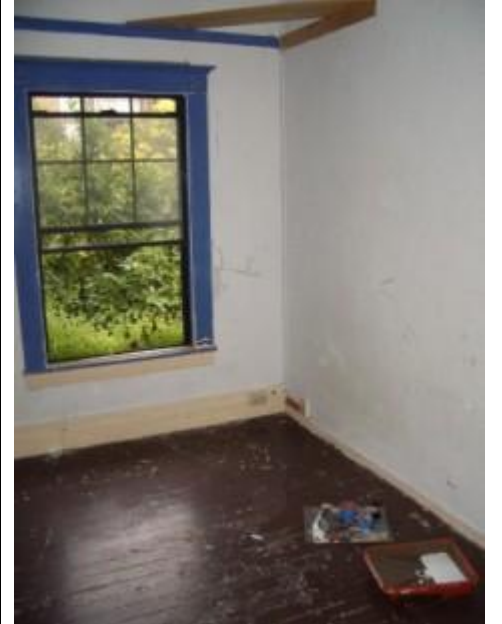





1) Parameters - $[, [1\ 2\ 3],]$ – Color 0.8 Brightness 0.05



2) Parameters - [2 3 6 11],[2 3],[1 2 3] – Color 0.8 Brightness 0.05

Original	Colored
 A dining room with a wooden table, white chairs, a side table with bottles, and a window with a view of the outdoors. A framed picture hangs on the wall.	 The same dining room scene, but with a color cast of purple and blue. The walls and floor appear to have a textured, almost crystalline appearance.
 An empty room with a large window showing green foliage outside. The room has white walls and a dark wooden floor.	 The same empty room, but with a color cast of purple and blue. The window frame and the floor appear to have a textured, almost crystalline appearance.
 A living room with a white sofa, a television, a bookshelf, and a coffee table. A vase of flowers sits on the coffee table.	 The same living room scene, but with a color cast of purple and blue. The walls and floor appear to have a textured, almost crystalline appearance.
 A bedroom with a bed covered in a plaid blanket, a desk with a chair, and a window with blinds. A framed picture hangs on the wall.	 The same bedroom scene, but with a color cast of purple and blue. The walls and floor appear to have a textured, almost crystalline appearance.

3) Parameters - [],[],[2 3] – Color 0.8 Brightness 0.05

Original	Colored
 A dining room with a wooden table, white chairs, a side table with bottles, and a framed picture on the wall.	 The same dining room scene, but with a strong magenta color cast applied to the entire image.
 An empty room with a window showing green foliage outside, a dark floor, and a small table with items on it.	 The same room scene, but with a strong magenta color cast applied to the entire image.
 A living room with a white sofa, a coffee table, a TV, a bookshelf, and a lamp.	 The same living room scene, but with a strong magenta color cast applied to the entire image.
 A bedroom with a bed covered in a plaid blanket, a desk with a chair, and a window with blinds.	 The same bedroom scene, but with a strong magenta color cast applied to the entire image.

Optimizations

The average runtime for a 400x400 pixel image is 40 seconds, the following optimizations were implemented to reach this:

- 1) Using only specific filters and channels for the clustering.
- 2) Minimizing the use of loops where possible and moving to matrix operations.
- 3) Use Histogram representation to eliminate similar values and compute bary centers quicker.
- 4) Normalize matrices to work with lower numbers.
- 5) Matrix binning of the filter results to work with a smaller amount of values in the histogram.

Conclusion

- Color differentiation is not enough to isolate a wall as shading and similar colored objects are hard to distinguish.
- The difference between clustering R/G/B channel is not big.
- The hue factor is very similar in most areas of real room images. This can originate from room's lighting color temperature or just pure color matching designs.
- Most of Laws texture filters are too grainy to work with on real life images, some filters can indeed isolate one part of the room from the other but will still leave grainy spots.
- Different images perform differently on laws filters therefore maybe a dynamic parameter choosing algorithm should be explored during the preprocessing stage of the image.
- Run time is a crucial factor in such programs and should be optimized from the start.
- It appears that segmenting by color and texture is not enough to complete this task correctly , in order to achieve more accurate results , edge detection and shape inference are most probably necessary.

Future Improvements

- 1) Use HSV Space to measure the proximity of colors. Combine color channels to compute color distance in a 3D color domain.
- 2) Experiment with larger texture filters (larger than 5x5)
- 3) Use graph cut clustering to speed up the clustering process.
- 4) Segment to more than 2 clusters and use relaxation labeling to identify the wall segments.
- 5) Use simple 3D shape inference to identify straight lines that can outline a wall (edge detection)

References

[1] Wall, Floor, Ceiling, Object Region Identification from Single Image, Zhong-Ju Zhang, Stanford University

[2] K. Laws. *Textured Image Segmentation*, Ph.D. Dissertation, University of Southern California, January 1980.

[3] <http://www.ccs3.lanl.gov/~kelly/ZTRANSITION/notebook/laws.shtml>