CS445 Final Project - Painting Walls in 3D

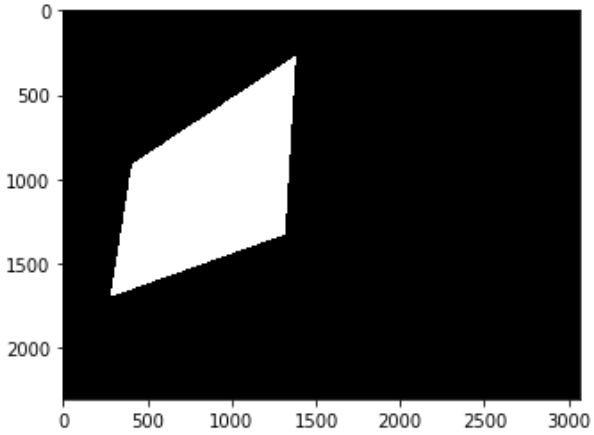
by Yeonju Kim (yeonjuk2) and Mohit Vyas (mohitv2)

Goal: paint graffiti on walls by taking into account their 3D structure as shown in the example below.

Approach Overview:

1. Obtain 3D point cloud by using COLMAP's [2] implementation of Structure from Motion (We used sample images from the dataset [1])
    - Take several pictures from different positions and angles.
    - Using the COLMAP library, get a sparse point cloud from images.

2. Specify the plane of the wall using a manually annotated mask. Then, use RANSAC to obtain the equation of the required plane in 3D.
    - Specify the required plane by using a mask for its corresponding pixels in the image.
    - From the sparse point cloud, we can get the 3D coordinates of the points inside the mask.



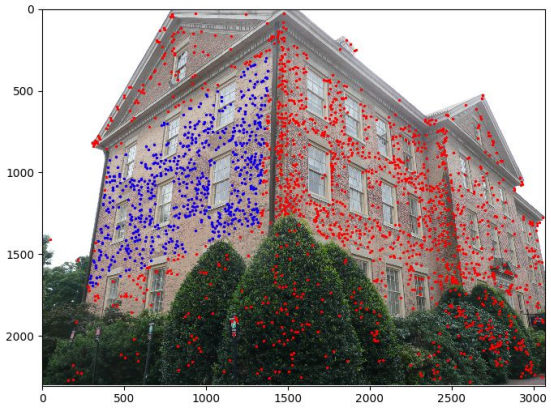
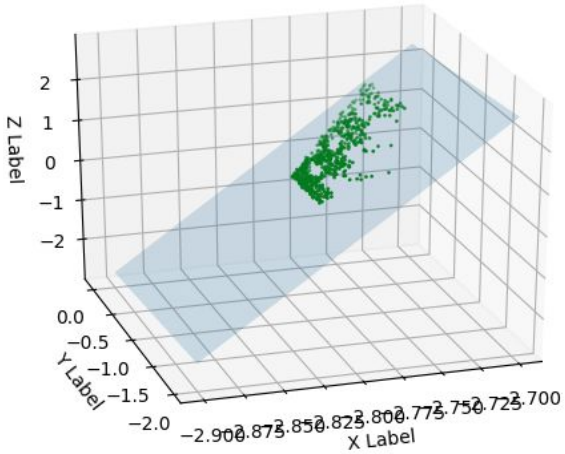


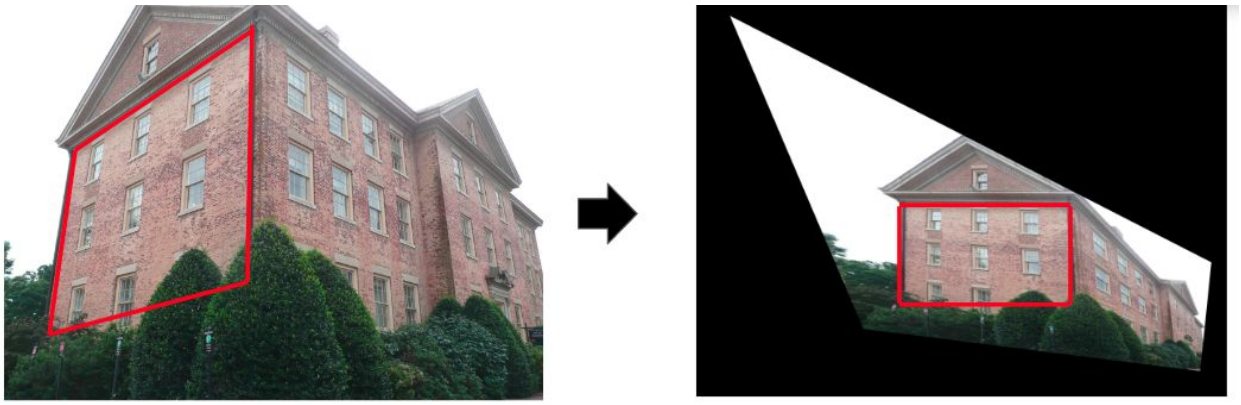
FIg1. mask
Fig2. Red & blue points: 2d points in sparse point cloud & blue points: masked points for finding a plane.

- Find the plane of the wall using RANSAC by repeating the following steps until sufficiently many points is on the plane.
    i. Select a random 3 points inside the mask and construct a plane equation which is fitted to the subset.
    ii. All other points are tested against the fitted plane. If the normal distance from the plane is smaller than threshold, it is considered as the point on the plane.



3. Obtain Projection Matrix
    - Derive the binormal and tangential unit vector from the normal vector.
    - Project 3d points into the plane we got from step 2 by using binormal and tangential vector as axes.
    - Obtain Projection matrix between the 2D coordinates of 3D points and the image plane. Used Singular Value Decomposition to compute Homography like Project-5.

4. Using the Projection matrix from step-2, warp the image such that the building plane is parallel to the image plane as shown in the example below.



5. Use graph-cut to segment out pixels that we need to repaint

6. Blend the new texture on the foreground pixels. We compare two blending methods: (i) Laplacian blending, (ii) Gradient-based domain fusion using mixed gradients from Project-3.

The above images show the result of laplacian blending and mixed-gradients based image blending of the drawing of a flower on top of the building.

7. Rotate and scale the new building face to obtain the final image with the painted wall.

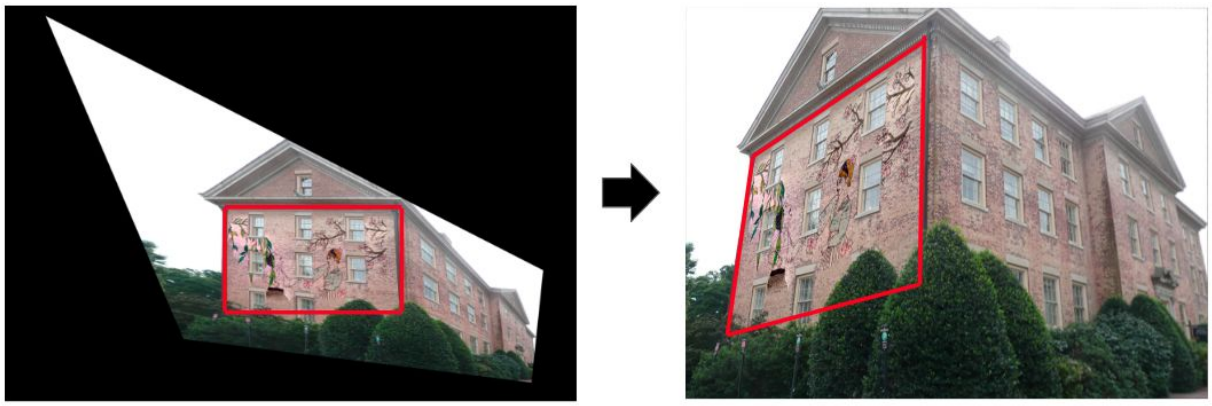
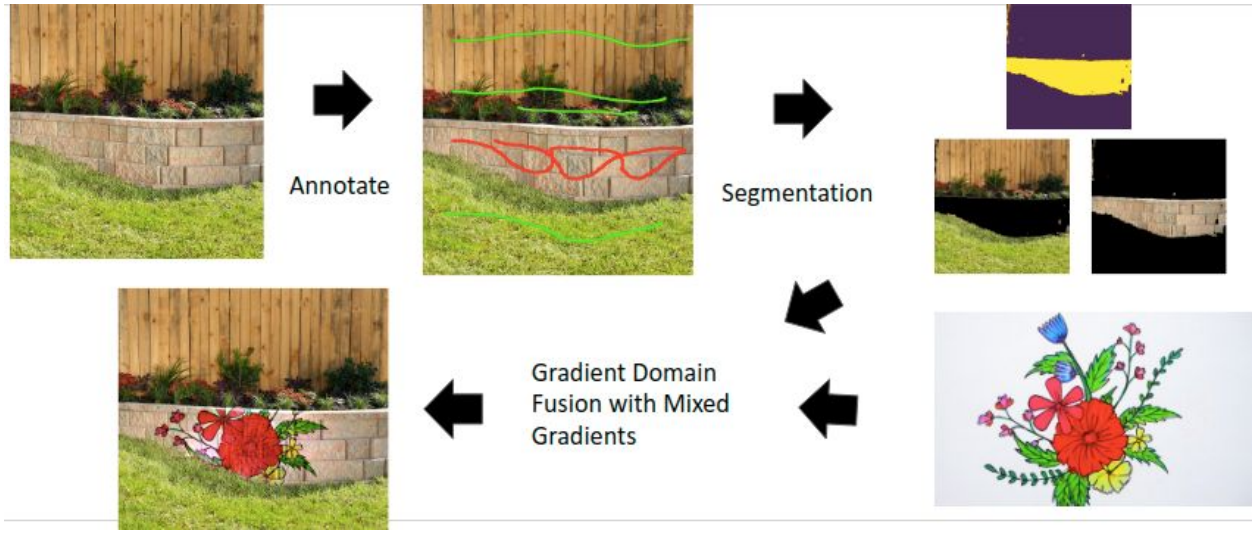
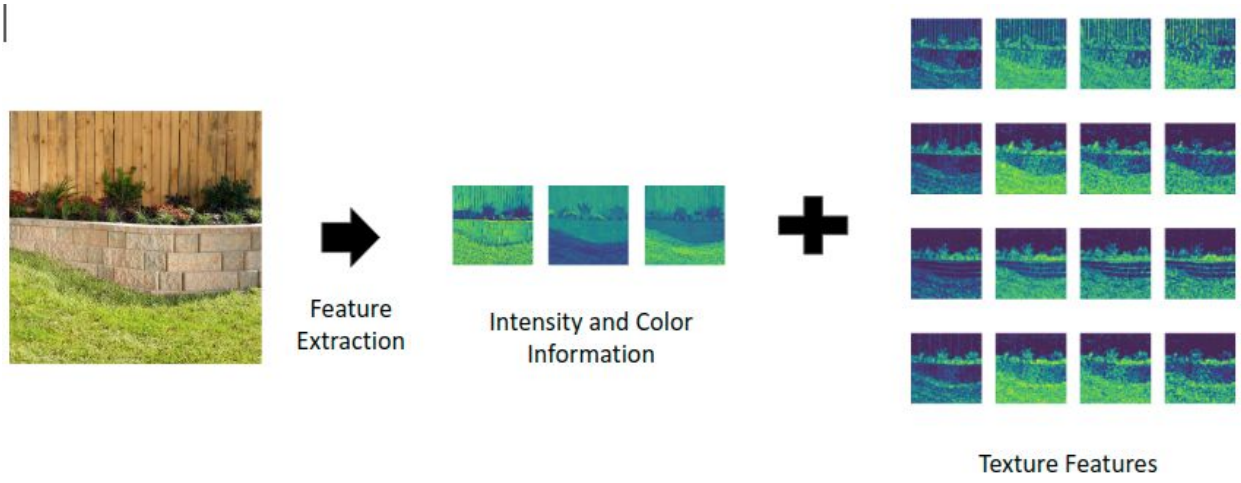


Image Segmentation: In this section, we discuss a few more details of the image segmentation part. The overall pipeline for wall painting is as follows:

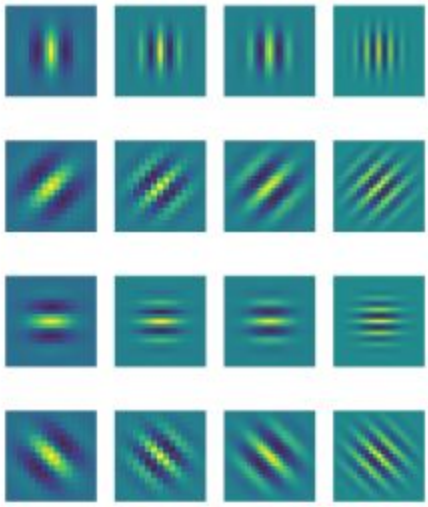


We first manually annotated regions containing foreground and background pixels. These are used to build models for foreground and background pixels which will be used in the segmentation of the required region in the image. Image segmentation works as follows:

1. Obtain various features for all pixels that might help us distinguish between foreground and background pixels. We consider the following features: (i) Three channels of “Lab“ color space representation of the image, (ii) local texture information in the form of windowed average response for Gabor filters [3] of varying standard deviations, frequencies of harmonic function and angles at which the harmonic function is applied on top of gaussian filter. Gabor filters are implemented using scipy’s implementation [4]. The below example shows how these features can be used to distinguish between foreground and background pixels.

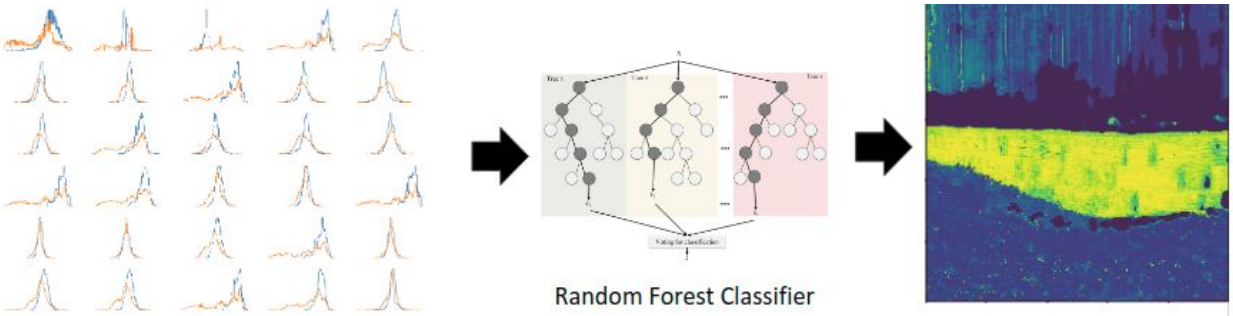


Note how the response to different filters at any given pixel gives us an indication of whether the pixel is part of the foreground or background.



Gabor filters as combinations of Gaussian filters with harmonic filters at different angles and standard deviations.

2. Use these features to build a binary classifier of foreground and background pixels. We use random forest classifiers [5], using its scikit-learn implementation [6].

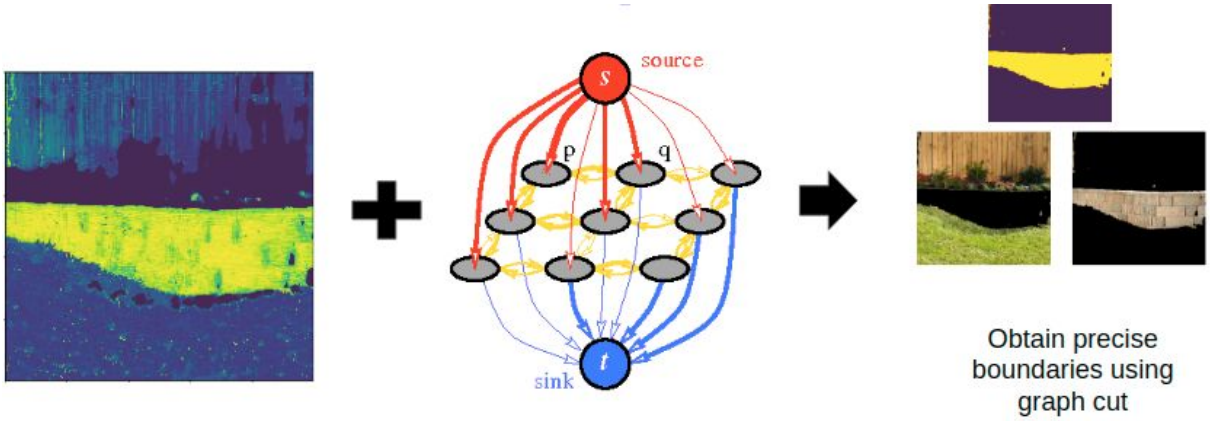


Response of foreground and background pixels for different features.

Probability Density Map for Foreground Pixels

This classifier gives us the probability "$P_{fg}(p)$" for any pixel "p" being part of the foreground. As we can see from the example, this itself gives us a good idea of where the foreground pixels are.

3. Next, we use the graph-cut algorithm to obtain precise boundaries to identify the regions of foreground pixels.



There are three kinds of edges in this graph, (i) source to pixel terminal edge, (ii) pixel to sink terminal edge and (iii) pixel to pixel non-terminal edge. The capacities of its edges are defined as follows.

a.) If a pixel “p” is labeled as foreground, then the terminal edge from source to “p” is assigned capacity infinite and capacity for “p” to sink terminal edge is “0”. This is reversed in case “p” is labeled background.
b.) If pixel “p” has the unknown label, capacity for source to “p” is “$-\log(P_{fg}(p))$” and for “p” to sink it is, “$-\log(1-P_{fg}(p))$”.
c.) For every pair of 4-neighbor pixels *p* and *p’,* there are edges from *p* to *p’* and *p’* to *p* with the same capacity of “$\lambda exp((f_i(p)-f_i(p'))^2/(2\sigma_i^2)$”. Here “$f_i$” is i^{th} feature and “σ_i” is the standard deviation of “f_i” in a window around “p”. We use three channels ‘L’, ‘a’ and ‘b’ as three features for defining these capacities. The intuition is that if the intensity of color change drastically then the penalty for assign p and p’ different labels should be small.

Calculation of min-cut and max-flow of the graph is computed using maxflow library [7] in python. OpenCV [8] and general libraries in python (numpy, matplot lib, etc.)[9] are used extensively throughout the code to manipulate images. Finally, the code for laplacian blending and mixed gradients based gradient domain fusion is taken from the submission of the project-3.

Results

1. Result1

Original image

Source image

Laplacian Blending

Gradient Domain Fusion using Mixed Gradients

2. Result 2

Original Image

Source Image

Laplacian Blending

Gradient Domain Fusion using Mixed Gradients

3. Result-3 (segmentation only):

Major Challenges:

- The model for foreground and background pixels should be pretty good for the graph cut to work well.
- Hard to obtain proper Plane equation and Projection Matrix due to the noise of the sparse point cloud.
- A drawback of mixed gradient domain fusion: the background color of the painting should be close to the average color of the wall, else it produces artifacts in the image.
- A lot of manual annotation is required to produce a working result.
- The full pipeline takes a lot of time to execute and has high chances of failure.

Failure Cases:

1. Hard to obtain proper Projection Matrix:

Parallelogram instead of Rectangle. Like in project-5, the results of warpPerspective sometimes diverge.

2. Due to the noise in 3D point cloud, the obtained plane is different from the actual plane of the wall. This can be seen in the example below where the windows are parallelograms with angles different than 90 degrees, rather than rectangles.

3. Various effects like high-intensity gradients, reflections, refractions can throw off various parts of the pipeline including 3D modeling and segmentation. In the example below, high-intensity gradients make the model of foreground pixels less effective than needed. This results in poor boundaries.

4. Sometimes it is difficult to exclude parts of the image that are very similar to the foreground. In the example below, the part of the wall in the bottom right looks very similar to the wall we want to paint. However, as described by the annotation, we want to exclude this portion of the wall. But since the pixels are too similar, some pixels from this region make into the final mask anyways.

Unable to exclude pixels with similar texture as foreground from the segmentation

5. Trying to paste objects on a plane not parallel to the wall gives weird-looking results.

Outside Resources:

Dataset:

[1]https://onedrive.live.com/?authkey=%21AAQumsDDwZBIW3w&id=C58A258D760E1B58%2146879&cid=C58A258D760E1B58 (south-building.zip)

Code/Libraries:

[2]https://colmap.github.io/
[3]https://en.wikipedia.org/wiki/Gabor_filter
[4]https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_gabor.html
[5]https://en.wikipedia.org/wiki/Random_forest
[6]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[7]http://pmneila.github.io/PyMaxflow/maxflow.html
[8]https://pypi.org/project/opencv-python/
[9]python, matplotlib, numpy

Images:

https://images.app.goo.gl/PqqKY82T33CyDU1d6
https://images.app.goo.gl/GVU7wRtVzhQ49GyT9
https://images.app.goo.gl/CbDQcvNjhHFFaZu17
https://images.app.goo.gl/gRLgdNHdHpeujzP5A