Automatic Panoramic Image Stitching using Invariant Features

Team 8
Shreyash Jain 2020101006
Gautam Ghai 2020101020
Mayank Bhardwaj 2020101068

Introduction

- The paper presents the problem of automatic panoramic image stitching.
- Previous approaches have used human input or restrictions (single axis of rotation) on the image sequences for matching images.
- The paper implements several techniques such as match verification using probabilistic model, bundle adjustment, automatic straightening, gain compensation and multi-band blending to solve this problem.
- The system is robust to camera zoom, orientation of input images, and changes in illumination due to flash and exposure/aperture settings.

Methodology

Solution proposed by the paper

Algorithm: Automatic Panorama Stitching

Input: n unordered images

I. Extract SIFT features from all n images

II. Find k nearest-neighbours for each feature using a k-d tree

III. For each image:
   (i) Select m candidate matching images that have the most feature matches to this image
   (ii) Find geometrically consistent feature matches using RANSAC to solve for the homography between pairs of images
   (iii) Verify image matches using a probabilistic model

IV. Find connected components of image matches

V. For each connected component:
   (i) Perform bundle adjustment to solve for the rotation $\theta_1, \theta_2, \theta_3$ and focal length f of all cameras
   (ii) Render panorama using multi-band blending

Output: Panoramic image(s)

Feature matching and Homography Estimation

- First, we extract the features for all n images, and each feature is matched to its k nearest neighbours in feature space in O(nlogn) time by using a k-d tree to find approximate nearest neighbours.
- For one image we only consider its matches with the best m images based on the number of matches.
- RANSAC is used for Homography estimation by randomly sampling correspondences and selecting sets of r features to compute homography matrix H for n trials.
- For n = 500 and r = 4, the probability that H is correct is very high. (pi is the inlier probability)

$$p(\mathbf{H}\ is\ correct) = 1 - (1 - (p_i)^r)^n.$$

Match Verification
(Probabilistic Model)

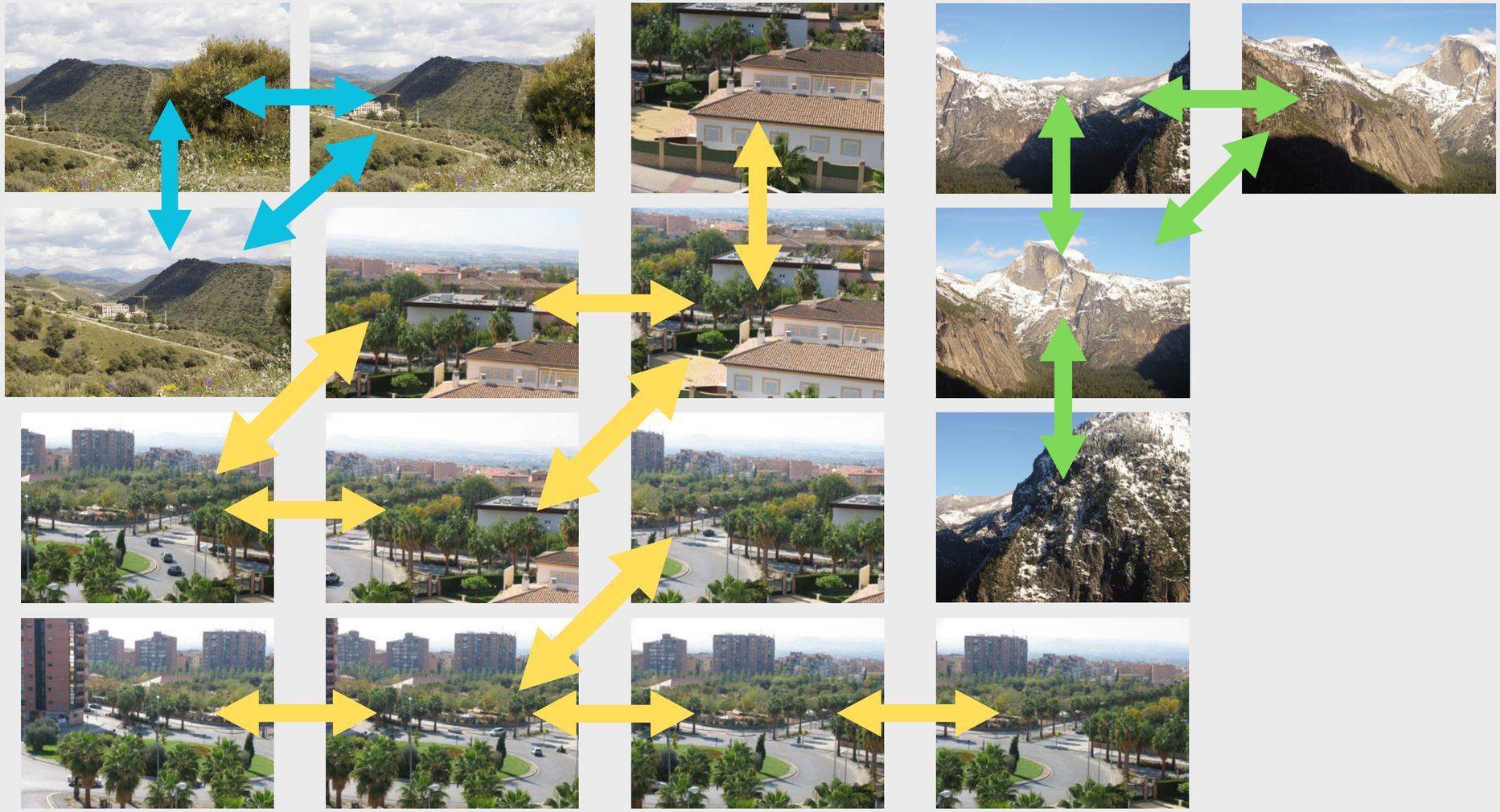
- We don’t know yet which pairs of images should indeed overlap in the final panorama, and which pairs actually don’t have any area in common.
- To verify the image matches, we compare the features matches that are consistent with the homography (RANSAC inliers) with the feature matches inside the area of overlap between the two images but not consistent (RANSAC outliers).

$$n_i > \alpha + \beta n_f$$

- Where,
  - n_i = Number of inliers
  - n_f = Number of features in the overlapping area
  - α, β = constants (8.0, 0.3)

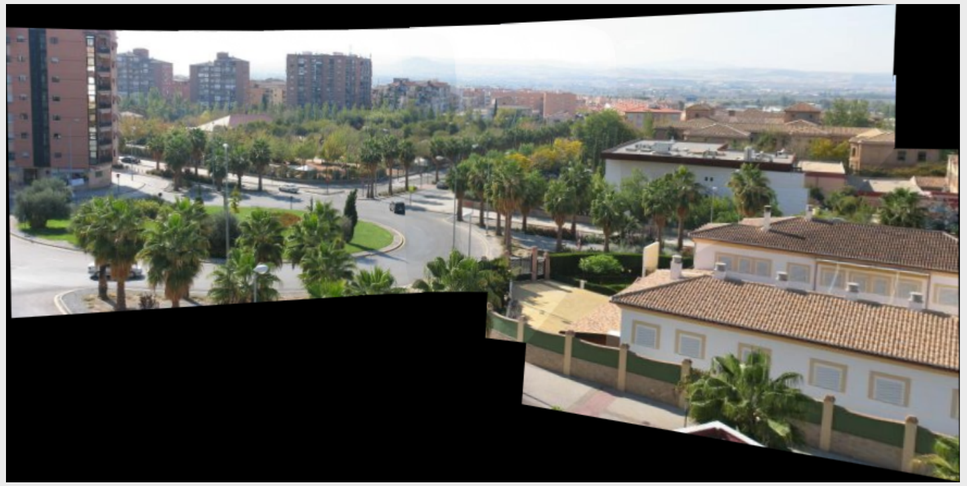
Find Connected Components

- We first divide all images into connected components based on their matches to one another.
- Consider a graph where each node is an image and the edge weights are the number of matches between two images. Each subgraph / connected component is treated as a separate panorama.



Identifying valid image matches

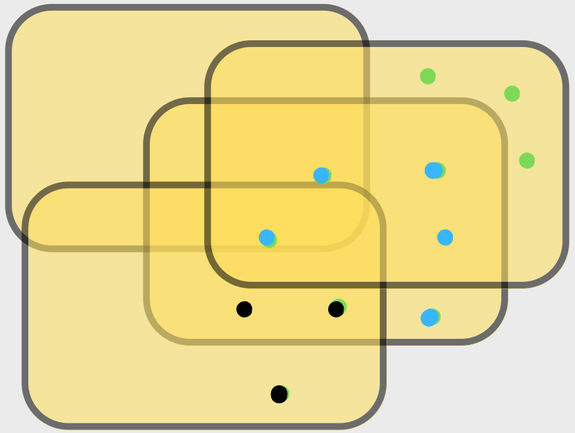
Extracting connected components



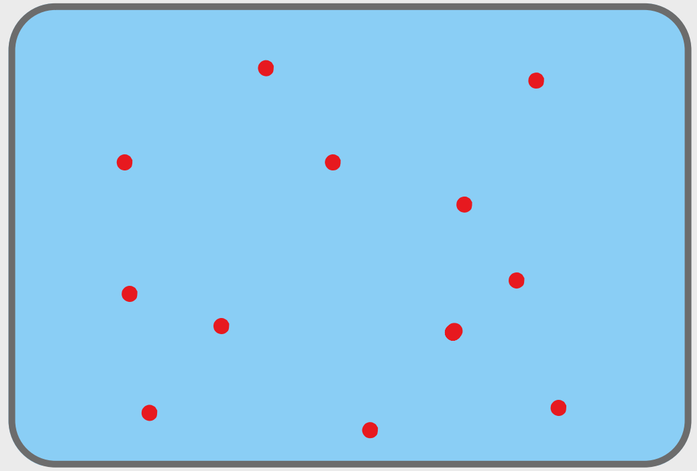
Final Panaromas

Bundle Adjustment

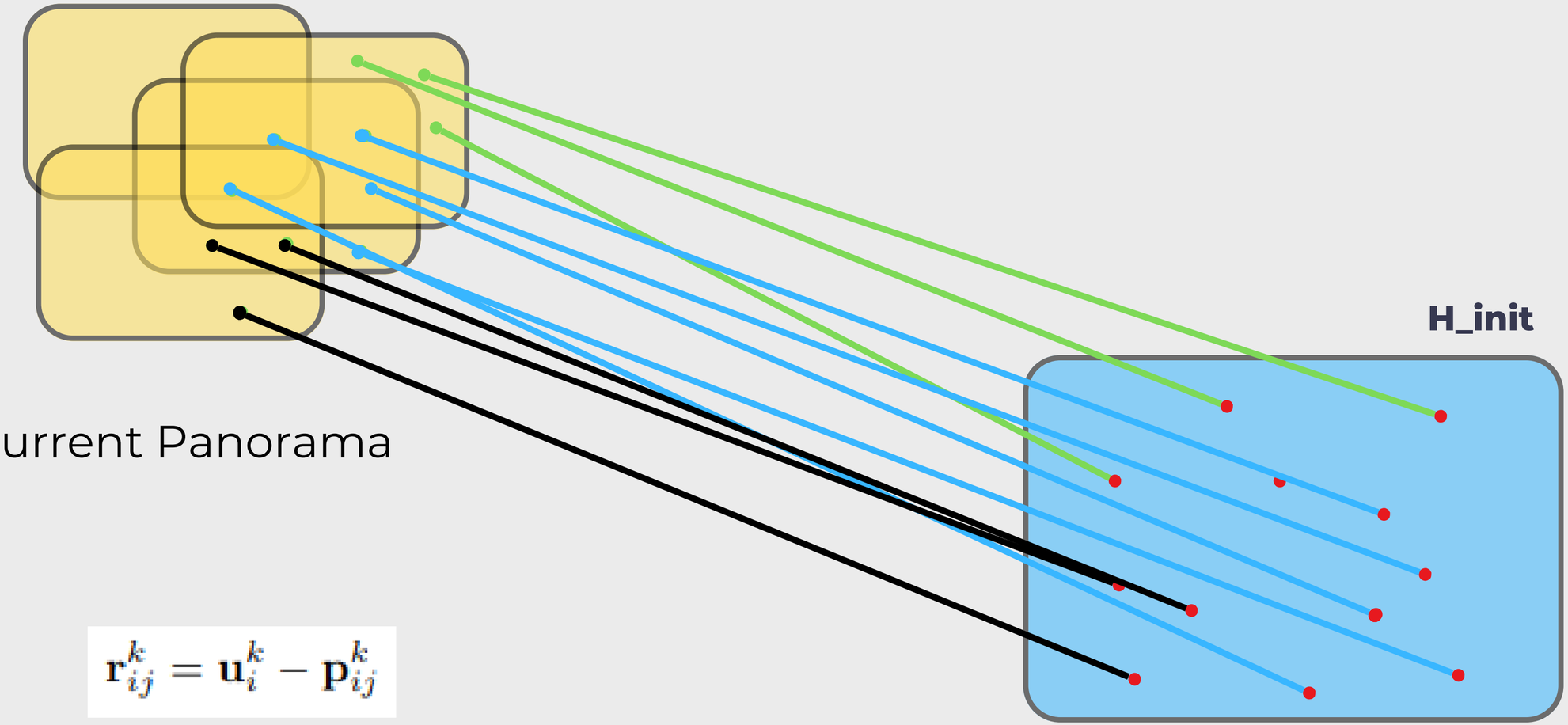
- Bundle adjustment is an essential step as the concatenation of pairwise homographies will lead to accumulated errors and disregard constraints between images
- We send in the images in a particular sequence and based on the original homography estimates we try to reduce the projection error of the image on the panorama by comparing its Euclidean distance from the matching feature points of images already in the panorama.
- This minimization is done with the help of the LM algorithm.



Current Panorama

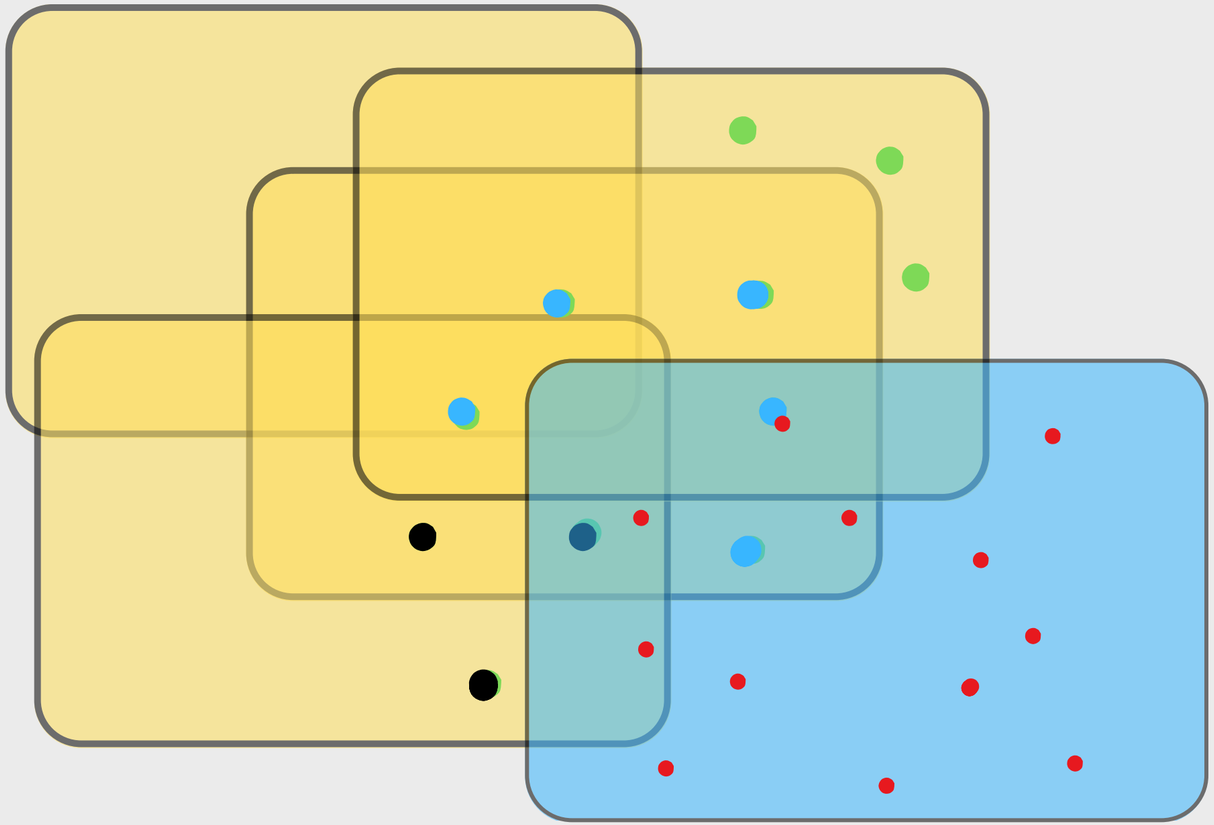


Current Image

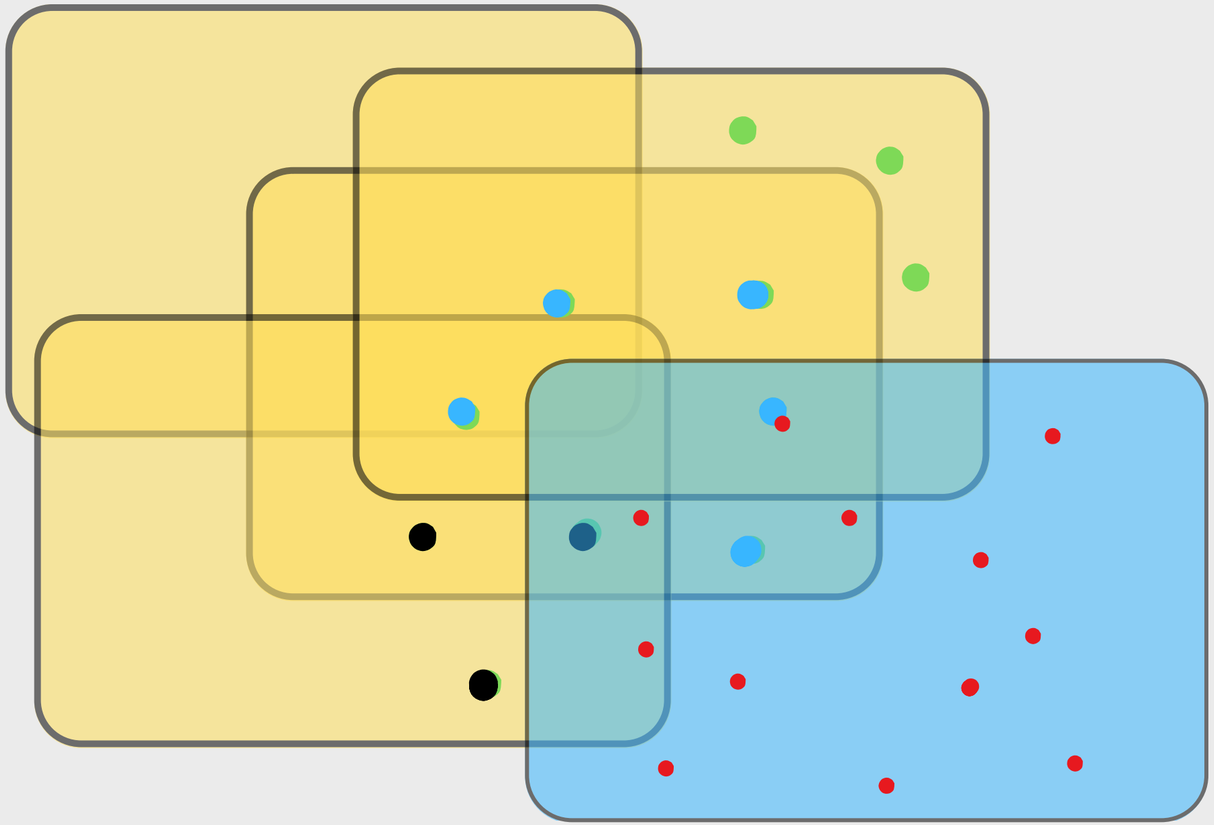


$$\mathbf{r}_{ij}^{k} = \mathbf{u}_{i}^{k} - \mathbf{p}_{ij}^{k}$$

$$e = \sum_{i=1}^{n} \sum_{j \epsilon \mathcal{I}(i)} \sum_{k \epsilon \mathcal{F}(i,j)} h(\mathbf{r}_{ij}^{k})$$



Updated Panorama



Updated Panorama

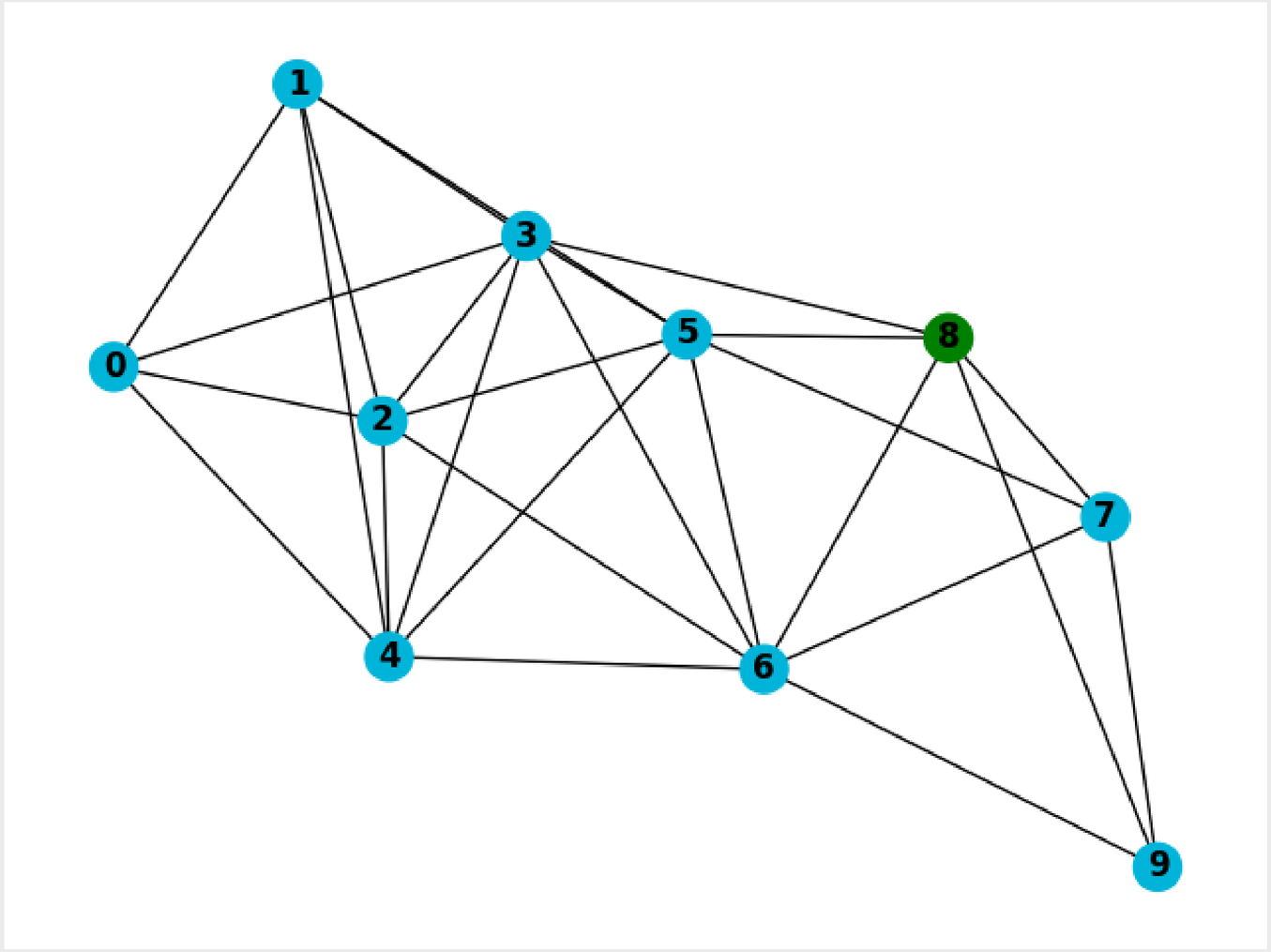
We are only storing the feature points for now!

Setting Initial Sequence

- To determine the initial sequence of images to pass into the panorama, we make a minimum spanning tree of the network of all images in which the edge weights are inversely proportional to the number of matches between two images/nodes.
- This ensures that the MST will have all images which have a high number of matches (less edge weight).
- The source/start point of the sequence is determined by three methods:
  - The node with the highest degree (most matching images)
  - The node with the highest sum of matches with all other images
  - Any random node

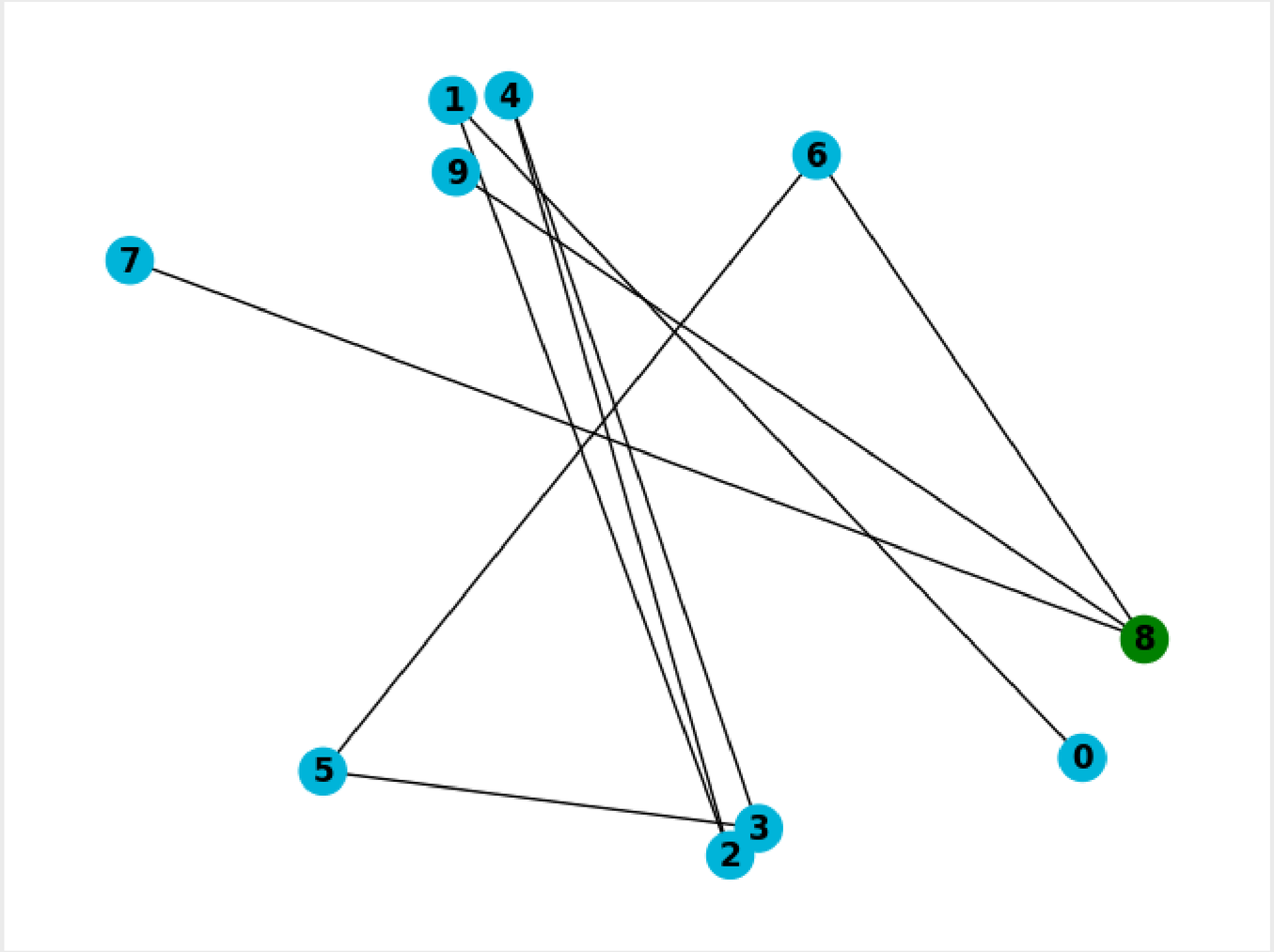
Setting Initial Sequence

- This start point affects the algorithm a lot because all images in the panorama are transformed into the frame of this source image.
- We then get the sequence starting from the source by performing a simple BFS in the MST.
- The BFS approach ensures that when I add a new image, its most matching image(s) have already been added to the panorama which gives more feature points to compute error with in the LM minimization step.



Initial network

MST (highest degree source)



Path : 8,9,7,6,5,3,4,2,1,0

Gain compensation

- In the paper gain compensation is used to correct for differences in exposure between the images being stitched.
- It has been set up as an error function which is the sum of gain normalized intensity errors for all overlapping pixels.
- g_i and g_j are the gains and R (i, j) is the region of overlap between images i and j.
- We approximate I(u_i) by the mean in each overlapping region I_ij. This simplifies the computation and gives robustness to outliers.
- To solve this quadratic function, we set the derivative to 0, which gives: which can be solved using a Linear solver.

$$\bar{I}_{ij} = \frac{\sum_{\mathbf{u}_i \epsilon \mathcal{R}(i,j)} I_i(\mathbf{u}_i)}{\sum_{\mathbf{u}_i \epsilon \mathcal{R}(i,j)} 1}.$$

$$e = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{\substack{\mathbf{u}_i \epsilon \mathcal{R}(i,j) \\ \tilde{\mathbf{u}}_i = \mathbf{H}_{ij} \tilde{\mathbf{u}}_j}} (g_i I_i(\mathbf{u}_i) - g_j I_j(\mathbf{u}_j))^2$$

$$g_j \left(\frac{2}{\sigma_N^2} \sum_{i=1}^{n} N_{ji} \bar{I}_{ji}^2 + \frac{\sum_{i=1}^{n} N_{ki}}{\sigma_g^2} \right) - \frac{2}{\sigma_N^2} \sum_{i=1}^{n} N_{ki} \bar{I}_{ki} \bar{I}_{ik} g_i - \frac{\sum_{i=1}^{n} N_{ki}}{\sigma_g^2} = 0$$

Multi-Band Blending

- Some image edges are still visible due to a number of unmodelled effects and we use multi-band blending of Burt and Adelson to account for that.
- The idea behind multi-band blending is to blend low frequencies over a large spatial range, and high frequencies over a short range.
- We initialize blending weights for each image by finding the set of points for which image i is most responsible.
- These max-weight maps are successively blurred to form the blending weights for each band, along with a similar blurring performed on the image itself to build each band

Multi-Band Blending

- For each band, overlapping images are linearly combined using the corresponding blend weights.
- Finally, we add the different bands together to obtain the final panorama.

$$I_{k\sigma}^{multi}(\theta,\phi) = \frac{\sum_{i=1}^{n} B_{k\sigma}^{i}(\theta,\phi) W_{k\sigma}^{i}(\theta,\phi)}{\sum_{i=1}^{n} W_{k\sigma}^{i}(\theta,\phi)}.$$

Results

Results obtained from different experiments

Note

- First, one of the difficulties of this problem is that there is no real way to evaluate an image stitching algorithm.
- Unlike a problem of classification or regression, there is no metric (or at least no metric that is being widely used and accepted), and methods are often compared on specific examples that showcase the differences between the panoramas (distortion of the shapes, misalignment, etc.).
- Therefore, we proceed in the same way, by showing the effect of the algorithm on a few examples rather than by providing numbers.

Gain Compensation

With Gain Compensation

Uniform intensity across the panorama

Increasing brigtness from left to right

Without Gain Compensation

Linear vs Multi-Band Blending

Linear Blending

Multi-Band Blending

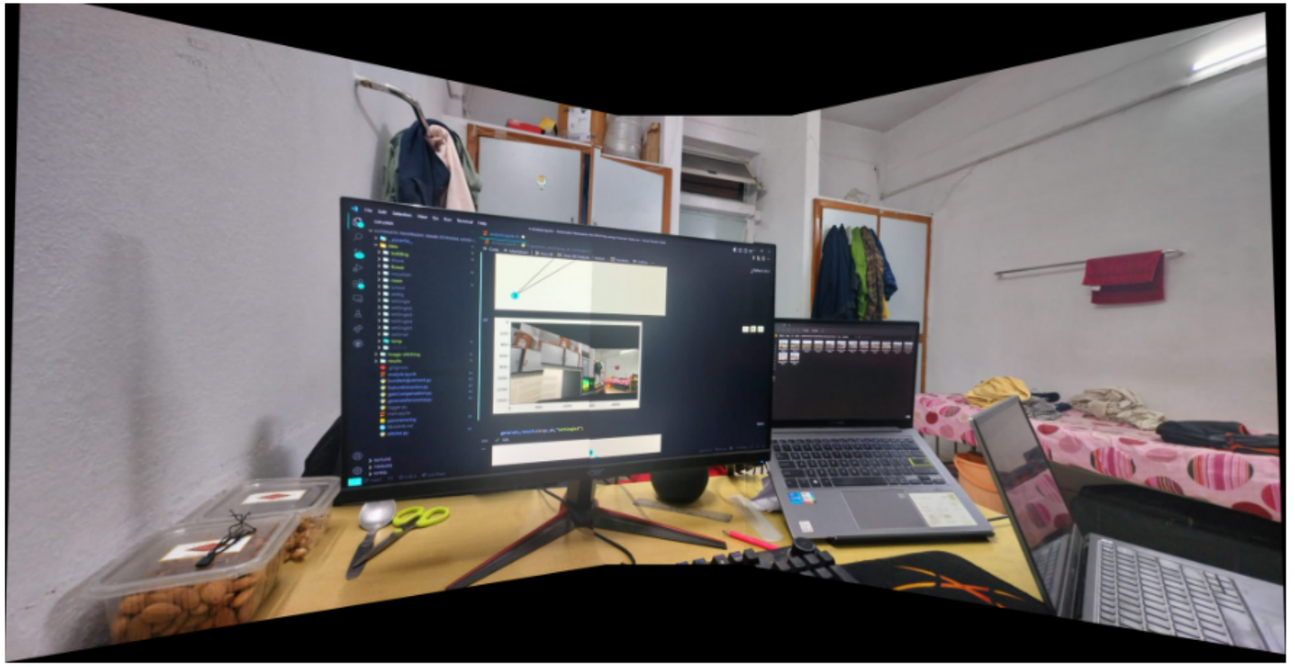
Linear vs Multi-Band Blending

Linear Blending

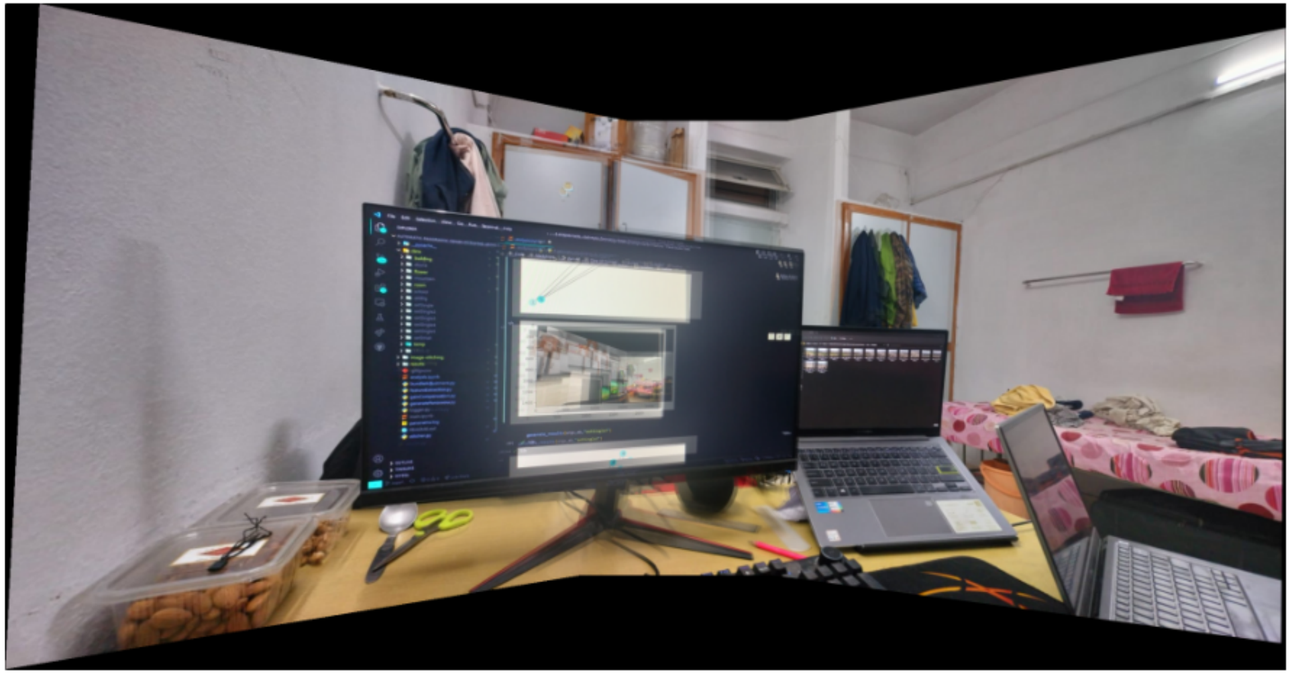
Multi-Band Blending

Multi Band Blending

Linear Blending



Multi Band Blending



Linear Blending

Varying σ

σ = 2

Varying σ

σ = 1

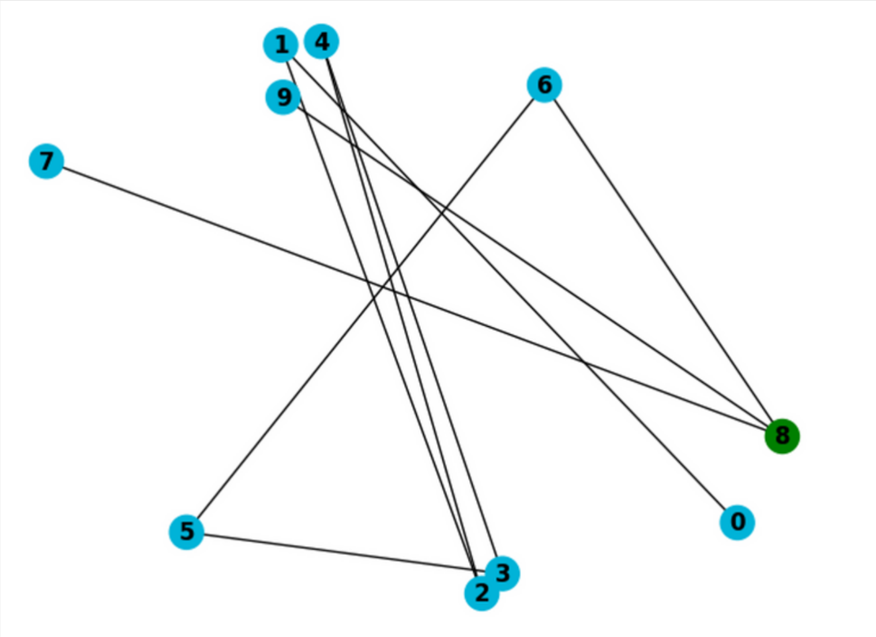
Varying σ

σ = 0.5

Varying σ

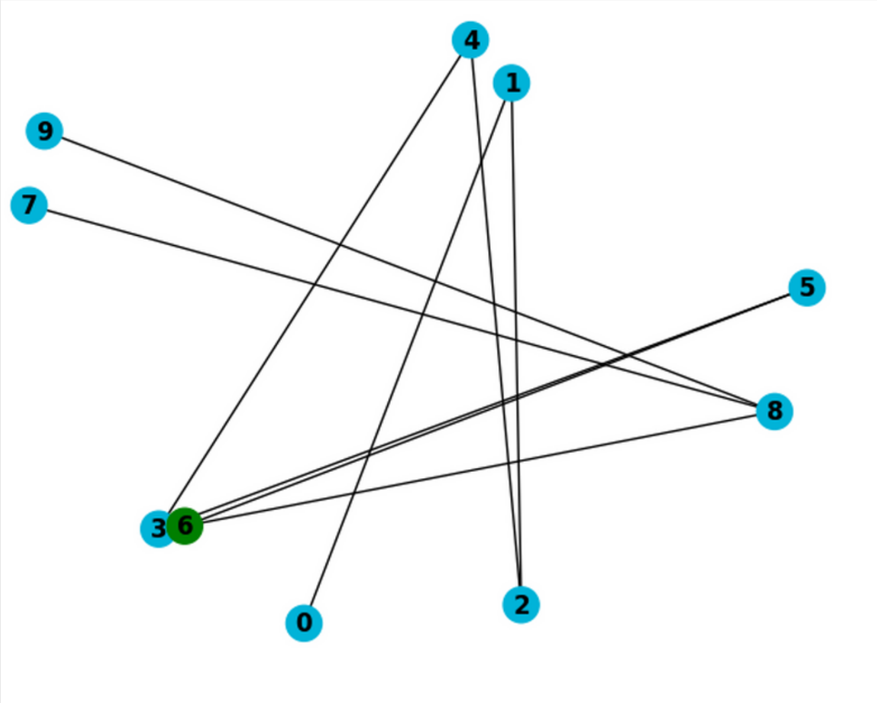
σ = 0.1

Varying Start Point in sequence



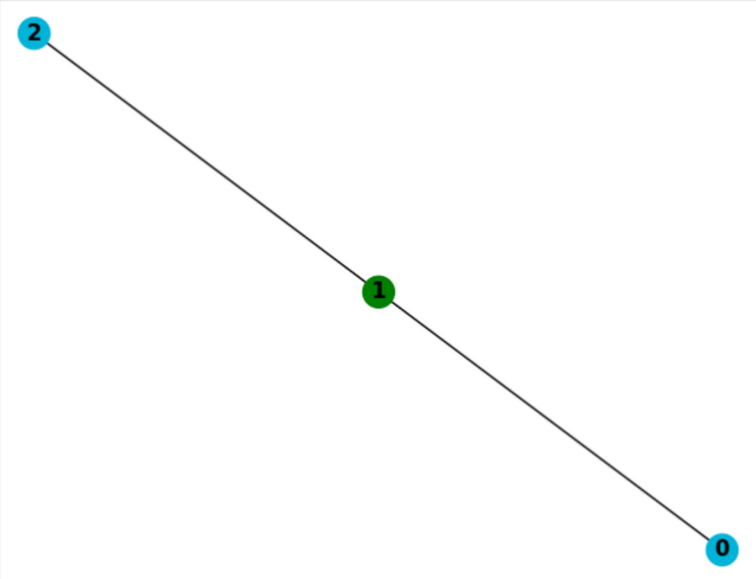
mode: Degree

Varying Start Point in sequence



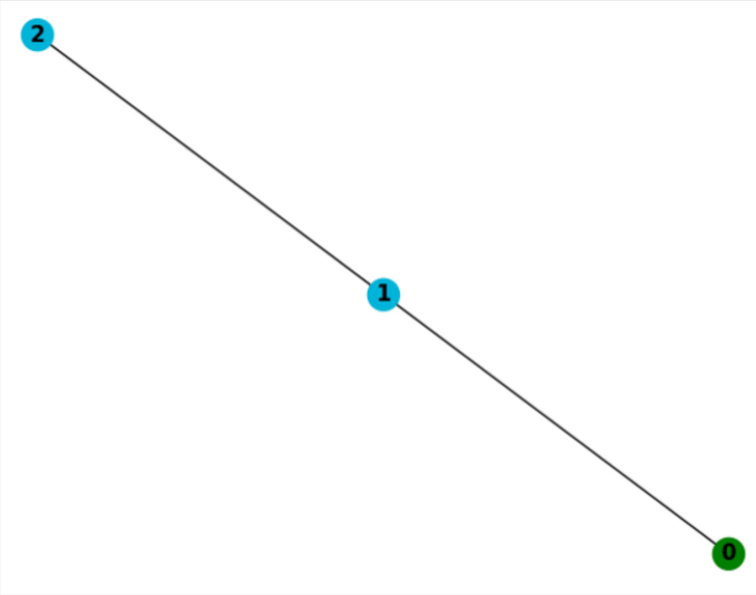
mode: Random

Feature Matching



FLANN matcher

Feature Matching

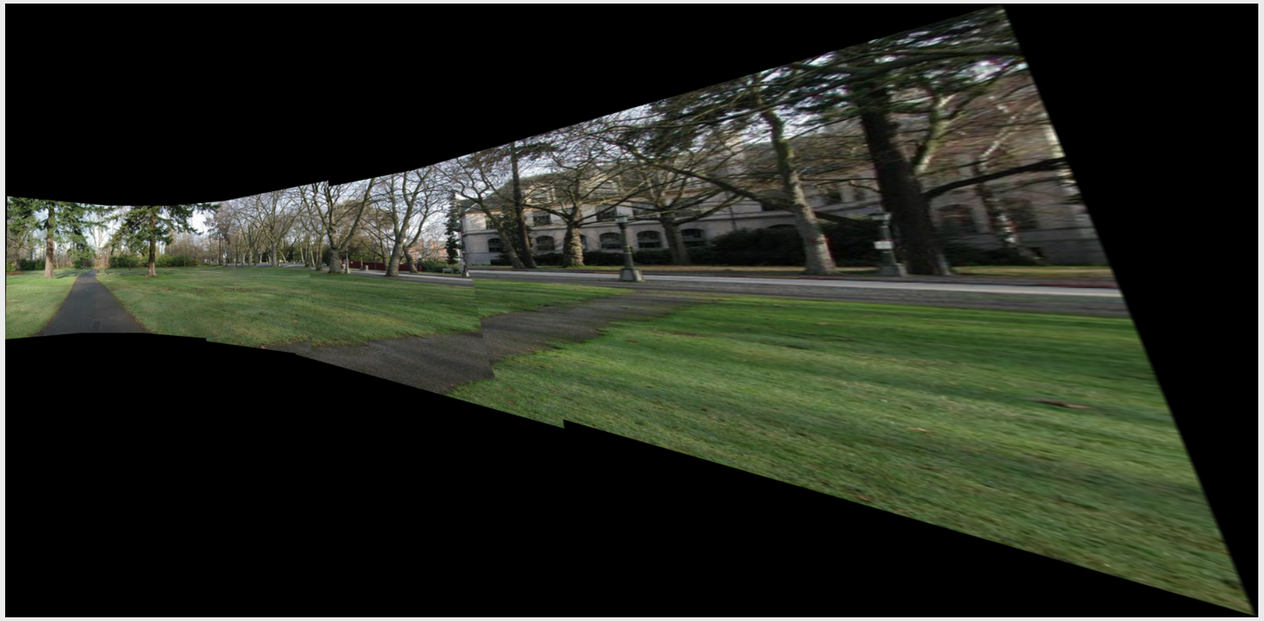


Brute Force
matcher

UDIS-D comparision

Future Work

- One improvement that we feel could be done in our project is the way we determine the source image in a panorama onto which all other images are transformed.
- Choosing the image based on the number of matches sometimes gave a little distorted and poor result as compared to some cases when the source was selected randomly. So having a better heuristic for this to determine a plane onto which all images are transformed (not necessarily always source) would be really great.



Work distribution

1. Feature Extraction and Probabilistic Model for Image Match Verification:
    - Shreyash and Gautam
2. Bundle Adjustment:
    - Shreyash and Gautam
3. Gain Compensation:
    - Mayank
4. Linear Blending:
    - Shreyash
5. Multi-Band Blending:
    - Gautam and Mayank

References

1. The paper we have implemented:
   http://matthewalunbrown.com/papers/ijcv2007.pdf
2. Homography Estimation:
   https://www.hindawi.com/journals/mpe/2014/897050/

Thank You