Comp Photography
Final Project

Ramesh Ramasubramanian
Aarsh Talati

Fall 2017

rameshmanian@gatech.edu,atalati6@gatech.edu

Search and Replace

The idea has been inspired from the “search and replace” functionality available in text editors and word processors. Just how we find for a word in the document and replace it in multiple pages, the idea is to make an edit to one image and have it propagated throughout multiple images captured by a photographer in a single session (i.e. same time, place, ambience etc.).
http://people.csail.mit.edu/billf/publications/Search-and-Replace.pdf

The Goal of Your Project

Original project scope:

The scope of our project basically would try to find edits made to a single person's face and would try to apply the same edit to other pictures in the same album. It is similar to search and replace functions normally found in word processing software.

We want to identify the edits made to a face, like adding/removing a mole, add or remove a tattoo and replicate the changes to the images of the person's face in a folder with different angles.

The Goal of Your Project

What motivated you to do this project?

Adobe Lightroom *does* have an option to synchronise adjustments like exposure, tone curve, white balance etc.(page 210) but unfortunately, Lightroom does not seem to have an option for syncing edits across multiple images. Being a hobbyist photographer myself, I've personally spent a lot of time making same fixes over and over again.

This is a compelling and visually effective feature, which can remove blemishes both natural and ones created by devices.

Scope Changes

- Did you run into issues that required you to change project scope from your proposal? Give a detailed explanation of what changed.

Yes. In order to save time and have a proof of concept, we limited our tests to making a single edit to a human. We used OpenCV’s Haar cascades for face detection and ORB for feature detection.

Showcase

Input

Output

Terminology

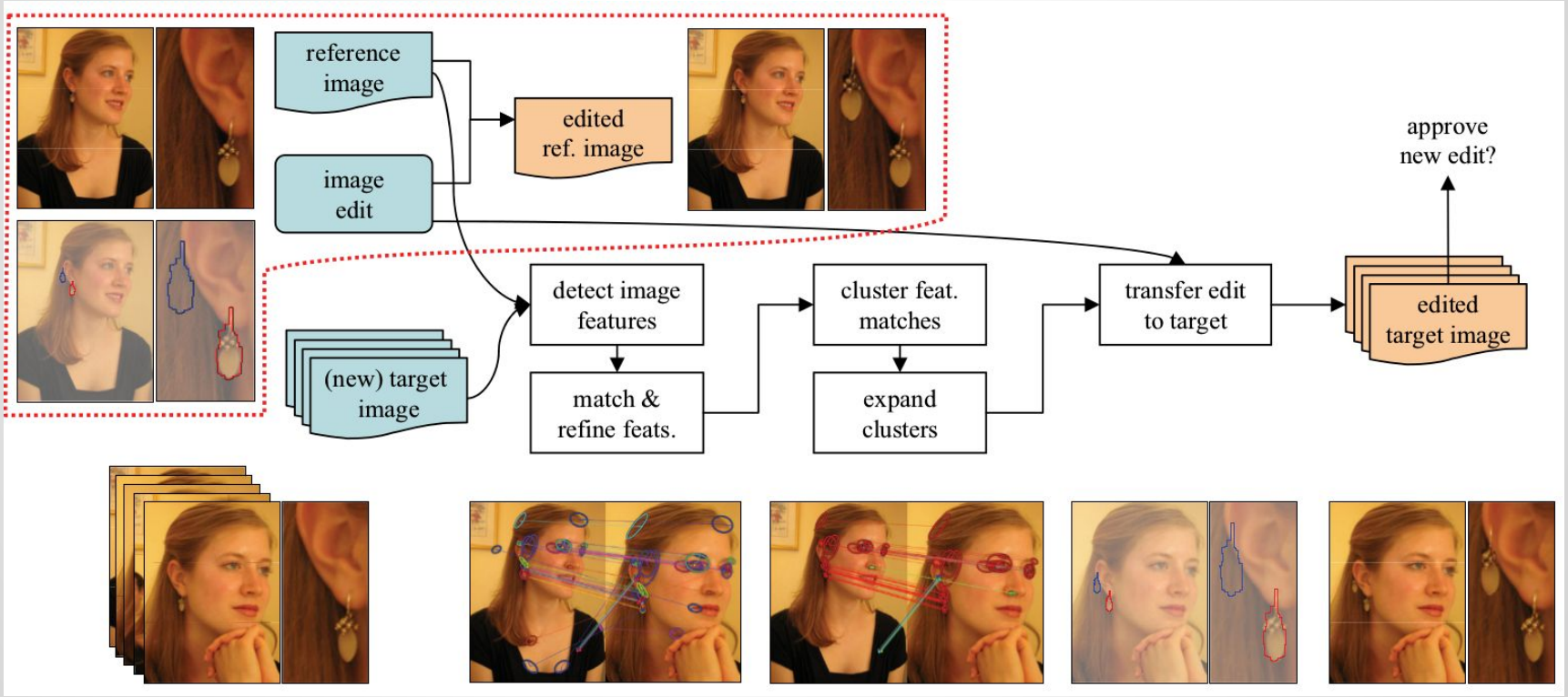
Album images: target images where the edit needs to be transferred or applied

Source reference image: Original unedited image to which the user applied edits
(a.k.a. **src ref img**)

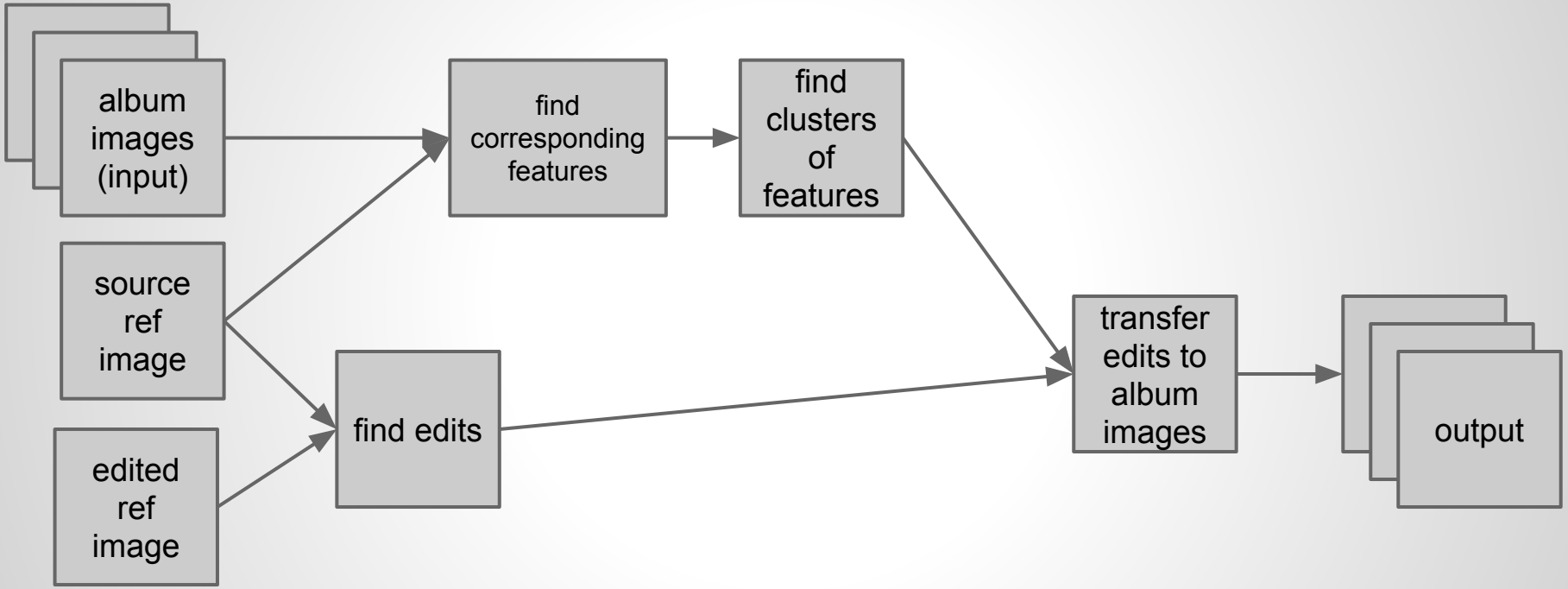
Edit reference image: The resultant image after user applied the edit.
(a.k.a. **edit ref img**)

Edit region: The region of difference between source ref img and edit ref img.

Project Pipeline

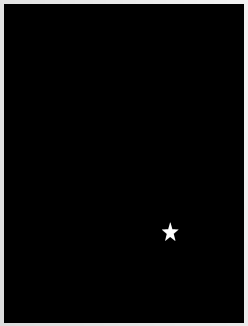


Project Pipeline



Project Pipeline

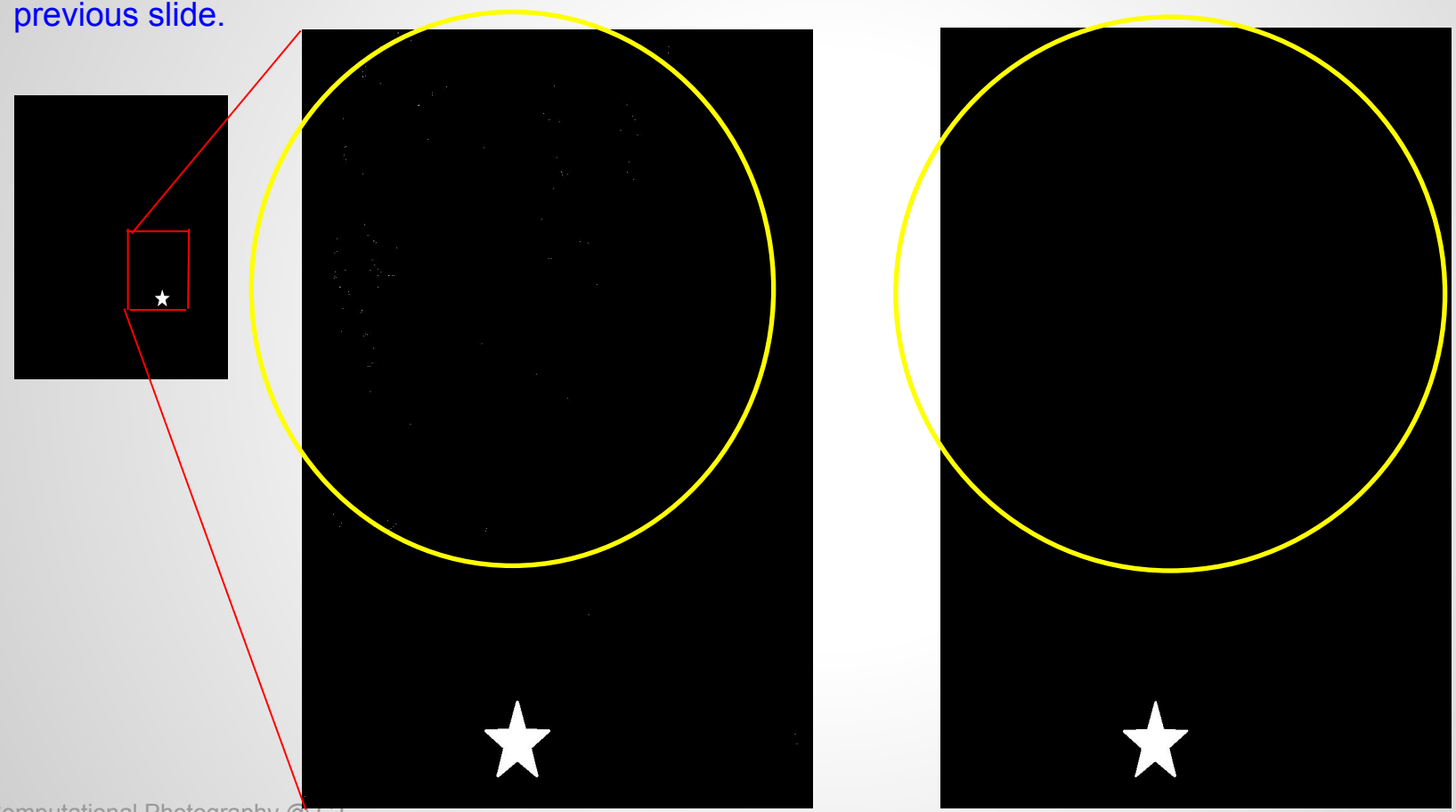
(Step 1: Finding Edits)



The edits could be found simply by converting images to greyscale and comparing corresponding intensity values. But quite often depending on image editing program used to make an edit, some noise might get introduced. To mitigate that, I decided to perform a closing operation which is a dilation followed by an erosion. Closing helped group areas which are close together, into a single object. Performing an opening operation afterwards, can help remove isolated noisy areas, using a structuring element of a smaller size (2x2 vs 5x5), in order to avoid removal of larger areas by mistake. Finally, masking out any extra information with the original image would leave the larger areas intact while removing the small islands.

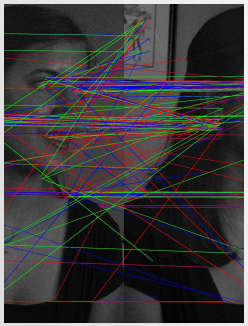
(Step 1: Finding Edits)

This slide shows the difference of before (left) and after(right) applying morphology as discussed in the previous slide.



Project Pipeline

(Step 2: Feature Matching)



Using ORB feature detector, we find corresponding features between two images and find matches. Matches were found using brute force matching cv2.BFMatcher() with Hamming distance. It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned.

After having spent significant time, we did not get the desired results (as described in the research paper). The authors of the paper used an external program to do feature detection (Oxford paper) and matching. They used Harris, Hess affine and MSER for detecting features. We restricted ourselves to just ORB. The authors were also able to get the affine frame in which the features were detected and our attempts to extract the affine frame failed. With more time, we have no doubt that we will be able to extract the frames and use them to compare and eliminate matches that don't have corresponding geometry.

Step 2: Feature Matching

Using the technique described in the previous slide, this image demonstrates the feature matching between src ref img and edit ref img.

The next slide demonstrates the feature matching output between src ref img and each album image.

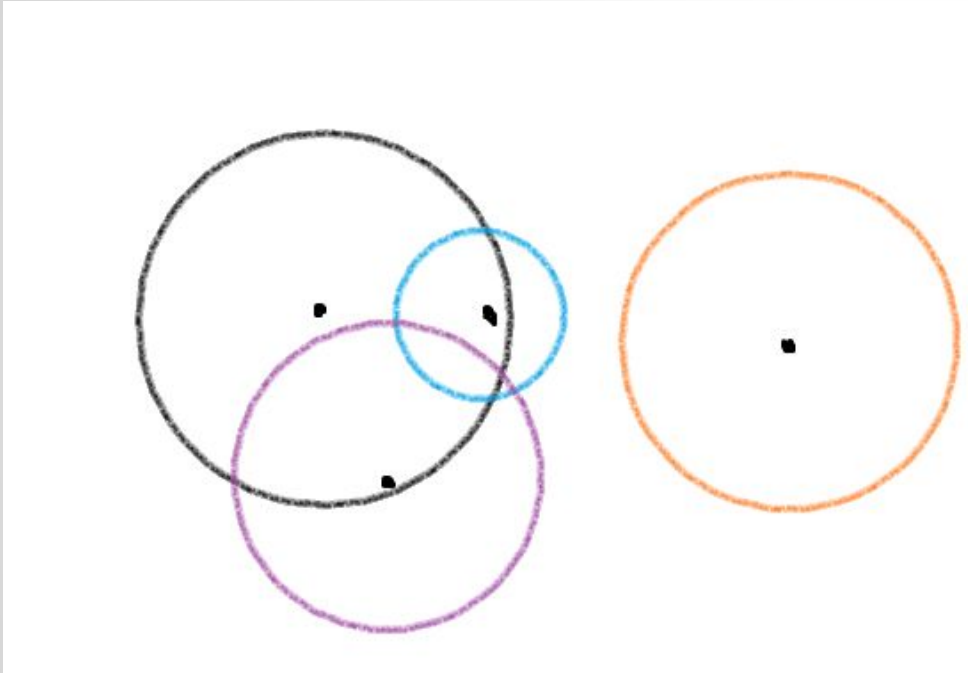
We chose to compare with unedited image so that we can minimize the change between src ref img and album image in order to improvise feature matching.

Step 2: Feature Matching

Step 2: Feature Matching

- After feature matching, we now have addressed correspondence problem (depending on how well does feature detection algorithm work for a given set of images). And we already know the edition region.
- Hence the next step is to improvise detected features using clustering.
- After that, we need to find cluster centroids near the edit region in the src ref img and then try to locate the target edit region for a given album image.
- Our initial strategy was to find N number of cluster centers inside the edit region or those nearest from the edit region.
- Then draw the arc and find the intersection point to find the target edit region center.
- The same has illustrated in the next slide.

Step 2: Feature Matching



- Since we know that how exactly far a center of a cluster is from the center of the edit region in the src ref img, the idea was to use approximation and use the same distance for the corresponding cluster center points in the album image and draw a circle with the radius being the approximated distance from src ref img. This might yield to an approximate location of the edit region.

Step 2: Feature Matching

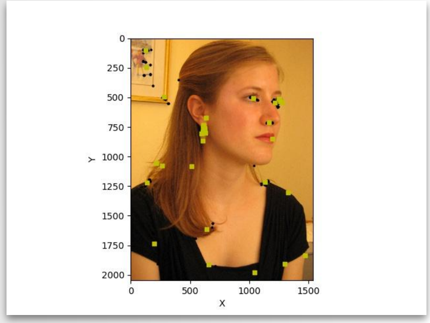
- Since we could not get that to work, we ended up using triangulation. This requires us to pass 3 points and distances in order to find target edit region. The algorithm is below which returns the coordinates of the edit region.

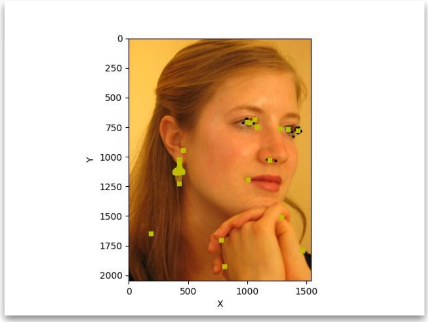
```
def get_location(P1,P2,P3,DistA,DistB,DistC):
    ex = (P2 - P1)/(numpy.linalg.norm(P2 - P1))
    i = numpy.dot(ex, P3 - P1)
    ey = (P3 - P1 - i*ex)/(numpy.linalg.norm(P3 - P1 - i*ex))
    ez = numpy.cross(ex,ey)
    d = numpy.linalg.norm(P2 - P1)
    j = numpy.dot(ey, P3 - P1)
    x = (pow(DistA,2) - pow(DistB,2) + pow(d,2))/(2*d)
    y = ((pow(DistA,2) - pow(DistC,2) + pow(i,2) + pow(j,2))/(2*j)) - ((i/j)*x)
    triPt = P1 + x*ex + y*ey
    return triPt
```

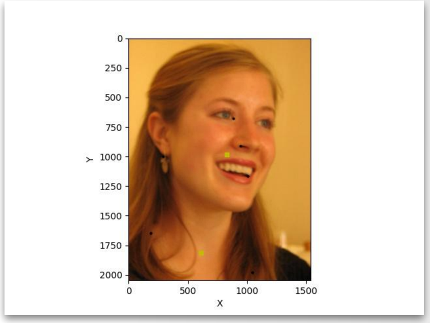
Step 3 - Clustering

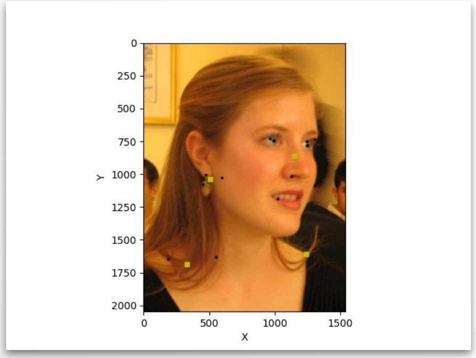
- One of the key goals in this project is to identify feature matches between the source reference image and the target image, so that when an edit is transferred, the geography of the target image where the edit would reside will resonate with the source reference image edit area.
- Clustering of features assists in the quality of the edit transfer region. In this step for clustering, we have taken the feature matches from the album (target) images and cluster them into groups. Number of clusters is determined by dividing the number of matches by 5. We use cv2.kmeans for identifying the clusters.
- Once the clusters and its centroids are ascertained, the matching features can further be refined by identifying which features may share the vicinity of the edited region, by identifying the centroids of the clusters to which the features belong.

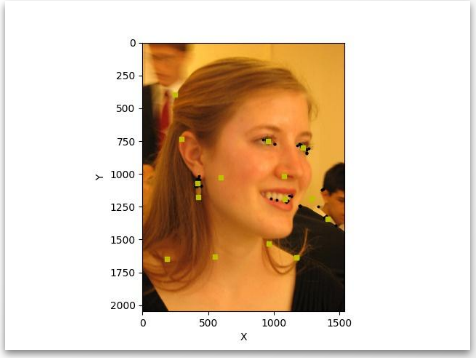
Cluster images











- The images in the next slide show the feature points, black in color, and the centroids of the clusters, in yellow.

Step 3 - Clustering

When the features close to the edit region are identified and the matching features in the target images are corresponded, the approximate edit transfer region in the target image can be identified. This is done by getting the nearest (Euclidean distance) features around the edit region and approximating the destination edit region based on that.

It is disappointing that we did not get enough time to get the edit region more accurate by some of the techniques discussed in the paper, including finding clustering by homographies. Identifying homographies for matches below 4 for a cluster was not possible. However even clusters with a dozen or so matches, we encountered error in getting back the homography using cv2.findHomography().

We chose not to look at cluster expansion as is done by the researchers due to increase of scope.

Project Pipeline

(Step 4: Transferring Edits)

After identifying target location for the edit transfer, and the edit region, this step became a little trivial. All that was left to do was to copy the edit region to target region.

Demonstration: Result Sets

- Demonstrate **at least three complete result sets**. If your results can easily be shown here in your slides, please do. Explain **in detail** what you are showing.
  - These may be incorporated into the Project Development section, but make sure they are complete.
- Ensure that you have at least three result sets unless you have already notified us that it is impractical for technical reasons.
  - Use private post to Instructors on Piazza. Not having time to take pictures is not sufficient.
- Provide links to videos or gifs, and large or numerous image sets.
  - Make sure they work, and will remain available until the end of the term.
  - Google drive and Dropbox are good stable sharing locations, there are others.
  - Must be viewable to everyone with the link.

Use several pages, as necessary

Demonstration: Result Sets (cont'd)

Input

Demonstration: Result Sets (cont'd)

output

Project Development

- We started our project with careful reading of the paper. We concluded that the paper was quite technical and challenging and in order to even get close to successfully, we will need two people and all our wits. We started with collaborative discussions on the pipeline and discussion to understand the paper itself. It was a non-trivial task. To give an example, we spent a fair bit of time reading and researching section 2.1 and trying to understand the technical details.
- We got through to identifying the pipeline needed to get our project done. We eliminated the use of face detection that was used by the authors. If we decided to include face detection, we already had some reliable results using haar cascades available in OpenCV. It took us the first week of our two week project to get this far, primarily because we needed to understand the paper in detail.
- We then concluded that we should use the images used by the authors. We contacted MIT and got the permission needed to use the images for this project (from Fredo Durand at MIT)
- We started work on multiple areas (pipeline tasks)
- We had less than satisfactory results from our feature detection and matching (explanations in our slides earlier). The matches were not consistently accurate and we spent a lot of time trying to understand why and how to solve them. We tried reading stackoverflow and other research papers including (http://lear.inrialpes.fr/affine/index.html#publications) , from the authors who wrote the feature detection and matching software used by MIT researchers.

Project Development (cont’d)

- We finished writing the feature detection using ORB, unsuccessful as it may be, and we wrote the basic clustering routine to group the features into cluster. This was also done partially. These are the two key elements and neither was finished fully to our satisfaction as the results were not good.
- We think we should have just taken only one aspect of it and did it thoroughly and spent the time. We did not anticipate the level of difficulty we encountered in either understanding the subject matter to the level required to achieve this nor did we have sufficient time to execute this.

Computation: Code Functional Description

Pipeline for albums to be read

- We identify the number of features to detect and matches to make. We also established the input and output directories of images. We created a directory for source reference image and edit reference image image under the albums/ref folder.
- We execute the pipeline for each and every image

Feature detection and matching

- We used ORB feature detector and BFMatcher, both from OpenCV for

Clustering

- Originally, we had spent a fair bit of time writing the clustering code from scratch, only to discover that cv2 already has a K-Means cluster method available. We used that method in the end. We chose to do random initial means with an iteration max of 20.

Edit transfer

- Edit transfer ended up being a trivial task of overwriting an array to the album image. But that was not my original plan. The plan was to use pyramid blending instead trivial approach of insertion blend, but due to time constraints we did not have the opportunity to get there.

Computation: Code Functional Description

Libraries/packages used

- We ended up using standard libraries like: OpenCV, numpy, scipy

Resources

- Hasinoff, S. W., Jóźwiak, M., Durand, F., & Freeman, W. T. (2010, March). Search-and-replace editing for personal photo collections. In Computational Photography (ICCP), 2010 IEEE International Conference on (pp. 1-8). IEEE
- http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html
- http://lear.inrialpes.fr/affine/index.html#publications
- https://github.com/opencv/opencv/tree/master/data/haarcascades
- https://gis.stackexchange.com/questions/66/trilateration-using-3-latitude-and-longitude-points-and-3-distances

Teamwork

- If you worked in a Team, describe your original division of labor, and how that worked out.
  - Ramesh worked on clustering, triangulation and I worked on feature detection, edit transfer, feature matching, haar cascades etc. But we actually worked very closely with each other and ended up staying in the tight loop of communication and collaboration. In the end we were really happy to get an opportunity to work together.
- Do you feel that the actual division of labor to develop your project, code, and report was equitable?
  - I feel that I had a pleasure working with Ramesh and I would have no hesitation working with him again in the future.

Appendix: Your Code

Code Language: Python

List of code files:

- *Main.py*
- *Cluster.py*
- *edit_detect.py*
- *Feature_detect.py*
- *Triangulation.py*
- *Utils.py*
- *hcascade.py*
- *haarcascade_frontalface_default.xml*

Credits or Thanks

- We would like to thank the authors of this paper for their help to get us the test images.

Credits or Thanks

Subject: Re: [Time Sensitive] Permission Request: Search-and-replace
Date: Thu, 16 Nov 2017 09:17:48 -0500
From: Fredo Durand <fredopdurand@gmail.com>
To: Aarsh Talati <atalati6@gatech.edu>
CC: Ellie Zucker <eliaranov@csail.mit.edu>, Sam Hasinoff <hasinoff@google.com>

Sure, no problem.

On Thu, Nov 16, 2017 at 9:07 AM, Aarsh Talati <atalati6@gatech.edu> wrote:

> Aarsh Talati
>
> atalati6@gatech.edu
>
> Georgia Institute of Technology
>
> Atalanta, GA
>
> **Subject:** Requesting permission and test data used for your paper, for educational use
>
> Greetings,
>
> I am a graduate student in the class of "Computational Photography". In this current Fall 2017 semester, I am working on a final project based on your paper "Search-and-replace editing for personal photo collections". Part of my work is to reproduce the results of the paper. It would help evaluate my work if I could compare my results with yours for the same albums.
>
> Sir, I would like to request your permissions and blessings to be able to use the same images as yours. I would also like to know from where can I access or download these pictures.
>
> Sincerely,
>
> -Aarsh Talati,
>
> Graduate Student, Georgia Tech