Stanford Navigation

Android Phone Navigation System based on the SIFT image recognition algorithm

Matt Yu
Department of Electrical Engineering
Stanford University
Palo Alto, CA
mattcyu@stanford.com

Uchechukwuka D. Monu
Department of Electrical Engineering
Stanford University
Palo Alto, CA
udmonu@stanford.edu

***Abstract* — A mobile navigation system is developed based on landmark detection. SIFT and a Hierarchical K-means tree are used to create a database of images of landmarks around Stanford University. A user takes a picture from their Android device and sends the query to a server running MATLAB. Correspondences are found and RANSAC is used to determine the best match. After receiving their location, the user is prompted to choose their destination and directions are given.**

Keywords: SIFT, feature detection, RANSAC, Hierarchical K-means tree, Android, image recognition, image segmentation

I. INTRODUCTION

Image recognition and matching has proven to be an extremely applicable algorithm across various disciplines, and in creating navigation systems, this is no different. Figuring out how to get from where you are to where you want to go has been an enduring problem for humankind. It gave rise to the compass, maps, GPS, and Google Maps. Paper maps in particular have been key tools used by people to help familiarize, plan and navigate through an unknown terrain. Several countries, facilities, conferences and a lot of venues provide maps to visitors of the destination. Despite the availability of this helpful tool, there is a lot that can be done to simplify the ambiguity of this 2D diagrammatic representation of an area while enhancing the navigation experience of the user. Our proposed solution is the Stanford Navigator System. This android application enhances the experience of a visiting tourist to the Stanford campus by identifying locations around campus and providing compass directions to selected destinations. Stanford visitors looking to identify their current location can take a picture with their phone and get back the name of the building or landmark in question. They are then prompted to select a destination through a lookup table and subsequently provided with compass directions from where they are to where they would like to go. The two major sections of this project are image matching and image segmentation. For the image matching section, rapid identification of the landmark or building, invariant to the orientation, scale or lightening is extremely important. There are many techniques available in the field of digital image processing that could help with automating this section of the project. Two extremely popular algorithms: Scale-Invariant Feature Transform (SIFT)[3,4] and more recently Speeded Up Robust Feature (SURF)[11] were the techniques considered when drawing up the proposal for this project. Although SURF has been documented to outperform SIFT in the areas of repeatability, distinctiveness, robustness and speed, SIFT is considered the golden standard and proved to be the preferred choice for the depth and functionality of our project. This paper presents the different components of the proposed EE368 Stanford Navigator Project. In addition to the overall introduction of the system covered in this section, part II looks into similar work that has been presented out there in the literature. We go into the detail of our methodology in part III. Experimental results, predominantly from the image matching segment of this project are displayed in part IV and further results from image segmentation are shown in part V. The results from the image segmentation portion of this project are preliminary and serve as a possible future direction of this project.

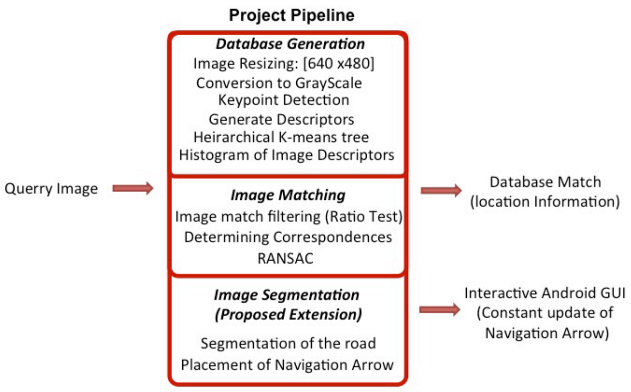


Figure 1: Project Outline

II. RELATED WORK

In this section, we shall briefly look at other applications of feature matching algorithms as well as current research alternatives to the different portions of this project. Image matching is widely applicable in areas of machine vision, environmental analysis, social experience enhancements and much more. The SIFT algorithm has been implemented in projects involving CD Cover recognition [5], Identification of Gallery Images [6], Restaurant Rating Provider [7], Road

Block Determination [8] to name a few. Within the field of mobile phone navigation systems, there has been research done in landmark-based pedestrian navigation systems [9]. This project focused on leveraging an online collection of geo-tagged photographs. Work has also been done in augmented videos and panoramas for pedestrian navigation [10]. This research focused on combining realistic views and location specific information on PDA's and next generation cellular phones. These research topics mentioned are a few of mant currently conducted research in this area of digital image processing.

III. DESCRIPTION OF METHODS/ ALGORITHM SELECTION & OVERVIEW

A. Scale-Invariant Feature Transform (SIFT)

When performing object recognition of images in the real world, it is extremely important that local image features are unaffected by nearby clutter or partial occlusion. Unfortunately, finding extremely unique features that will enhance image matching is a sufficiently difficult problem and the SIFT algorithm [3,4] was developed and shown to perform efficient recognition using local image descriptors sampled at a large number of repeatable locations. The SIFT algorithm was the image matching algorithm of choice for our project. It is considered the gold standard of image feature extraction and helps to detect robust keypoints, generate feature descriptors for these keypoints and eventually compare features and descriptors from the query image to that of the generated database to determine the closest match. A key functionality of the SIFT algorithm is its invariance to image translation, scaling, rotation and its partially invariance to illumination changes and affine or 3D projection. This made it an ideal method of choice for the development of our image database. The SIFT algorithm can be split up into multiple parts:

Scale Space Construction: This section of SIFT serves to create internal scale invariant representations of the original image. This is done by progressively blurring and shrinking the original image with a Gaussian function. This blur helps to get rid of details and produces different levels (octaves) of the image. The amount of blurring introduced by the Gaussian function is important and is dependent on the sigma of the Gaussian function. The number of octaves and scale is dependent on the size of the original image. SIFT creates a default of 4 octaves and 5 blur levels. Figure (2) demonstrates the formulation of the octaves and scales.

$$L(x,y,\sigma) = G(x,y,\sigma) * I(xy.y)$$
$$L(x,y,\sigma) = Blurred\ Image$$
$$G(x,y,\sigma) = Gaussian\ Blur\ Operator$$
$$= \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$
$$I(x,y) = Original\ Image$$

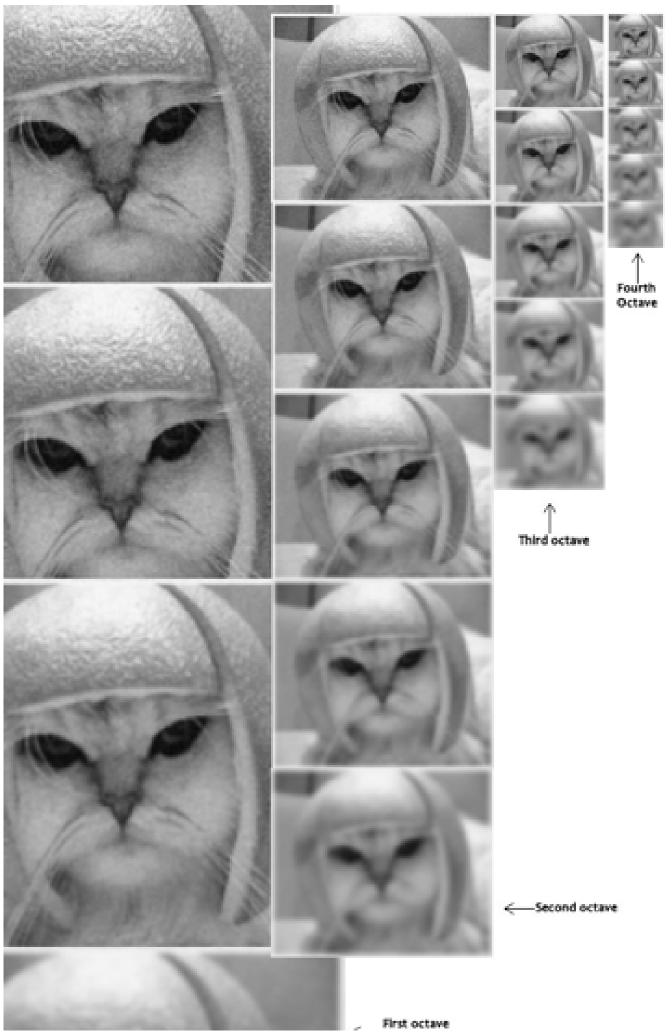


Figure 2: SIFT Octaves and Scales
(Courtesy of Reference 12)

Laplacian of Gaussian Approximation: The blurred images obtained from the scale space construction section of the SIFT algorithm are then used for the generation of another set of images known as the Difference of Gaussians (DoG).

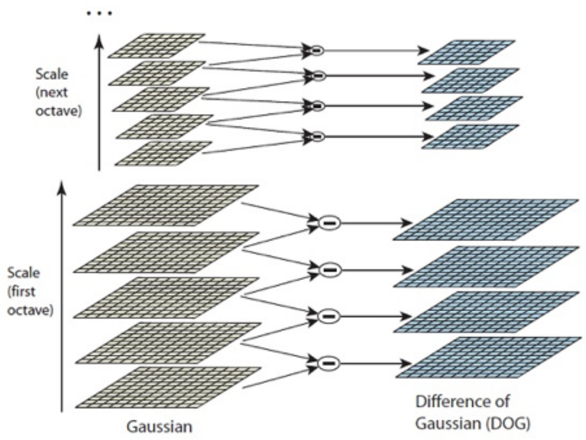


Figure 3: Difference of Gaussians Calculations [12]

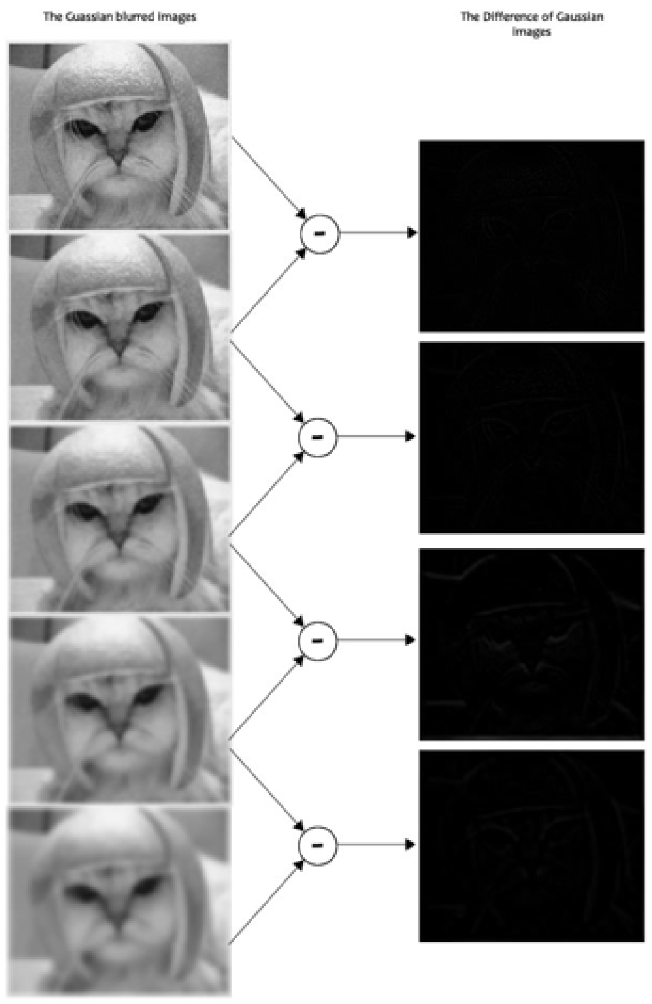

The Guassian blurred images
The Difference of Gaussian images

Figure 4: Difference of Gaussians Images [12]

The Laplacian of Gaussian operator is essential for locating edges and corners of an image. These edges and corners are good structures for finding keypoints. However, the problem with calculating a LoG is its sensitivity to noise and its computational intensity. This is where the DoG comes in. The DoG's serves as a good approximation of the LoG's. In scale space, simply subtracting the images obtained provides a fast and efficient alternative to calculating the LoG while adding the extra benefit of being scale invariant.

Detection of Keypoints: To detect the key points from the calculate DoG images, one has to locate the maxima/minima in these images. This is done my comparing neighboring pixels in the current scale, the scale above and the scale below. Local maxima are determined by comparing the DoG stack with its dilation by a three pixel square structuring element. The local minima is found by comparing the DoG stack with its erosion by the same structuring element.

Further processing is done to filter through the keypoints generated in the step above. This is necessary to increase the efficiency and robustness of the algorithm. Two major reasons for the filtering off of a keypoint are edge location and low contrast. Thresholding helps to accomplish this filtering.

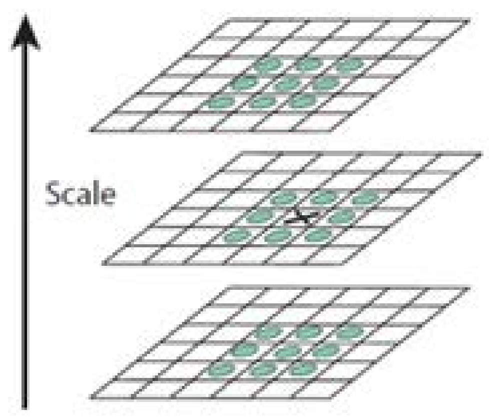


Figure 5: Detection of Keypoints [4]

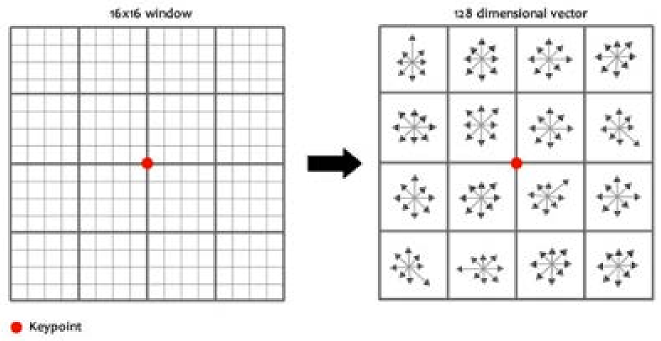


Figure 6: Determination of the Feature Descriptors

Feature Descriptor: The three steps described above were used in determining the features of each image. We will now talk about the feature descriptors. These are known as the "fingerprints" of each keypoint. This fingerprint is created by generated a 16x16 window around the keypoint, which is further split up into 4x4 windows. For each 4x4 window generated, we need to calculate the gradient magnitudes and orientation. These values are then binned. In total, there are 128 values calculated. These values are unique to the feature and help in distinguishing them from other features.

B. Creation of the Database

A database containing 114 training images from 35 locations on the Stanford campus was created using the SIFT extracted features and descriptors. 2-5 images were captured for each location and each image was resized to a resolution of [640x480]. This resolution generated ~1000 descriptors per image feature. A k-means tree was then generated from the descriptors of each image using the vl-hikmean function of the vl_feat library. Each path of the image was placed into bins and the final structure was stored for later comparison with the query image. The code written for this section can be used to create a database for any image set. The description of the k-means tree and binning are further described below.

C. *Hierarchical K-means Tree*

Each image in the database has about 1000 features (obtained from SIFT), each of which can be described by a 128-dimensional vector. All of these features (~100000 in total) are then used to create a tree with K branches, N leaves, and $\log_K N$ levels. At the first level of the tree, the space spanned by the feature vectors is split up into K regions (done via a clustering algorithm). At the second level, each region in the first level is again split up into K regions. This process happens until we reach the leaf nodes. Now, a feature is pushed through the tree to determine which leaf node it is most similar to. Since we choose K, N such that we have many (10 times, generally) more features than we have leaf nodes, many features will fall into the same bin. In this way, we are categorizing all possible feature vectors in this 128-dimensional space as one of N possible bins. As well, each of the N bins were weighted based on the number of features which well into the bin. Let N_i be the number of features which fall into bin i, then the weight associated with bin i, w_i is calculated as:

$$w_i = \begin{cases} \log\left(\frac{N}{N_i}\right), & N_i \neq 0 \\ 0, & N_i = 0 \end{cases}$$

Furthermore, we can now describe each image by the number of its features which fall into each bin. Creating this histogram is equivalent to describing the image by a N-dimensional vector. Thus, when we want to compare any two images, we can take the l_2 norm between the two vectors as a measure of their similarity. In this project, we use this comparison as the first filter. When the user sends a query image, a histogram of the images are created and then compared with the histograms of all the database images. The 10 closest matches are then further filtered to determine the best image.

Notably, we first attempted to compare images via the following algorithm. For each feature in the query image, push it through the k-means tree to find its corresponding bin. Then, every database image which also has a feature in the same bin will be incremented by one to generate a score. After running through all the features in the query image, the database images with the highest scores are the ones most like the query image. The biggest problem with this algorithm is that query images with many features in the same bin will naturally create very high scores for database images who have any number of features in that bin even if this number is very different than the number in the query image. Testing revealed that this algorithm did not perform well.

D. *Finding Correspondences*

Each of the 10 closest database images are then compared with the query image in order to determine correspondences. This consists of going through each feature in the database image and finding the closest two features (again l_2 norm) in the query image. If the closest feature is much smaller than the next closest feature, ie:

$$\frac{d_1}{d_2} < threshold$$

then we say that those are useful features and create a correspondence between them.

E. *Random Sample Consensus (RANSAC)*

RANSAC takes in the set of correspondences as an input. Then, it chooses three of these correspondences at random in order to generate an affine model which describes the transformation between the first and second image. Next, the algorithm applies this transformation to the first image. The transformed locations of the features are then compared to the actual location of the features in the second image. Pairs of features which are within a certain distance of each other are considered inliers whereas other features are discarded. The image with the greatest number of inliers is returned as the matching image. Note that if the number of inliers is too low, we indicate that none of the images matched.

F. *Image Segmentation*

One extension to this project was to determine where walkways were located via image segmentation and paint a yellow line in the direction a user would travel to get to their destination. Individual images were segmented as follows. A patch of pixels in the bottom center of the image was assumed to be a good representation of a walkway. The logic behind this assumption is that users who want to get directions to a location will normally already be standing on some kind of walkway. Therefore, when they point their phone in front of them, the bottom center of the picture will be a good representation of the walkway. Then, a texture filter was applied to the entire image, creating a binary mask where features with similar texture to the initial sample were kept. Erosion and an area filter were further used in order to remove noise. Finally, each remaining region was labeled. The results are shown in figure (7).

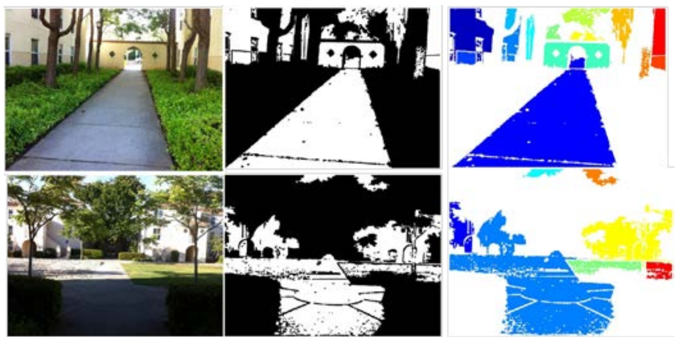


Figure 7: Original images (left), after texture/area filter (center), region labeling (right).

In the top set of images, the algorithm seems to perform well. There is a clear walkway which dominates the image. However, in the bottom set of images, the walkway is much more difficult to determine. The combination of similar color, texture, and varying lighting conditions all contribute to uncertainty on how to further filter the image. Something that could be attempted in the future is to filter the image in different domains separately (HSV, texture, etc) and intelligently combine the outputs of the filters.

IV. Experimental Results

The database was build from 114 pictures of 35 different locations on Stanford's campus. 3-5 images per location were captured at different orientations and during different lightning conditions. Figure [8] shows a sample of these database images for the Economics Building, Bechtel International Center and the Hewlett Building. After the database was setup according to the algorithm breakdown shown in Figure [1], testing against these same database images were carried out to determine the validity and robustness of this algorithm. There was a 100% matching result for these images. Random query images of different locations around the Stanford campus were then taken and tested against the database. If the location was present in the database in a similar position and of similar lightning conditions, a positive match was made. The initial setup of the algorithm used a weighting system described in the methods section for the database image descriptors instead of the binning implemented in our final system. This weighting technique proved not to provide a precise match compared to the binning system. Table[1] below shows the matching statistics for the query images against the database. Troubleshooting involved playing around with the image resolution, and different hierarchical k-node and NLeaves values. Optimum parameters for this initial system were an image resolution of [640x480], 10 k-nodes and 10,000 NLeaves.

Figure [8]: Sample database

Query Image Resolution	**[640x480] Database Images**	**[480x420] Database Images**	**[320x240] Database Images**	**[640x480] Random Images**
Database Matching Accuracy [640x480]	71% (114 Images)	42%	42%	50% (59 Images)

Table [1]: Database Image Matching Statistics for different images

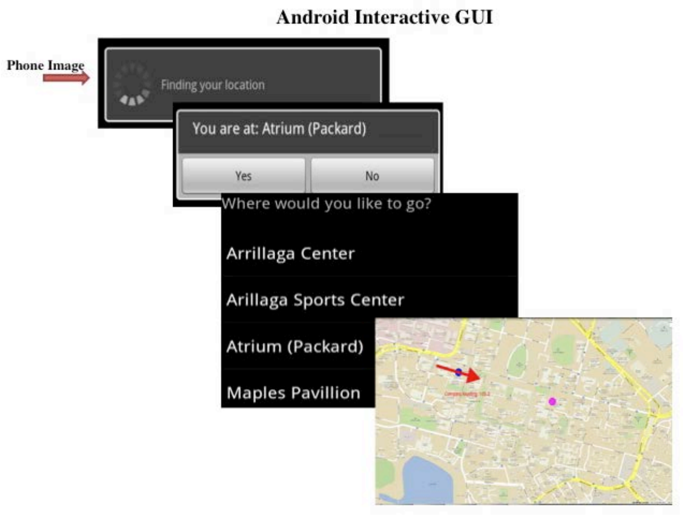
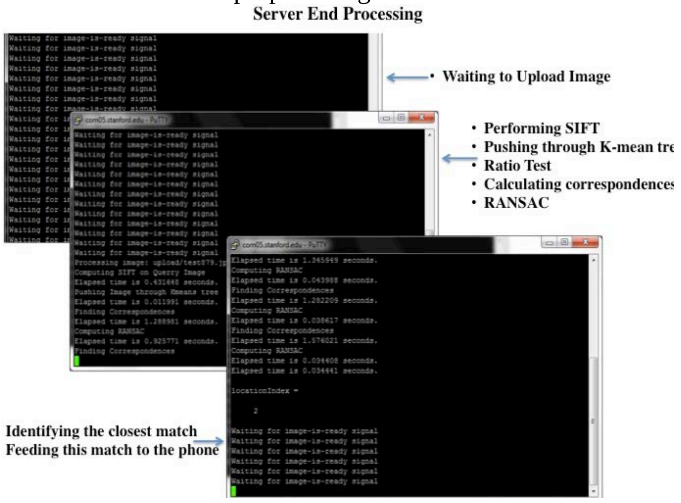


Figure [9]: Progression of the android feedback for our proposed algorithm


Server End Processing
Waiting to Upload Image
Performing SIFT
Pushing through K-mean tre
Ratio Test
Calculating correspondences
RANSAC
Identifying the closest match
Feeding this match to the phone
com05.stanford.edu - PuTTY
com05.stanford.edu - PuTTY

```
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
Waiting for i
```

```
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Processing image: upload/test879.jp
Computing SIFT on Querry Image
Elapsed time is 0.431648 seconds.
Pushing Image through Kmeans tree
Elapsed time is 0.011991 seconds.
Finding Correspondences
Elapsed time is 1.288981 seconds.
Computing RANSAC
Elapsed time is 0.925771 seconds.
Finding Correspondences
```

```
Elapsed time is 1.345949 seconds.
Computing RANSAC
Elapsed time is 0.043988 seconds.
Finding Correspondences
Elapsed time is 1.282209 seconds.
Computing RANSAC
Elapsed time is 0.038617 seconds.
Finding Correspondences
Elapsed time is 1.376021 seconds.
Computing RANSAC
Elapsed time is 0.034408 seconds.
Elapsed time is 0.034441 seconds.

locationIndex =

     2

Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
Waiting for image-is-ready signal
```

Figure [10]: Server End Processing

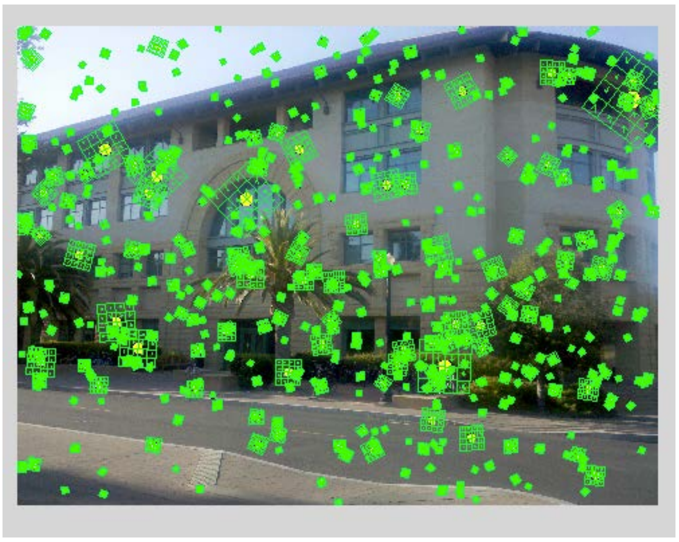


Figure [11]: Gates Image showing the SIFT calculated image features and descriptors

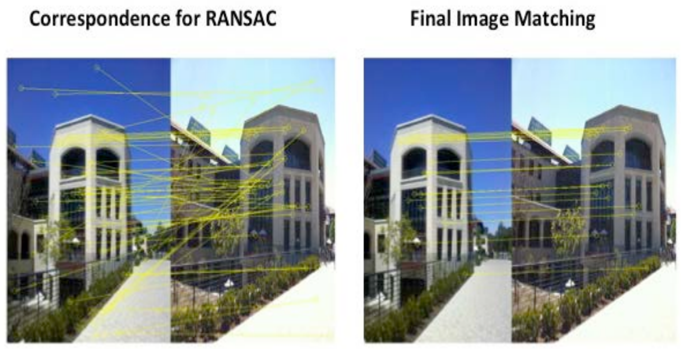


Figure [12]: Pre and Post Matching Images

The speed of our algorithm proved to be a considerable issue. SIFT and hierarchical means averaged 87secs and 157sec when setting up the database. During image matching, the RANSAC algorithm proved to be the bottleneck of the algorithm, taking a average of 50sec for each image of the database that the query image was compared to. Our major focus was to make the algorithm more precise than efficient and we were successfully able to accomplish this.

V. Conclusions

Navigating in a known or unknown region can prove to be a very huge undertaking that can either be a tedious or enjoyable part of anyone's journey. With the improvement in technology and the advent of camera phones, intelligent, interactive, efficient and robust augmented navigation systems are becoming a highly researched and sought after application. Our project implemented a section of this solution by successfully matching query images against a harvested database and providing the user with information of their location and destination. The scope of our project encompassed the Stanford campus. However, future improvements to our system will look to increase the database to other locations. Although speed proved to be a handicap of our implemented system, high matching statistics verified the feasibility and functionality of this system. Preliminary research and testing was conducted to explore the potential of image segmentation in enhancing the user experience. These results have been documented. However, further literature review needs to be conducted to determine the actual feasibility of implementing this system into our project. Overall, this project serves as a solid first step in the grand scheme of an augmented reality navigation system. There is room for improvement by way of increasing the size and scope of the database, the speed of the algorithm, hardware application base and overall GUI.

Acknowledgment

Thank you to our project mentor Vijay Chandrasekhar for providing advice and guiding us in the right direction, our TA David Chen for explaining many confusing concepts and providing a sounding board for our ideas, and our instructor Professor Bernd Girod for teaching us the algorithms used in this project.

References

[1] Girod, B., "Course Notes," EE368 – Digital Image Processing, Spring 2012

[2] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. (references)

[3] Lowe, D.G., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, November 2004, pp. 91-110

[4] Lowe, D.G., "Object Recognition from Local Scale-Invariant Features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999.

[5] Mina Makar, "EE368 Project Report – CD Cover Recognition Using Modified SIFT Algorithm", Spring 2006.

[6] Hoffman, Kimball & Russell "EE368 Project Report – Identification of Paintings in Camera-Phone Images.", Spring 2007.

[7] Cooper, Hwang, "EE368 Project Report – IMRESTAURANT: Matlab for Feature-based Restaurant Logo recognition", Spring 2010.

[8] Halawany et al, "Detection of Road Curb From Mobile Terresstial Laser Scanner Point Cloud."

[9] H. Hile et al., "**Landmark-Based Pedestrian Navigation from Collections of Geotagged Photos**," Proc. ACM Int'l Conf. Mobile and Ubiquitous Multimedia (MUM), ACM Press, 2008.

[10] T. H. Kolbe, "Augmented Videos and Panoramas for Pedestrian Navigation," Proc. 2nd Symp. Location Based Services and TeleCartography, G. Gartner ed., Geowissenschaftliche Mitteilungen, 2004.

[11] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346--359, 2008

[12] http://www.aishack.in/2010/05/sift-scale-invariant-feature-transform/

APPENDIX

Uchechukwuka Monu and Matt Yu both made significant contributions to the project proposal, algorithm development, evaluation, image collection, poster, and final report. Uchechukwuka focused on performing a literature review on related material and developing the code in MATLAB final report while Matt handled the server setup and Android development.