Proceedings of The National Conference
On Undergraduate Research (NCUR) 2016
University of North Carolina Asheville
Asheville, North Carolina
April 7-9, 2016

How your phone recognizes your home: An investigation of mobile object recognition

Thanh Vu and Daniel Piros
Department of Computer Science
Lafayette College
Easton, Pennsylvania 18042 USA

Faculty Advisor: Dr. Amir Sadovnik

Abstract

Often what is effortless for a human brain challenges machines the most. Visual recognition, a fairly easy task for humans, can be surprisingly difficult for machines due to variations in angle, size, and lighting. The challenge is amplified on mobile platforms because of computational constraints. There have been a number of studies on image recognition, but few focus on algorithms that run completely on portable devices. This work presents an improved image retrieval method that can run on mobile devices in real time without the need to access a remote server. First, the speed and accuracy of different known keypoint detectors and descriptors were studied to select the best one. Then, the results were further optimized by filtering best matches, exploiting the user's location, and extending a grayscale descriptor to include color. The algorithm successfully matched various objects to locally stored sample images with an improved accuracy of 98.5% in less than a second. In addition, since no well-structured database was publicly available, new data sets comprising hundreds of photographs of college buildings and academic posters taken from a combination of distances and angles were built. These data sets will be made publicly available. Finally, an application in the form of an electronic tour guide is presented, where users instantly gain detailed information on buildings or posters by taking pictures of them with their phones or tablets. Although this work focused on images of buildings and posters, the algorithm could potentially be used in other image recognition algorithms.

Keywords: Image Recognition, Mobile Devices, Building, Poster

1. Introduction

1.1. Objective

With the advance of computer technology, increasingly more human tasks are now digitized and assisted by software in mobile devices. This research, building on this trend, aims to investigate a method for image retrieval on portable devices. More specifically, the algorithm should be able to recognize an image of a building or a poster from its picture taken by a user's smartphone or tablet. Recognition is done by retrieving a training image of the specific building/ poster from the database. Because everything is done offline on the device, it is necessary to carefully consider the size of the descriptors, the runtime, and the accuracy of different retrieval algorithms in order to cope with the computational and memory constraints. Additionally, the retrieval method was implemented in a tour guide app for Android devices.

Section 2 describes the three main jobs of the system, which are essentially detecting keypoints, extracting descriptors, and matching descriptors. Several methods of filtering matches developed in order to improve the algorithm's performance are also described. Section 3 presents the detailed statistics of each algorithm in terms of its runtime and accuracy. Section 4 provides the final implementation of the application and future work.

1.2. Related Work

A common approach to object recognition is to use visual descriptors. The general idea is to represent images of objects with sets of signature descriptors that are invariant to scale, rotation, and lighting. An algorithm then uses these descriptors to compare and retrieve different known objects. Two of the most robust, efficient, and well-known methods are SIFT[1] (Scale-Invariant Feature Transform) and SURF[2] (Speeded-up robust features). These descriptors, however, require large computational power, which challenges their implementation in real-world applications. Many researchers proposed to solve this with binary descriptors such as BRIEF,[3] ORB,[4] BRISK,[5] and FREAK.[6] Several studies show that the binary approach can produce comparable retrieval efficiency with significantly higher speed.[7][8][9][10][11] This suggests greater potential for real-time applications, especially on mobile devices where computational power is usually limited.

As smartphones have become more ubiquitous and are getting more powerful for the last 20 years, many researchers focused their works on mobile platform. In 2000, Keith Cheverst et. al. presented the GUIDE system,[12] an electronic context-aware tourist guide that receives dynamic and positioning data via a wireless communications infrastructure. A field study on user reaction of tour guide app using GUIDE system was later done by Nigel Davies et. al. in 2005.[13] In 2003, Gausemeier and Bruederlin described a method of object recognition and tracking for augmented reality device.[14] The system relies on a database of 3D-models to determine the location and orientation of live objects. Within the last decade, there have been increasingly more studies that target landmark recognition directly, such as [15], [16], [17], [18], [19], and [20]. Some focus more on the recognition task itself while others also suggest tour guide applications for portable devices such as mobile phones or Google Glass.

This research extends the previous works in two ways. First, our software can be used for both buildings and posters, including academic posters, which are more challenging to retrieve due to large amount of text. Second, the system is implemented in an Android app and is capable of working offline without the need to send data to a remote server for computation.

2. Methodology

As mentioned above, the entire retrieval process is done locally on the device without involving any server. The application needs to carry out two main tasks – building the training library and identifying the query object. During the former, descriptors of all training images in the library are computed, and stored as files on the device. This task only needs to be done once. After that, whenever the application runs, the training library can be loaded directly from those files. The latter, image recognition, is executed at the user's command. When an image is captured, its descriptors are computed using the same algorithm used to build the tour's library. Then, each query descriptor is matched to a corresponding descriptor of a training image in the library. Different filters are also applied to improve the matching results. To retrieve the matching results, the matched descriptors are labeled by the training images they were extracted from. The best-matched image, which is the one that has the highest number of matched descriptors after being filtered, is returned as the result of the retrieval process. Once the image has been recognized, certain information, which can be attached to these training images, can be displayed to the user.

2.1. Detecting Keypoints And Computing Descriptors

Recognizing an object captured in a picture is quite easy for humans but relatively difficult for machines. Parts of the reasons are due to geometric transformations such as viewpoint change, scaling, and rotation. Pictures of the same object can have various sizes and shapes if taken at different angles and distances. Other challenges include change of lightning conditions and partial occlusions. A description needs to be found which can match the distinct features of an image with that of known objects in the database. The goal of this research is to make the image retrieval as robust as possible for specific objects, buildings and posters. Several detectors and descriptors were examined in order to find the most suitable one. The main process starts with a detector trying to detect the locations of important features in a query image such as corners. Then, a descriptor tries to describe these keypoints in a way that is invariant to lighting, scale, and rotation. Finally, a matcher, which is discussed in more details in Section 2.2, tries to match the query descriptors with the training ones to retrieve the correct object. This research focuses on binary descriptors, which are bit strings, for example, of length 32 as in ORB. Three different combinations of detectors/descriptors are investigated: ORB detector/ descriptor, FAST detector/ ORB descriptor, and BRISK detector/ descriptor. As mentioned in Section 1.2, these descriptors are more favorable for mobile devices since they usually require less computation, less memory, and operate much faster than SIFT and SURF.

2.1.1. FAST detector

FAST[21] is a high-speed corner detector based on the Accelerated Segment Test (AST). The algorithm looks at a circle of 16 pixels in the neighborhood of a candidate point *p* to determine whether the point is a corner. Point *p* is detected as a corner or a feature if there exist n contiguous pixels in the circle that are all brighter than $Ip + t$ or darker than $Ip - t$, where *Ip* is the intensity of point *p* and *t* is predefined threshold. The authors also use a machine learning approach with decision trees and non-maximum suppression, resulting in significant improvement in both the speed and the quality of detection process.

The combination of the FAST detector and the ORB descriptor has been being among the most commonly used due to the fact that ORB itself is developed from FAST keypoints. Thus, the result of this combination was also studied to compared to that of other algorithms.

2.1.2. ORB detector/ descriptor

ORB[4] is a combination of detector and descriptor that was created based on FAST and BRIEF (Binary Robust Independent Elementary Features).[3] It was developed by Ethan Rublee et al. in 2011 for real-time applications, aiming to outperform SIFT and SURF in terms of runtime but still possess equivalent matching efficiency.

The authors of ORB use FAST to detect the keypoints, apply a scale pyramid of the image to produce multiscale features, and filter the top N points using a Harris corner measure.[22] Unlike the original FAST, the ORB detector also uses Robin's intensity centroid[23] to calculate corner orientation of the features. After the feature points are found, they extract descriptors with a modified version of BRIEF. Since BRIEF is originally dependent on the direction, ORB generates the steered BRIEF descriptors that are invariant to rotation using the orientation of the identified features. The final descriptors are binary strings created by comparing the intensities around the keypoints.

2.1.3. BRISK detector/ descriptor

Similar to ORB, BRISK[5] (Binary Robust Invariant Scalable Keypoints) is a binary keypoint detector/ descriptor inspired by the AGAST detector,[24] an accelerated version of FAST, and the BRIEF descriptor. To solve the scale invariance issue of FAST and AGAST, the BRISK detector searches for corners not only in the image plane but also across scale dimensions. It detects points of interest both in octave layers of the scale-space image pyramid and in layers in-between. Additionally, the algorithm examines the characteristic direction of the keypoints to obtain rotation invariance. Afterwards, the authors apply a sampling pattern of local points on concentric circles around each keypoint and compare the intensity values to generate a bit-string descriptor of length 512. Compared to high-performance algorithms like SIFT and SURF, BRISK offers comparable matching accuracy while using much less computation, resulting in significantly faster runtime.

2.2. Matching Descriptors

Since all the algorithms examined use binary descriptors, the matching for all combinations is done using the Hamming distance – the number of bits differing between two descriptors. For each query descriptor, a training one is called a match if its binary code is the most similar to that of the query. In other words, it would have the smallest Hamming distance to the query descriptor compared to other training descriptors. This search is done using brute-force matching, which means comparing the query descriptor with each and every training one. This is a simple and fairly good method for a small database of descriptors, with a complexity of O(N), where N is the number of descriptors.

2.3. Optimizing The Recognition

2.3.1. resizing input images

Before the features are detected, all the input images, including both the training and the query, are scaled down to no more than *300 × 300* pixels. There are two reasons for this additional adjustment. First, by shrinking the pictures, irrelevant features can be eliminated. Since the main object of this application is building recognition, important features that define them are expected to be clear and easy to identify, while irrelevant ones tend to be small and insignificant. Thus, reducing the pictures' size helps remove the noise while still preserving the distinctive keypoints of the buildings. Second, lower resolution means less keypoints in total. As long as the number of features is large enough to identify the buildings, a smaller set of keypoints helps lessening the program's runtime and memory needs, which is critical for a mobile phone's application.

2.3.2. filtering matched results

To make the application more robust and suitable for buildings, two filters are used consecutively. Both of these filters are applied to matched images returned by the descriptor matcher. First, the location filter, as the name suggests, takes advantage of the buildings' and user's locations to filter out the best matches. The program only compares query buildings with training ones that are 50 meters away or closer. Buildings further away are expected to be negative matches, and thus, filtered out.

Second, in the Best vs. Second Best filter (BSB), the relationship between the best and the second best matched buildings is taken into account to determine if the retrieving result is reliable. The software calculates the number of matched descriptors and names it as either *Best* or *2ndBest* based on what building it belongs to. Then, the difference (*Best – 2ndBest*) and the ratio (*Best – 2ndBest)/ Best* is considered. A best match is a good match if its statistic satisfies the inequality:

$$\text{Difference} * \text{Ratio} > \text{Threshold} \tag{1}$$

That is

$$(\text{Best} - \text{2ndBest}) \wedge 2 > \text{Threshold} * \text{Best} \tag{2}$$

Based on experimental data, the default Threshold is chosen to be *5*, which is fairly efficient, especially when combined with location filter.

2.3.3. using color descriptor

Using a color descriptor is an additional method for improving the performance of the program particularly on academic posters' recognition. The binary descriptors described in Section 2.1 are extracted from grayscale images, which therefore describe a feature based on only the intensities of related pixels. When the objects of interest are academic posters, the images involve a great number of alphabet characters, which make it difficult for a machine to distinguish these posters. Thus, the binary descriptors were modified to exploit colors to identify images.

To do so, three one-channel images are created from the input image, using Red, Green, and Blue values of its pixels. The feature points are still extracted normally from the original image. This collection of keypoints is then used to generate Red, Green, and Blue descriptors from the one-channel images. In other words, every keypoint has three specific descriptors for its color channels. These descriptors are then concatenated to create one final color descriptor for that point. To increase the speed of computation and save memory, the length of each color descriptor is reduced from 32 to 16. Thus, the color descriptor after concatenation is of length 48. The new descriptor especially helps match academic posters since many of the differences depend on the colors.

3. Data and Discussion

3.1. Database And Setup

For the experiments, data sets of buildings and posters should cover a reasonable range of angles and distances that a user may take pictures from. Because no accessible database could be found that match our requirements, two data sets of Lafayette buildings and Lafayette posters were built. An additional data set of Caltech buildings was also used because of its availability, despite its lack of structure.

3.1.1. lafayette buildings

The main database used for evaluating the application's performance is a set of pictures of Lafayette College's buildings. Each building has 20 pictures captured from 5 angles and 4 distances. There are 10 different buildings, together making up a data set of 200 images. The different angles and distances are illustrated in Figure 1. Particularly, angle 0 refers to images taken directly in front of the buildings, while distance 0 indicates the distance where the buildings can fit mostly, if not entirely, into the pictures with the minimum amount of unnecessary background. Out of the 200, the 10 images of angle 0 and distance 0 are selected to be the training images. The query ones involve the other 190 pictures. The original resolution of these pictures is *4128 × 2322*, but then reduced to *300 × 168* inside the application as mentioned in Section 2.3.1. Each image file also had the GPS location from which the image was taken.

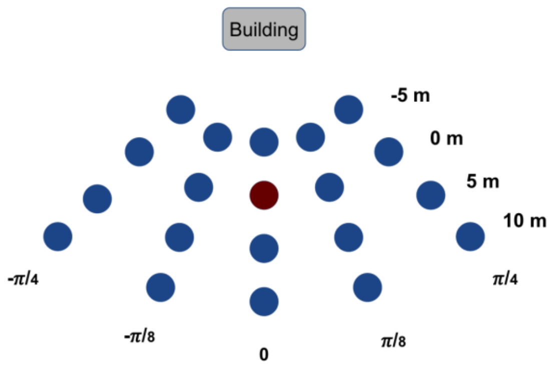

Figure 1. Perspectives of buildings.

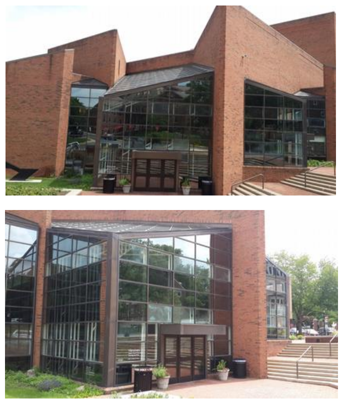

Figure 2. Example of Lafayette building

Figure 1. Each dot represents a position from which a photo of the building was taken. The image at the red dot, located at angle 0 and distance 0, is used as the training image while the rest are query images.

3.1.2. caltech buildings

This data set contains 250 pictures of 50 buildings of California Institute of Technology.[25] Each building has 5 images with some variations of angles and distances. One of the five pictures is selected to be the training image for each building, while the other four are used as query images. Although the dataset is not organized by distance and angle, it is used as another way to verify the effectiveness of our algorithm. It should also be noted that since the location data of its images of buildings are not available, the location filter could not be used. The images' resolutions are resized from *2048 × 1536* to *300 × 225*.

3.1.3. posters

There are two sets of posters, including on-campus events and academic posters. These databases were built using data collected around Lafayette College's campus. With posters as the objects, no location filter should be applied considering the fact that most of the posters in an event are often geographically near each other.

For the event poster data set, there are pictures of 20 objects, including 20 training images and 60 query images from 3 different angles. For the academic poster data set, the number of posters is 50 but each poster is only queried once. In other words, there are 50 training images and 50 query ones. All images are initially of size *4128 × 2322* and reduced to *300 × 168*.

3.2. Accuracy

With the filters described in Section 2.3, a query image might result in no match when a building is unrecognizable. Hence two numbers are reported to measure the effectiveness of the algorithm. Matching rate is the percentage of images that were identified correctly. Accuracy is the percentage of images that were not labeled incorrectly (unrecognizable buildings are not considered labeled). Thus matching rate measures how often the correct label is returned while accuracy measures how often a wrong answer is returned (vs. no answer or a correct one). Different combinations of feature detectors and descriptor extractors have been examined to find the most suitable for the application.

Insights of how the program behaves without the location filter and with no filter at all are also presented. It is necessary to consider the efficiency of BSB filter alone since there may be cases where images' locations are unavailable. Since all algorithms produce improvement when using the filter, this paper only shows the statistic for the FAST – ORB combination as an illustration.

3.2.0. overview of the results

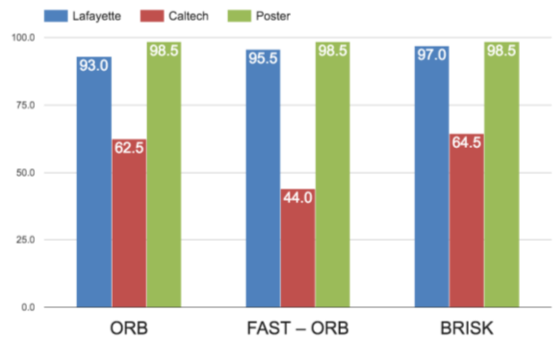

Figure 3. Matching Rate (%)

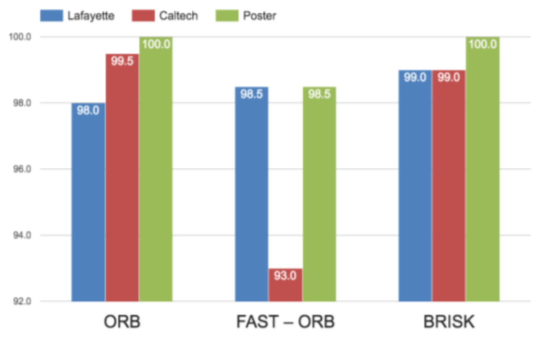

Figure 4. Accuracy (%)

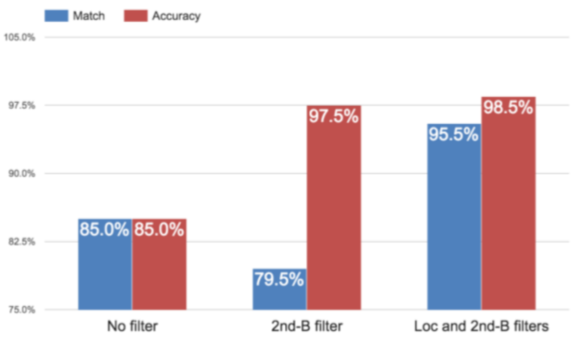

Figure 5. Improvement using filters (FAST - ORB)

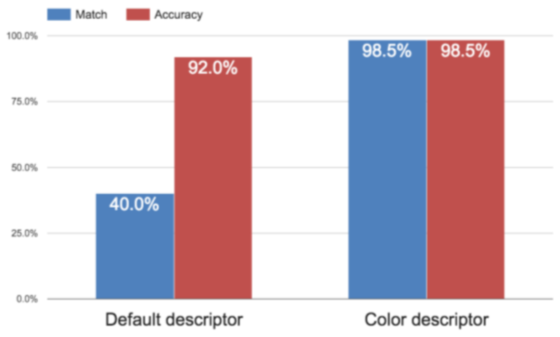

Figure 6. Improvement using color descriptors (FAST - ORB)

3.2.1. lafayette buildings

For the Lafayette building dataset, the accuracies are calculated and provided in Table 1 and 2 using the structure in Figure 7.

Angle	*- π/4*	*- π/8*	*0*	*π/8*	*π/4*	*Distance(m)*
						-5
			T			*0*
						5
						10

Figure 7. Accuracies at different positions

The row indicates the distance from the building while the column denotes the angle from the front view. The training image is at position T. The other positions are the query images and have the distances and the angles relative to T.

Table 1. Improvement of the accuracies when using filters with FAST - ORB (%)

(%)	No filter					BSB filter					Location and BSB filters				
Accuracy	90	90	100	90	70	100	100	100	100	100	00	100	100	100	100
	100	100	100	90	90	100	100	100	100	90	100	100	100	100	100
	100	100	100	70	70	100	100	100	90	90	90	100	100	100	100
	70	90	90	60	30	100	100	100	90	90	90	100	100	100	90
Avg Acc.	85.0 %					97.5 %					98.5 %				
Matching Rate	90	90	100	90	70	100	100	100	80	50	90	100	100	100	90
	100	100	100	90	90	100	100	100	90	70	90	100	100	100	90
	100	100	100	70	70	90	100	100	70	50	80	100	100	100	90
	70	90	90	60	30	60	80	80	50	20	90	100	100	100	90
Avg M.R.	85.0 %					79.5 %					95.5 %				

Table 2. Accuracies of different algorithms (%)

(%)	ORB – ORB					FAST – ORB					BRISK – BRISK				
Accuracy	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
	100	100	100	100	90	100	100	100	100	100	100	100	100	100	100
	90	100	100	100	90	90	100	100	100	100	90	100	100	100	100
	90	100	100	100	100	90	100	100	100	90	90	100	100	100	90
Avg Acc.	98.0 %					98.5 %					99.0 %				
Matching Rate	80	90	100	100	90	90	100	100	100	90	100	100	100	100	100
	90	100	100	90	90	90	100	100	100	90	100	100	100	100	100
	80	100	100	100	90	80	100	100	100	90	90	100	100	100	100
	80	100	100	100	80	90	100	100	100	90	90	100	100	100	90
Avg M.R.	93.0 %					95.5 %					97.0 %				

3.2.2. caltech buildings (no location)

Table 3. Improvement of the accuracies when using filter with FAST - ORB (%)

(%)	No filter	Location and BSB filters
Accuracy	48	93
Matching rate	48	44

Table 4. Accuracies of different algorithms (%)

(%)	ORB – ORB	FAST – ORB	BRISK – BRISK
Accuracy	99.5	93	99
Matching rate	62.5	44	64.5

3.2.3. posters with color descriptors

Table 5. Improvement of the accuracies when using color descriptors with FAST - ORB (%)

(%)	Default descriptors (BSB filters)	Color descriptors (BSB filters)
Accuracy	40	98.5
Matching rate	92	98.5

Table 6. Accuracies of different algorithms (%)

(%)	ORB – ORB	FAST – ORB	BRISK – BRISK
Accuracy	100	98.5	100
Matching rate	98.5	98.5	98.5

3.3. Timing

The time to query an image was measured using an Android phone. Comparisons were made for both Lafayette and Caltech data sets. Generally, the ORB - ORB combination is faster than the other two. Especially in the case of Caltech buildings, its runtime is 3 times faster than FAST - ORB's and 4 time faster than BRISK - BRISK's.

The detection time was also recorded for color descriptors with Poster data set. The ORB keypoints and ORB descriptors remain the fastest combination of algorithms. Unfortunately, the time for computation is considerably higher than that of the original binary descriptors. This seems to be partly due to the greater length of the descriptors. The detailed statistics are displayed below.

(ms)	ORB – ORB	FAST – ORB	BRISK – BRISK
Lafayette	844	1253	1071
Caltech	436	1385	1771
Posters (color descriptor)	1704	2133	2118

Table 7. Runtime of algorithms with different data sets (ms)

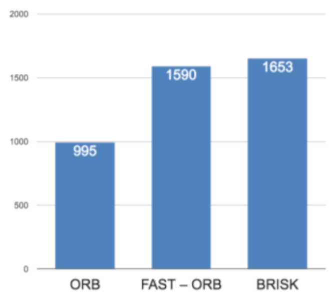

Figure 8. Average runtime (ms)

4. Application Outcome

4.1. Implementation On Android Devices

The application is implemented using Java and OpenCV. Its source code can be found online at OpenCVTour project on Github.[26] Based on the data from Section 3, the ORB detector/ descriptor was chosen to be implemented.

When the user builds the training library, descriptors are computed for all images in the database. These features are then saved as YAML files inside the device for matching images later. This helps avoiding repeated computation of training descriptors. Under the user's command, an image of a tour item, which can be a building or a poster, is queried. After extracting its features, the program decides whether that tour item matches with any from the library and displays the result to the user.

4.2. Future Work

In our program, the runtime of retrieving images is O(N), with N is the size of the training library, due to brute-force matching. That means the speed of the application is strictly related to the volume of its database. Therefore, the application is more suitable for small sets of data. For it to work efficiently with large data sets, other matching algorithms could be examined. Additionally, the color descriptor has a strong potential for improving image retrieval because it takes color factor into account.

5. Conclusion

With this research, different descriptors were compared in order to find the best performance for building and poster retrieval. Additional optimization methods have also been investigated to improve the program's speed and accuracy. The algorithm was successfully implemented in an application called OpenCVTour for Android devices. The app offers users a tour-organizing tool that utilizes building recognition. It uses the ORB features with our optimization methods, including resizing images and filtering matching results.

6. Acknowledgements

The authors wish to express their appreciation to Lafayette College's EXCEL Research program.[27] They also express their appreciation to faculty research advisor Dr. Amir Sadovnik for his guidance throughout this study.

7. References

1. D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision 60(2) (November 2004):91-110.
2. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," Computer Vision and Image Understanding 110(3) (June 2008): 346-359.
3. Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua, "Brief: Binary robust independent elementary features," ECCV'10 Proceedings of the 11th European Conf. on Computer Vision, (2010): 778–792.
4. Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. "Orb: an efficient alternative to SIFT or SURF" (paper presented at 2011 IEEE Int. Conf. on Comp. Vision (ICCV), Barcelona, November 6-13, 2011).
5. Stefan Leutenegger, Margarita Chli, and Roland Yves Siegwart, "Brisk: Binary robust invariant scalable keypoints" (paper presented at IEEE Int. Conf. on Computer Vision (ICCV), Barcelona, November 6-13, 2011).
6. Alexandre Alahi, Raphael Ortiz, Pierre Vandergheynst, "FREAK: Fast Retina Keypoint" (paper presented at 2012 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Providence, RI, June 16-21, 2012).
7. O. Miksik, K. Mikolajczyk, "Evaluation of local detectors and descriptors for fast feature matching," (paper presented at 2012 21st International Conference on Pattern Recognition (ICPR), Tsukuba, Nov. 11-15, 2012).
8. H.-C. Sperker, A. Henrich, "Feature-based object recognition — A case study for car model detection" (presented at 11th Int. Workshop on Content-Based Mult. Indexing (CBMI), Veszprem, June 17-19, 2013).
9. Canclini, A.; Cesana, M.; A. Redondi, M. Tagliasacchi, J. Ascenso, R. Cilla, "Evaluation of low-complexity visual feature detectors and descriptors" (presented at 2013 18th International Conference on Digital Signal Processing (DSP), Fira, July 1-3, 2013). Figure 1. Perspectives of buildings. Figure 1. Perspectives of buildings.
10. Akash Patel, D. R. Kasat, Sanjeev Jain, V. M. Thakare, "Performance Analysis of Various Feature Detector and Descriptor for Real-Time Video based Face Tracking," Int. Journal of Comp. App. 93(1) (May 2014): 37-41.
11. Şahin Işık, Kemal Özkan, "A Comparative Evaluation of Well-known Feature Detectors and Descriptors" (presented at he Int. Conf. on Advanced Tech. & Sciences (ICAT'14), Antalya, Turkey, August 12-15, 2014).

12. Keith Cheverst , Nigel Davies , Keith Mitchell , Adrian Friday , Christos Efstratiou, “Developing a context-aware electronic tourist guide: some issues and experiences,” (presented at Proceedings of the SIGCHI conference on Human factors in computing systems, The Hague, The Netherlands, April 1-6, 2000).
13. Nigel Davies , Keith Cheverst , Alan Dix , Andre Hesse, “Understanding the role of image recognition in mobile tour guides” (presented at Proceedings of the 7th international conference on Human computer interaction with mobile devices & services, Salzburg, Austria, September 19-22, 2005).
14. J. Gausemeier, J. Fruend, C. Matysczok, B. Bruederlin, and D. Beier, “Development of a real time image based object recognition method for mobile AR-devices” (presented at AFRIGRAPH '03 Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa, 2003).
15. J. Hu, J. Sawyer, and J.-Y. Herve, “Building detection and recognition for an automated tour guide. In Systems” (presented at Man and Cybernetics, Taipei, Oct. 8-11, 2006).
16. J.-H. Lim, "Scene Recognition with Camera Phones for Tourist Information Access" (presented at 2007 IEEE International Conference on Multimedia and Expo (ICME 07), Beijing, July 2-5, 2007).
17. Y. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neve. “Tour the world: Building a web-scale landmark recognition engine” (presented at IEEE Conference on Computer Vision and Pattern Recognition (CVPR 09), Miami, FL, June 20-25, 2009).
18. Y. Li, D. J. Crandall, and D. P. Huttenlocher, “Landmark classification in large-scale image collections” (presented at 2009 IEEE 12th International Conf. on Computer Vision, Kyoto, Sept. 29 - Oct. 2, 2009).
19. Hani Altwaijry, Mohammad Moghimi, and Serge Belongie, “Recognizing locations with google glass: A case study” (paper presented at IEEE Winter Conference on Applications of Computer Vision (WACV), Steamboat Springs, Colorado, March 24-26, 2014).
20. Theophile Dalens, Josef Sivic, Ivan Laptev, and Marine Campedel, “Painting recognition from wearable cameras,” HAL (2014): 13, https://hal.inria.fr/hal-01062126/document
21. Edward Rosten, Reid Porter, and Tom Drummond, “Faster and better: A machine learning approach to corner detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol:32, issue:1, (2010): 105–119.
22. C. Harris and M. Stephens, “A combined corner and edge detector” (presented at Alvey Vision Conf., 1988).
23. P. L. Rosin, “Measuring corner properties,” Comp. Vis. and Image Understanding 73(2) (Feb. 1999): 291-307.
24. E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, “Adaptive and generic corner detection based on the accelerated segment test” (presented at the European Conference on Computer Vision (ECCV), 2010).
25. Caltech buildings, last modified June 29, 2011, http://www.vision.caltech.edu/malaa/datasets/caltech-buildings/
26. OpenCV Tour, last modified Sep. 14, 2015, accessed May 27, 2016, https://github.com/WriterOfAlicrow/OpenCVTour
27. EXCEL Scholars Program, Lafayette College, accessed May 27, 2016, https://academics.lafayette.edu/undergraduate-research/excel-scholars-program/