Mirror Detection Using A Smartphone Camera

by

Raja Revanth Palepu

A Project Report Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by

Dr. Joe Geigel

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

May 2017
Dedication

This project is dedicated to my mother who will always be the prominent reason for who I am today and I will forever be in debt to her. I would also like to dedicate this project to my family and friends for being there and for being an influence in my life in various ways.
Acknowledgments

I would like to take this opportunity to thank my advisor, Professor Joe Geigel. It was his idea to pursue this, a first of its kind project as my capstone. His guidance, assistance and timely input were a major part in the developing of this project. I would also like to thank my colloquium in-charge, Professor Reynold Bailey for providing useful suggestions to make the report, project design and the poster. I am thankful to the CS department at RIT for offering most productive coursework, their helpful resources. I am forever grateful to all the professors in charge of the courses that I had taken for imparting their knowledge.

Finally, I would like to thank my friends for their consistent support and words of encouragement throughout my Master’s degree pursuit.
Detection of mirrors can be problematic if sophisticated equipment or setup is not present like a LIDAR (LIght Detection And Randing) system [11]. This project is a smartphone based lightweight application that detects the presence of a mirror in real time which is achieved by finding unique keypoints in the scene by obtaining frames from the front and back cameras of the user’s smartphone and finding unique keypoints using SIFT (Scale-Invariant Feature Transform) descriptor [9]. The keypoints and descriptors that are generated for the respective frames are then matched based on a threshold condition. If the condition is satisfied, the application triggers a notification that the mirror is present in the scene. The primary purpose of the application is to enhance the experience of capturing mirror selfies.
Contents

Dedication .......................................................... ii

Acknowledgments ................................................... iii

Abstract .............................................................. iv

1 Introduction ....................................................... 1
   1.1 Background .................................................. 1
   1.1.1 Mirror Detection ......................................... 2

2 Design and Implementation ..................................... 3
   2.1 Hardware and Software ....................................... 3
   2.2 Simultaneous Camera Display ............................... 4
   2.3 Frame Conversions ........................................... 5
   2.4 Feature Detection and Description ......................... 5
      2.4.1 SIFT(Scale-Invariant Feature Transform) .............. 5
      2.4.2 SURF(Speeded Up Robust Features) .................... 6
      2.4.3 ORB(Oriented FAST and Rotated BRIEF) .............. 6
   2.5 Feature mapping ............................................. 7

3 Analysis ........................................................... 8
   3.1 Using SIFT .................................................. 9
   3.2 Using SURF .................................................. 9
   3.3 Using ORB ................................................... 10

4 Limitations ....................................................... 13
   4.1 Hardware .................................................... 13
   4.2 Software ..................................................... 13
   4.3 Light, Position and Angle ................................... 14
## 5 Conclusions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Current Status</td>
<td>15</td>
</tr>
<tr>
<td>5.2 Future Work</td>
<td>15</td>
</tr>
<tr>
<td>5.3 Lessons Learned</td>
<td>16</td>
</tr>
</tbody>
</table>

**Bibliography**

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>System Design</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>System Design</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>When the mirror is detected</td>
<td>9</td>
</tr>
<tr>
<td>3.2</td>
<td>When the mirror is not detected</td>
<td>10</td>
</tr>
<tr>
<td>3.3</td>
<td>SIFT test case</td>
<td>11</td>
</tr>
<tr>
<td>3.4</td>
<td>SURF test case</td>
<td>11</td>
</tr>
<tr>
<td>3.5</td>
<td>ORB test case</td>
<td>12</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Mirrors are utilized in day to day lives for various purposes such as to view one’s reflection, driving, dentistry, curved mirrors in building and so forth. There are dedicated systems and setups that are used to detect mirrors in a given scenario. What if there was a way to detect mirrors using just the hardware that is present on a smartphone? As we are aware that ownership of a smartphone is increasing day by day around the world. Detection of a mirror is a generic problem that does not have a simplistic solution. Developing a common solution to this issue can be applied to various problems like Facebook AI system [8] that is trying to determine if an image is a mirror selfie or not and tracking if a driver is looking into the rearview mirror of the vehicle. This project is a lightweight application that can detect a mirror in a real world scenario. This solution is targeted to aid users with capturing mirror selfies. Additionally, with future work and further enhancement, the application could be applied to the problems as mentioned earlier.

1.1 Background

Mirror detection can be a complicated and laborious task in the field of Computer Vision. No system so far is considered to be the most efficient and effective in detecting mirrors consistently. There are various factors involved in detecting a mirror. An add-on burden is when the detection needs to be done in real-time. Some of the factors need to be taken into consideration are:

- **Lighting** - If the image or the real-time scene does not have the right amount of light
then this could affect the image preprocessing steps and so forth. Excess or scarcity in light can affect the resulting image.

- **Surface type** - There is a possibility that there could be various reflective surfaces and some may be partially reflective. This could result in some false positive results. This in turn affects the performance of the system.

- **Cost** - Depending on the requirement, the cost may vary. One can have a system entirely dedicated to detecting mirrors, but the price for this framework could be high. Also, updates and changes to a system like this can affect the budget and could be complex.

### 1.1.1 Mirror Detection

Although there have been some vital systems that detect mirrors or mirror-like surfaces. These systems involve special hardware setup and a system like LIDAR has had widespread use. A system involving projector-camera [4] setup where a sinusoidal pattern is projected onto target scene. The camera captures the resulting images where one can notice that the other objects or surfaces that are not reflective tend to be illuminated. These lit up surfaces can be separated from the image to detect the reflective surfaces that are present. The downside to this system is, a projector-camera combination setup is required and this cannot be done solely using the hardware components on a smartphone.

Another sound solution for detective reflective surfaces or mirror is the LIDAR (LIght Detection And Ranging) system [11]. This system involves a using a robot with LIDAR system to detect mirrors and user the respective information along with a Bayesian framework to map and localize. This way, the reflections can be avoided and the mirror or mirror-like objects could be tracked continuously. The disadvantage of this system is the complete LIDAR setup. A sound LIDAR system could be expensive and this work also showed that it is flawed where it is entirely unable to detect and track reflective surfaces all the time.
Chapter 2

Design and Implementation

This project is possibly the first of its kind as there has not been any work in the past that had mirror detection as primary objective using a smartphone due to which there has not been much relevant work. The aim is to use the in-house hardware on a mobile device to detect the mirror in real-time. An overview of the final system design is illustrated in figure 2.1. Sections of the design and development can be split into the following sections:

2.1 Hardware and Software

Android operating system based smartphones are one of the most widely used mobile devices around the world. Android OS [1] is Linux kernel based and sound open source which makes it all the more advantageous for manufacturers to make devices based on this OS and for developers to make applications. As displaying both the cameras at the same time was one of the objectives of this project, Android devices running 4.4 version (KitKat) or higher was considered. It is common to have a smartphone now with a quad-core processor, the processor of choice is a Qualcomm Snapdragon 801 processor. The minimal primary storage considered on-board is 16 Gigabytes and 2 gigabytes of RAM to aid the processor. Android Studio is the main IDE that is used to develop Android applications. OpenCV [7] library 3.2 is imported and utilized by the application for the image processing, feature extraction and so forth.
2.2 Simultaneous Camera Display

Android’s Camera API [2] is used to handle the front and back facing camera on the device. Two SurfaceView objects are assigned to the respective Camera objects that control both the cameras. SurfaceView [3] is the class that is used for display a live preview feed from the camera. The purpose of showing the both the camera’s live feed to understand how the output from both cameras is similar to an extent if a mirror is present in front of the user.
2.3 Frame Conversions

One frame from each camera is obtained. Since front facing camera acts as a mirror, this frame needs to flipped across the y-axis so that the image is eligible for matching and mapping the key features from both the pictures. The frames are initially of byte() data type, they are converted to the OpenCV Mat format. This Matrix is of RGB format, so this is again converted to a grayscale Mat. The reason for performing these conversions is the feature descriptors that are used to extract and describe the keypoints in images work mostly with grayscale images.

2.4 Feature Detection and Description

Feature detection is the process of finding salient points of interests such as find corners and edges. These points are then provided a thumbprint or a signature that describe and differentiate them. This process of description vector generation is feature description.

2.4.1 SIFT(Scale-Invariant Feature Transform)

SIFT [9] is one of the most widely used feature descriptors. The essence of this descriptor is to find keypoints in an image that are unique and are not affected by scaling, rotation and translation. Additionally, SIFT is not affected by illumination. figure 2.2 shows how SIFT works. At first, the difference of Gaussian is obtained by creating a group of steadily gaussian blurred pictures. In the next stage, the minima and maxima in the scale-space are found. From these, the keypoints are picked. Any unwanted keypoints are not considered. Later, the gradient histogram is obtained for keypoints and the space around them. This histogram is used to construct the feature vector or the description vector for the keypoints.
2.4.2 **SURF(Speeded Up Robust Features)**

SURF [5] in general works in a similar manner as SIFT but faster this is because a Hessian-based blob detector is used by SURF to get the points of interest. The multi-scale analysis is performed to get the points that are scale and translation invariant. The HAAR wavelets responses in x and y-direction are used to obtain the orientation invariant keypoints.

2.4.3 **ORB(Oriented FAST and Rotated BRIEF)**

ORB [10] is one of the most recent feature descriptors. A few of the advantages of this descriptor is that it is open source and works faster than SIFT or SURF comparatively. At first the FAST(Features from Accelerated Segment Test) is used to obtain the keypoints. Since FAST does not work well with orientation, for each patch the intensity weighted centroid is calculated such that vector from the corner point of the patch to this intensity centroid provides the orientation. For feature descriptor, BRIEF(Binary Robust Independent Elementary Features) is used. Although the issue with BRIEF is, it does not work well when rotation is involved. To overcome this, the orientation vector from the modified FAST is used to steer the BRIEF descriptor in obtaining the descriptor vector.
2.5 Feature mapping

The features that are obtained from the feature descriptors for each binary image are matched either based on the Hamming distance or FLANN (Fast Library for Approximated Nearest Neighbors). Brute force matching using Hamming distance [6] is one of the most common and fast XOR function based methods used to match binary image features. Here, each descriptor is compared from one image to the other for finding matches or similarity between them depending on the number of bits that vary between the similar feature vectors. Whereas, FLANN makes use of hierarchical clustering tree or uses randomized kd-trees. The choice of tree is based on the features set.
Chapter 3

Analysis

The complete design and implementation phases went hand in hand. There were many changes due to the trial and error combinations of displaying camera previews with the image conversions and using the right feature descriptor. OpenCV’s camera preview components did not work well for showing both the camera’s live feed at the same time and this is where the Android’s SurfaceView comes into play. The SurfaceView objects run on GUI threads and these threads are used for displaying various views. The SurfaceView class is used for faster and continuous rendering because of which the Camera API uses the SurfaceView to show the live feed from any camera.

Although this simultaneous display of front and back-facing cameras works on paper, it does not run on Android devices that do not have the Snapdragon 801 processor. This strange issue will be explained why in the limitations section. Of all the combinations and various implementations were tried and tested, the following three are the important ones. The dual camera mode, input frame conversions are the same across all the three systems including brute force matching based on Hamming distance. The only difference is the feature descriptor that was used for extracting and describing keypoints. The final application results appearance in case of mirror detected is visible in Figure 3.1 and in case of not detected, in Figure 3.2.
3.1 Using SIFT

From Figure 3.3, one can observe that the number of features detected on both the input images are more than the minimum threshold requirement for a match. Most of the features were matched appropriately from one image to the other. Although, it is noticeable that some of the features tend to be mismatched and mapped, such as some of the keypoints on the wall or the user’s clothing is assigned onto the user’s skin keypoints. There could be various reasons behind this such as, orientation, illumination and intensity of that region may have had resulted in similar description vectors. However, SIFT still proved to have a significant number of suitable matches.

3.2 Using SURF

SURF is supposed to be faster and as good as SIFT and SURF does live up to the expectation of producing more features and good matches. Despite being faster and producing more keypoints, SURF when compared to SIFT, has more number of mismatched points mapped from one image to the other. The mismatched and mismapped features can be
Figure 3.2: When the mirror is not detected

seen in Figure 3.4 and could result in a higher number of false positives. Therefore, this implementation had to be dropped.

3.3 Using ORB

The initial choice for a feature descriptor was ORB. Some of the reasons being, ORB is faster, open source and was known to produce keypoints that matched the likes of SIFT and SURF. However, after implementing the application with ORB, the number of features that were generated and mapped from one image to another were less than the minimum threshold requirements to deduce that a mirror is present in the scene. Figure 3.5 shows a test case for this implementation and ORB never provided good results which were one of the reasons why the final implementation led to using SIFT as the feature descriptor.
Figure 3.3: SIFT test case

Figure 3.4: SURF test case
Figure 3.5: ORB test case
Chapter 4

Limitations

4.1 Hardware

Due to the main purpose of this project being a lightweight application that uses the hardware that is present in the mobile device, it eventually forces a limitation on the hardware. It is hard to produce more promising results by utilizing only the front and back facing cameras. Some Android phones have or had IR (infrared) blasters. This component could have been used to send IR signals which will be reflected back by the mirror. The reflected signal could be received which in turn could have been a way to detect the presence of a mirror in the front of the user.

Another component is found on some of the mobile devices is the small Laser emitter. This emitter is used for auto-focus during image capture. The emitter also could have been used to emit short-range laser beams which will bounce off the reflective surface and reflected beams could be received by the device. Additionally, the camera sensors also make a difference in the resulting image. Most of the front-facing smartphone cameras do not have an advanced sensor like the back-facing camera; this creates the problem of getting features that may not be matched and mapped well from one image to the other.

4.2 Software

Android’s robust, developer friendly framework at times can be a curse. There are multiple ways to show the camera preview i.e. either by using the OpenCV’s CVCameraView and JavaCameraView is used to show the camera preview. Although, this cannot be used to
show both the cameras previews at the same time because CVCameraView does not have
the ability to do so. The Android’s native SurfaceView can be employed twice as Surface-
View runs on the GUI threads, this can be exploited to show both the camera’s live previews
at the same time. Unfortunately, This cannot be implemented on devices that do not have
the Snapdragon 801 processor. One of the potential reasons behind this is, 801 processor
has two image signal processors built-in thus, enabling the usage of dual camera mode on
Samsung Galaxy S5 and HTC One M8 devices. Also, this makes it another limitation on
the hardware for this project to work.

4.3 Light, Position and Angle

The lighting conditions place a crucial role in capturing an image. A bad camera sensor in
an affecting lighting condition can produce a bad input image which will not yield a good
set of keypoints. So, light has an effect on this project’s performance. As for now, the
assumption is that the mirror is present in front of the user as the intention is to provide
a better mirror selfie capturing experience. There is scope in future for this application to
detect mirrors anywhere in the scene. The position and angle at which user is clicking the
mirror selfie will affect the performance of this project. Beyond a certain angle or position,
the number of features matched from both the images may not reach the minimum threshold
thus resulting in a false negative. This can be seen in the front and back image combination
in figure.
Chapter 5

Conclusions

5.1 Current Status

The overall project’s design and development stages have been a continuous learning process where in-depth analysis of SIFT, SURF and ORB was done. There have been many constraints that lead to the current application. As it was pointed in the implementation phase, a descriptor like ORB surprisingly underperformed when compared to the likes of SIFT and SURF. ORB found only a few points which were less than the required minimal threshold amount, but the problem was when the matched keypoints had to be mapped from one image to the other. For example, consider fig() a facial keypoint from the image on the left side was matched and assigned to the point on the right hand image’s user’s t-shirt button.

In the case of SURF, there are many outliers from one image that were mapped and matched to various points that are not the same. This was one of the primary reasons why SURF was not considered for the final development of the application as the unwanted mismatches would lead to many false positives. On the other hand, the application did not show signs of lag in the final implementation even when both the cameras previews were displayed simultaneously and one frame per two seconds was taken as input.

5.2 Future Work

As stated in the beginning, the project is a generalized approach to detecting the presence of a mirror in real time. Therefore, the application could be enhanced or modified for face
detection and recognition. Facebook’s AI system is unable to recognize or tag photos that are mirror selfies. As images can be captured using the Facebook’s smartphone application, this project’s idea could be added to the application. This way, if a user is capturing a mirror selfie, then the image could be categorized as a mirror selfie in hand.

There is scope for a smartphone to be a part of a robotic setup where the smartphone could be docked into a robotic structure that can move along with the user. The device could utilize this application to determine if a mirror is present in the scene. Additionally, some smartphones have IR blasters and Laser system for autofocus purposes. These can immensely improvise the current project in detecting a mirror in real time environment.

5.3 Lessons Learned

Developing this application showed how problematic mirror detection could be when there is limited hardware. This project demonstrated how arduous smartphone application development could get. The application development involved many trade-offs and changes in design and implementation. An interesting takeaway from the project was how SIFT is still leader amongst the feature descriptors. For this project, SIFT turned out to be the winner even against new choices like SURF and ORB. Finally, the project provided a sound knowledge of image processing and the three feature descriptors, SIFT, SURF and ORB.
Bibliography

[1] Android (operating system) - wikipedia.


[8] Jason Koebler. Facebook is trying to figure out how to automatically detect mirror selfies.

