IU Lost and Found: An Application of Computer Vision on Mobile Devices

Xiner Zhang
Indiana University Bloomington
1201 S. Adams St
Bloomington, Indiana
+(1)949-228-8063
zhang504@iu.edu

Shengyu Chen
Indiana University Bloomington
2501 E. Cargill Dr
Bloomington, Indiana
+(1)812-606-9239
chen435@iu.edu

Qian Feng
Indiana University Bloomington
SICE Luddy Hall,
Bloomington, Indiana
fengqian@indiana.edu

ABSTRACT
In this paper, we discover the possibility of Computer Vision can improve daily life of college students and faculties at IU. In order to do that, we design and implement a mobile application called IU Lost and Found. Retrieving a lost object is not an effortless experience, while IU have a large number of classrooms and multiple Lost and Found offices. Running around these offices on campus can take the entire day. Despite, most of the offices close earlier than the students and instructors get off classes. We want to solve this problem with IU Lost and Found. The main idea behind IU Lost and Found is to simplify the process of IU students or faculties' experience of retrieving their lost item by utilizing the characteristics of mobile devices and Computer Vision algorithm, including feature detecting and matching. With IU Lost and Found, they should be able to check and find their lost object in Lost and Found office with couple taps on their smartphones.

Categories and Subject Descriptors
H.3.1 [**Computing methodologies**]: Artificial intelligence: – *Computer Vision*

General Terms
Algorithms, Measurement, Performance, Design.

Keywords
Application, Computer Vision, Feature Matching, Mobile Devices

1. INTRODUCTION
This better never happen, but imagine one day you lose your key chain. You know it's somewhere at school but not sure where exactly did you lost it. Traditionally, you will have to search every room you have been to that day, and all the Lost and Found offices in these buildings. IU has a large campus. Doing so may take hours. And even if you look everywhere, there is still a possibility that you can't find it at all and all your effort paid to find it becomes meaningless.

There are some existing products on the market that attempts to solve this problem. For example, there are products allow you to attach additional tracking devices that have Bluetooth [1] or GPS build in to it. Attaching these devices to your personal belongings will increase the possibility of retrieving lost items. Although these products are able to tell the user an accurate location of their personal belonging, they are not always affordable and requires external devices. It also requires one device for each of your personal belongings, which is very inconvenient.

In this paper, we purpose a Lost and Found system, called IU Lost and Found, which target the students and faculties at IU. IU Lost and Found is able to simplify the retrieval of a lost belonging on campus. The main goals for IU Lost and Found includes:

- Integrating all found item inside of a database with their current locations.
- Able to accept image input of the lost item from the Android application.
- Match the lost item with the found items in the database and return the possible locations to user with minimum time and high accuracy.
- Avoid external devices with Bluetooth or GPS attaching to person belongings.

In this project, we design, implement and evaluate the IU Lost and Found on Android simulator. This application is able to find one or more matches of a lost item in the database of found items. We evaluate it within this paper while supporting computer vision algorithms that perform the following image tasks: image feature point detection, feature point matching, and matching optimization. Using these algorithms enables the IU Lost and Found to return a result of match found or not found. The core idea of the application is to compose image matching between one and multiple images with a high accuracy and high efficiency.

2. IU LOST AND FOUND OVERVIEW

In this section, we begin by systematically describing IU Lost and Found and the capability of Computer Vision on mobile devices. The main use-case of this application is to do image matching and return back results including if the match item is found in the database, and all the locations of the found items. Moreover, IU Lost and Found should return these result with minimum latency and high accuracy.

2.1 Existing Object Tracking Product

In Table 1, we summarize various object tracking products that are commercially available on the market along with their form factors and capabilities. A common observation is that most products either requires an external device with extra cost, or requires a build-in GPS like FindMyIPhone, on IOS devices. IU Lost and Found is able to make a leap forward by avoiding these limitations while getting the location of lost items.

	No External Device	No Extra Cost	Find Various items
Protag Elite	No	No	Yes
Find my Phone	Yes	Yes	No
Lost and Found	Yes	Yes	Yes

Table 1: Comparison of IU Lost and Found with recent and/or popular object tracking product available commercially

2.2 Design Consideration

The primary design goal for IU Lost and found is to support fast and accurate image matching. In this vein, the following key design considerations have shaped the design of IU Lost and Found:

- **Fast Image matching**: Computer Vision algorithms are usually not fast due to the long image processing time. This leads to our most crucial challenge which is to design this application to be as quick as possible. Therefore, consider the weaker computing power of mobile devices, our primary approach is to handle all the calculation on web server to save computing time for complex Computer Vision algorithms, and return back the results to mobile devices. During the process, we tested the application with multiple Computer Vision algorithm to find the best combination of algorithms for our project with minimum latency.
- **Minimal Accuracy Loss Optimizations:** The selected algorithms should also have a high accuracy in image matching. Because we want to avoid having the user going to the lost and found office without a correctly matched item. As such, we utilize the RANSAC algorithm and a threshold for good match to increase our general accuracy for image matching.
- **Mobile Interface Design:** In order to have a working project, there should be a simple mobile interface to accept user input, send it to the server, and get the result after the computation finish on the server side.
- **Database holds Found information:** In order to quickly and automatically retrieve all found items, a database will be necessary on our server. For each found item, it should have enough information including a URL to the image path and a tag for object's current location. In order to simplify the logic, we created a JSON file as our database. All the information for one found items will be stored as a JSON object.

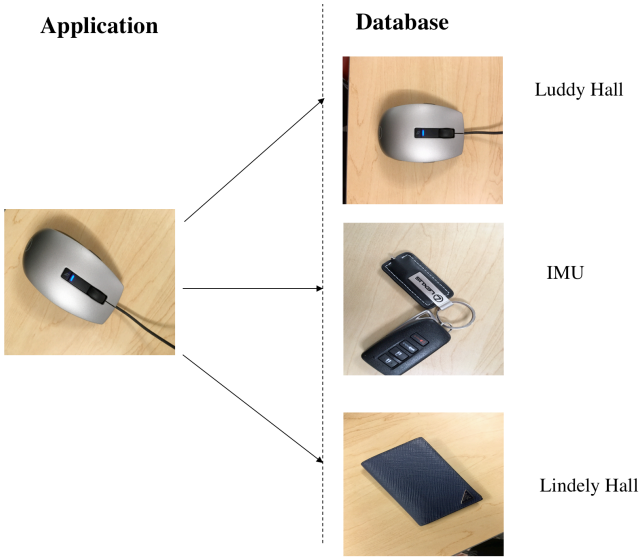

Figure 1: Relational example between the mobile application input and database storage.

3. IU LOST AND FOUND DESIGN

As previously discussed, IU Lost and Found will have a database to maintain all the found items information, including the image URL, and current location. The user will send a picture of their lost object, the server can check if their lost has a match in the database and then prompt the

result back to the user. If the result shows a match is found, the application will prompt the user with its current possible locations. Because their might be similar objects found by different offices. This input and output logic is represented in Figure 1.

In the following sections, we will briefly discuss the high-level idea for our major functionality which is image matching.

3.1 Interest Point Selection

The first step to find a matching image is to select feature points from an image. Several algorithms have been designed in order to accomplish this. The key idea for feature point selection is to find the "interesting" points on each image which identify the object inside of an image [2]. Selecting the feature points in the first place will allow us to have a more accurate result during the feature points matching stage. If we skip this process and start matching without the feature point selected, the result has a large possibility to be biased by many other factors such as lighting of the image or the angle of the object. we will only be caring about the points which have significant changes. These points are usually the edges of the object in the picture. We will not start point matching until we know where is the object located at inside of the picture.

3.2 Feature Point Matching

Once we have detected all the interesting points in images, we can establish some feature matching between two images. Feature point matching algorithms compares the feature points in two images. Base on the Euclidean distance in feature space [3], we can select and rank the potential matches. The points with closet Euclidean distance will be consider as a pair of matching points and stored in a list. There are also multiple algorithms to accomplish that, which we will discuss more in the Implementation section.

3.3 Matching Optimization

Although the combination of feature selection and feature matching already generate a list of matching point and greatly increase the accuracy of image matching. There is still a big chance that unrelated feature points will be connected which will leads to an incorrect result. In our selected feature points, there will be numbers of outliers produced by image noise. Consider these points as interest points will affect the accuracy of feature matching. Therefore, we need to separate the outliers and the inliers before we give a final result for match or not match. We determine the outliers and the inliers by comparing them with a threshold. This threshold is determined by the distance relationship between all matching points. We will only save the feature points that is greater than this threshold.

3.4 Determination of a Match

After we find and optimized the matching features, we need to utilize this information to make a conclusion, that is whether the pair of images is a match or not. We make this decision based on the number of feature points we found. When there are few matching points between the images, the two images are most likely to be different object. On the other hand, if we allow too many matching points, then a lot of the selected points will actually be outliers such as noise. After multiple times of testing, we set the magic number to 10. If there are more than 10 matching points between two images than we conclude that those pair of images are a match. Otherwise, we determine that those two images contain two different objects.

After the previous steps are finished, we will find one or more matching images in the database. Next, we prompt the user with the final result. The result will include a message that if the lost item is found in the database or not. And if a match is found, it can also prompt the user will the current possible locations, so that the user who lost this object can easily retrieve it as prompted.

4. IMPLEMENTATION

We build IU Lost and Found Interface on Android environment. We implemented our matching algorithms in Python with the support of OpenCV3 packet.

4.1 Accommodation to Mobile Devices

The image matching algorithms are relatively slow because they will need to check each pixel multiple times in single image. Because, as we learnt, that the computing power is much better on cloud servers compare to computing it on a mobile device. Doing such computation on a mobile device will produce a large latency while waiting for response. In addition, it will cost large amount of CPU space and consume great amount of energy. In order to avoid image matching algorithms running locally on android virtue devices, the image processing tasks are all conducted on server. We also store all related data including the found images on the server to reduce the storage space in the mobile device.

4.2 Feature Point Detection

Our first choice for feature point detection algorithm is SIFT. Because SIFT is designed to be a point detection algorithm with higher accuracy. We imported the SIFT algorithm from OpenCV3 to detect the feature point from our source image and target image that we selected in our database. During the computation of SIFT we will construct a scale space and apply Laplacian of Gaussian to blur the image in order to find more significant feature points. Than we will get the key feature points by finding the maxima and minima in the Difference of Gaussian images. And finally, we will assign an orientation to the key points and generate SIFT feature points. After SIFT

algorithm selection, we have got the relative suitable feature points from both images.

4.3 Feature Point Matching

Next, we use Bruce Force Matching algorithm also imported from OpenCV3 package to compare the feature point we selected from both image in the previous step. If the feature point matching stage, we can store the matched points and temporally stored them inside of a list. In order to get more concrete matching points. After we get some selections of matching points, in order to get more concrete results, we look through all the pair of points again and only save the points with threshold of 0.7*n.distance. The algorithm is stated as below.

```
good = []
    for m,n in matches:
        if m.distance < 0.7*n.distance:
            good.append(m)
```

4.4 Feature Point Optimization

Finally, we want to double check how many feature points we current have between two images, if the number of feature points is less then number what we need (number of feature points needed is set to ten). If the number of matching points is less than 10, it means we do not have enough feature points to conclude that two images is a match.

If the number of matching point is more than number we need. We can use RANSAC algorithm to select the best 10 feature point we need and finally find the homography between those two images.

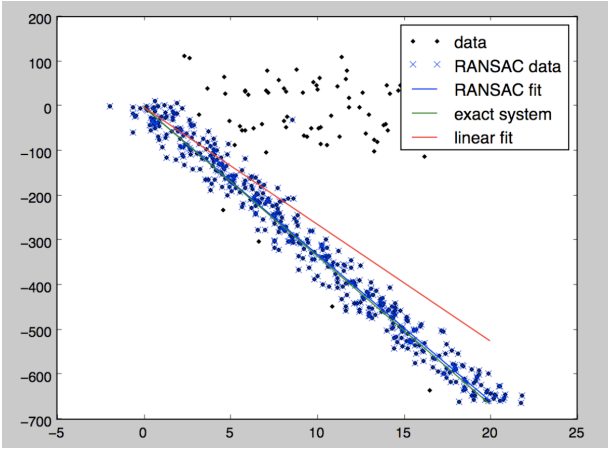

Figure 2: RANSAC Optimization to eliminate outliers of feature points [9]

4.5 Algorithm Selection Optimization

In the previous section, we discussed the basic ideas we used to compute image matching. However, we also checked some additional algorithms that does the same job. We import SURF, Brisk and ORB algorithm to replace for feature point selection. In the meanwhile, we also use Flann [7] matching algorithm instead of BF [8] Match algorithm to find the matching point. Flann is an optimized version of BF Match because it finds the approximate neighbor points and only check those to find a match. The performance will be further discussed in section 5.1.

According to 8 combination result, we compare the time consumption and image matching accuracy to get the most efficient combination we need

5. PERFORMANCE EVALUATION

We have conducted a series of experiments to evaluate and optimize the performance of IU Lost and Found using Android Virtue Device – Nexus 5X API 26. In general, we discover that the time for the program to search through the database and conduct images matching is highly depended on our choice of algorithms. The quickest combination of algorithms can result in a 30 times difference in speed (ORB and Flann compare to SURF [4] and BF). However, the selection of algorithm does not show significant affect to the image matching accuracy. An important factor that affect accuracy is the image resolution. Result shown that after a comparison between two high resolution images and two low resolution images, the previous one will show a significant higher accuracy.

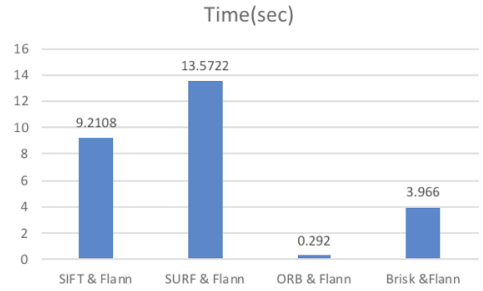

Figure 3: Time take for image matching in a database size of 10 with Flann Based Matching.

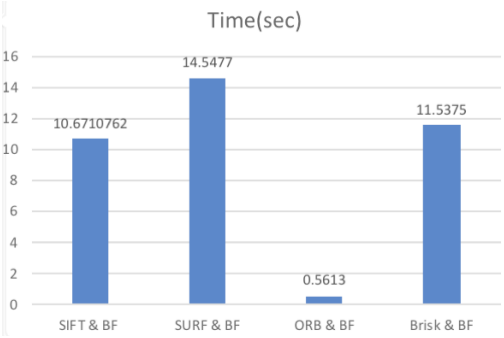

Figure 4: Time take for image matching in a database size of 10 with BF Matching.

5.1 Speed Optimization

We evaluate the time spend from inputting an image, conduct image matching algorithm in a database with size of ten, to outputting a result. The performance is shown in figure 3 and 4.

We implement three different algorithms for interest features selection which are SIFT, SURF [4], ORB [6], and Brisk [5]. During out implementation, SIFT was our first choice because it is one of the most widely used algorithm for interest point selection. SIFT will select the feature point with less outliers. SIFT is a great algorithm to select feature points in complex images. However, in our case, the input image and the images in database mostly have simple structure, etc. keys, mobile phones, and wallet. With this considered, ORB is actually a much quicker algorithm to use as a feature point selection algorithm. It is an efficient alternative to SIFT or SURF

There is also a significant difference in speed produced by feature matching algorithms. We implement our programs with two feature matching algorithms, BF Matcher and Flann Based Matcher. BF Match checks all the feature points relationship with brute force. Because every point is check, BF Match will be able to give the best matches. On the other hand, Flann is an algorithm that approximate feature point neighbors and then compute for the best match. Flann also accepts a threshold for us to determine how accurate it should be. As we discovered earlier, since our images are relatively simple, we have a high tolerance for feature point matching. Therefore, we also pick the faster algorithm to implement the application.

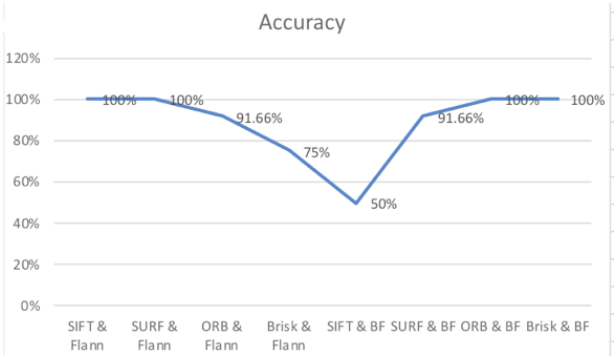

Figure 5: Search accuracy for all combination of matching algorithms.

	Time	accuracy
High Resolution	0.292	100%
Low Resolution	0.97	91.66%

Table 2: Time and accuracy performance with high-resolution input and low-resolution input.

Figure 6: Imaging matching algorithm tested between difference images with same object to test accuracy of the algorithm.

5.2 MATCHING ACCURACY

We select multiple images of the same object. These pictures of the same objects are taken in different background, angles, and lighting environment. We than conduct the image matching with between these images of same object. An example is shown in Figure 6. Therefore, for each testing set with size N, we will test the accuracy of the application:

$$(N-1)+(N-2)+\ldots\ldots+2+1 \text{ times}$$

We count the times that the application returns the correct result as a success match. Then, we will be able to use these data to then calculate the accuracy of our program by divide success match counts over total counts.

We were expecting different algorithms to have different accuracy. However, it is unexpected that the combination of algorithms actually produces extremely similar results. With a high-resolution input, all of the combination of algorithm is able to produce a search accuracy greater than 90%. This is shown in table 2. We have learnt that because our input images and our images in the database are relatively simple, the accuracy therefore won't be able to affect by the feature point matching algorithms. So, while we are doing our final choosing for algorithms, we put the speed as our top priority.

However, there is one thing interested us during the evaluation. The matching accuracy can be affected by the resolution of the images. Table 3 shows that the accuracy and time spend of image matching with different resolution. We discover that the accuracy of the application dropped about 10 percent. And the time spent for the computation is also greatly delayed. We believe this is because in low resolution, it is relatively harder for the interest matching algorithm to find enough feature points. Without enough

feature point, it does not satisfy the threshold for enough matching points.

6. LIMITATION AND FUTURE WORK

There are several limitations for our project includes:

- We only tested out algorithms on virtue device with higher version of Android OS. It is possible that result will differ when testing on an actual mobile device especially with lower version of Android.
- We only implemented the android application by assuming the user is the person who lost an object. In order to have a fully functioning application, we can implement an interface for Lost and Found office managers. Once, they receive a found item, they can upload it to the database.
- Another limitation is the lower accuracy when applying the algorithm to low-resolution images. These images will less likely to find a matching image because in low-resolution images, a lot less feature points will be selected. Therefore, it may not reach the threshold we set. One of our possible future work is to optimize our matching determination function. If the program can set a smaller threshold for low-resolution images, than we can increase our accuracy for low-resolution images' matching.

7. CONCLUSION

As a result, we successfully design and implement IU Lost and Found by accomplishing our goal. Which is to have application that simplify the experience of retrieving lost item at IU. Our solution does not require the user to purchase additional devices that attached to their personal belongings. We also optimize the time spend for image matching by conduct out calculation on web server instead of on the mobile devices. IU Lost and Found is nearly able to guarantee a correct search result with a high-resolution input. We believe an application like IU Lost and Found can greatly benefits all the students and faculties at IU campuses by simplify their experience when they lost an important object on campus.

8. REFERENCES

[1] Haase, Marc and Handy, Matthias. BlueTrack – Imperceptible Tracking of Bluetooth Devices. University of Rostock, 2004, in Ubicomp Poster Proceedings.

[2] Laptev, Ivan and Lindeberg, Tony. Interest point detection and scale selection in space-time, Computational Vision and Active Perception Laboratory.

[3] Richard Szeliski, 2004, Computer Vision: Algorithms and Applications

[4] Bay H., Tuytelaars T., Van Gool L. (2006) SURF: Speeded Up Robust Features. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg

[5] Stefan Leutenegger, Stefan Leutenegger, Stefan Leutenegger " BRISK: Binary Robust invariant scalable keypoints"in Computer Vision (ICCV), 2011 IEEE International Conference in Barcelona, Spain, IEEE, 2011

[6] Ethan Rublee,Vincent Rabaud,Vincent Rabaud "ORB: An efficient alternative to SIFT or SURF" in Computer Vision (ICCV), 2011 IEEE International Conference in Barcelona, Spain : IEEE 2011

[7] Marius Muja,David G. Lowe "Fast Matching of Binary Features" in Computer and Robot Vision (CRV), 2012 Ninth Conference in Toronto, ON, Canada: IEEE 2012

[8] Ondrej Miksik,Krystian Mikolajczyk,"Evaluation of local detectors and descriptors for fast feature matching" in Pattern Recognition (ICPR), 2012 21st International Conference in Tsukuba, Japan: IEEE 2012

[9] http://yongyuan.name/blog/fitting-line-with-ransac.html

Personal Contribution

Shengyu Chen:

1. write the source python code for image feature point detection(SIFT, SURF, Brisk, and ORB), feature point matching (Flann match and BF match), and further image matching and find the homography.

2. write and modify JSON file for database used and http request used

3. do the data evaluation for time consumption and accuracy evaluation and write a few parts of final report

Xiner Zhang

1. design the IU lost and found functionality

2. write the android client code and create the main android user interface.

3. write the main part of final report