2011 IEEE 13th International Symposium on High-Assurance Systems Engineering

# Validation of Object Recognition Framework on Android Mobile Platform

Vivek Tyagi, A .S. Pandya, Ankur Agarwal, Bassem Alhalabi
Department of Computer Science
Florida Atlantic University
Boca Raton,FL 33431

*Abstract*— **In recent years there has been great interest in implementing object recognition frame work on mobile phones. This has stemmed from the fact the advances in object recognition algorithm and mobile phone capabilities have built a congenial ecosystem. In this paper, we propose a framework to overcome design challenges and provide an evaluation methodology to assess the system performance. We use the emerging Android mobile platform to implement and test the framework. We performed a case study using the proposal and reported the test result.**

## I. INTRODUCTION

In recent years, there have been great advances in the field of computer vision and ecosystem for its implementation on mobile devices. High resolution cameras are now available at low price and are now part of most of the mobile devices. Also, mobile phones now are equipped with the computing power equivalent of desktops from five years ago. There have been great strides in the data connectivity of these devices. This has resulted in a client server architecture application ecosystem for mobile phones.

With dramatic improvement in these areas, efforts are now underway to put object recognition technology on mobile phones. As with any new technologies there are lots of challenges in implementing an optimum solution. We believe that proper evaluation framework should be developed along with the application framework to better understand the challenges in implementation. In this paper we propose a framework to implement and test object recognition technology on the mobile phones.

## II. BACKGROUND

Implementing an object recognition framework on mobile devices has the several challenges. Cameras on the mobile devices do not provide distortion-free images; these images have geometric and photometric distortions. For mainstream applications, the system must meet certain time constraints. Although the processing power on Mobile devices has increased, object recognition algorithms take a significant amount of computing resources. Also, the application should send as little information as possible over the network to overcome bandwidth constraints. For implementing an efficient application in this domain, we require an end-to-end system architecture and test framework.

### A. Related work

The most ambitious application of object recognition on mobile platform has been with Google goggles [3] by Google. The first step is to find a robust and computationally effective feature descriptor for the image. SIFT[1] has been proven to be a robust feature for object detection. However, SIFT is very demanding in terms of computational resources. This is where SURF [2] has proven to provide a good balance of robustness and computational efficiency. There have been various implementations of the SURF[4] algorithm on Android. For this paper, we have taken a java-based SURF implementation JopenSurf[5] and try to port it to the Android platform. Implementation of test framework to assess the effect of various conditions is a major contribution of this paper.Most of the related work on the field presents the results as a measure of the accuracy of the system. There was no methodology of testing the accuracy under different test conditions. In this paper we provide this framework.

## III. PROPOSED SYSTEM ARCHITECTURE

The system is implemented using client server application architecture (Figure 1). The image processing and feature extraction steps are carried out on the client end. In this paper, the client end application is a mobile application. The client extracts the SURF vectors from the image and sends it to the server. The matching of features and information retrieval is done on the server side. The server communicates the result back to the client in Extensible Markup Language (XML) format. This XML data is parsed by the client application and is presented to the user on the mobile screen. On the mobile end, we implement the client application on the Android Mobile Platform. Sever components are web applications that run on a J2EE platform.

### A. Mobile Component

The application component on the mobile end is implemented on an Android platform. The mobile client is responsible for sending the feature vector to the server and displaying the results of the classification. The first part of the functionality involves image acquisition and feature extraction. For displaying results, the client interface application parses the XML feed received from the server to display the results.

### B. Speeded Up Robust Featuress

The speeded Up Robust Feature is an image detector and descriptor proposed by Herbert Bay et al. in 2006 [2].The original SURF algorithm is composed of three stages. In the first stage, interest points are found in the image by a Fast-Hessian Detector. In the second stage, Haar wavelet

1530-2059/11 $26.00 © 2011 IEEE
DOI 10.1109/HASE.2011.62

responses for both x and y directions are calculated around the interest point and the most dominant direction is chosen to achieve rotation invariance. In the last stage, Haar wavelet functions are used to calculate a descriptor of interest points surrounding the area.

### C. Server Application

The server end of the application is responsible for implementing the classification algorithm. Before a classification algorithm can be run, we need to create a reference database. This database stores the Meta data for the object of interest and the feature vector. First, we gather the images from the relevant categories to populate our reference database. For example, if we are including books and CD covers as categories, we gather images for this data set. The sources are chosen so as to also include Meta data. In first step we populate the image data base. In the second step, we convert the image to feature vector by applying SURF algorithm. This feature vector, along with the image information, is then stored in the database. Here the feature vector received from the client end is matched with the stored image set. The matching algorithm uses a "distance" criteria for deciding upon the best match for the given query vector. The distance criteria depend upon

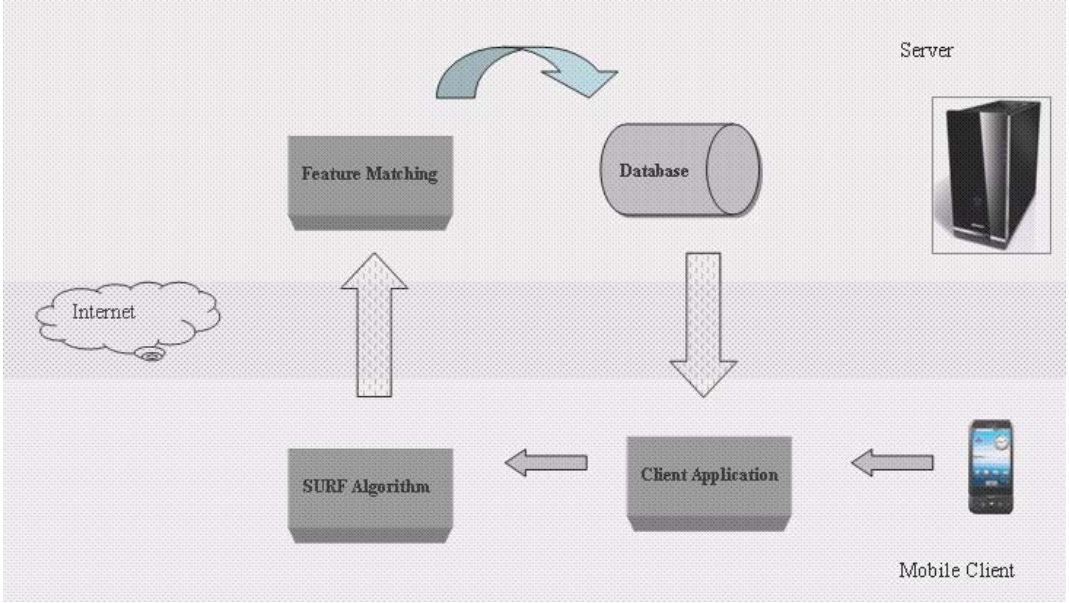


**Figure 1 System Architecture**

the choice of classification algorithm used in the process. Here we use the Euclidian distance as measure for the similarity of the feature.

### D. Matching Algorithm

There are many ways in which a matching algorithm can be implemented. Here we use the Euclidian distance as measure for the similarity of the feature set. Each image has a set of SURF feature vectors associated with it, which are 64 dimensional vectors. The image retrieval system works by first calculating the match number, which is the measure of "similarity" between two images. It is calculated as follows.

Let $I$ and $J$ be two images. After applying the SURF algorithm, these images are represented as follows:

$I = \{i_1, i_2, i_3, ....., i_n\}$

$J = \{j_1, j_2, j_3, ....., j_m\}$

Where $i, j \in R^d$ here d=64.

To compare the images we have to match each of the feature vectors in image I with that in image J, and then the reverse.

For this we use the Nearest Neighbor Search to find the match for vectors in $I$.

Let $IJ_{match}$ be the number of vectors in $I$ that found a match.

Repeating the same procedure, we find

$JI_{match}$

The total number of matches thus found is termed as Match Number:

Match Number $M_{ij} = IJ_{match} + JI_{match}$.

Match Ratio =Match Number/Total Number of Pairs

$$MR_{ij} = \frac{\text{Match Number}}{\text{Number of vectors in I +Number of vectors in J}} = \frac{M_{ij}}{|I|+|J|}$$

For simplicity $MR_\Theta$ is expressed as a percentage.

The Match Ratio is the measure of degree of similarity for the images $I$ and $J$ .

For the database search:

Let $D \in \{d1, d2, d3, ...dn\}$ and $q$ be the query image. We first calculate the Match Ratio for the query image for each image in the database.

$$\{MR_{qd1}, MR_{qd2}, MR_{qd3}, ..., MR_{qdn}\}$$

We then select a threshold for selection $MR_{\Theta}$.

The Matched images are those that have $MR \geq MR_{\Theta}$.

## IV. PERFORMANCE AND ACCURACY VARIABLES

The object recognition system on mobile devices suffers from Latency and Field Conditions issues. We define latency for the system as the time elapsed from the capture of the image to the display of the results. The latency can be subdivided further into client side latency and server side latency. For testing the latency, we used the Wi-Fi data channels for data transmissions. The software was tagged with system time function call to capture the time stamp. These time stamps were retrieved via log files to further calculate the time difference. Major factors in field conditions are Rotation, Viewing angle, Light Conditions, Scale,

## V. TEST METHODOLOGY

For testing system performance, we conduct a series of tests on the framework to identify performance and effectiveness. The goal is to present quantitative assessment of field conditions on the discriminative power of framework. For this, we employ a presentation methodology called response curve. A response curve is a plot of Match Ratio values for all the images in the database for a given image. The response curve shows the discriminative state of the query image with respect to the rest of the database. An image has a higher probability of being recognized correctly if it has a single clearly defined peak in the response curve. This means a proper threshold can be used to get match of for the image in a database. A sample database of 20 images is taken as query space.We then take a image from this dataset and simulate a particular filed condition for example rotation. The response curve of the test images is plotted.

## VI. CASE STUDY RESULTS

In this section, we present the different test conditions and the respective results. All the case study simulation was done using GIMP[6].

### A. *Rotation*

The Rotation is a type of affine transformation. Rotation of input image with respect to the view stored in the database can occur when the user clicks a picture of the target object in an angle different from what is stored in the database. We observe that system is not very robust with the rotation as the response curve doesn't have a clear peak.


**Figure 2 Example Rotation**

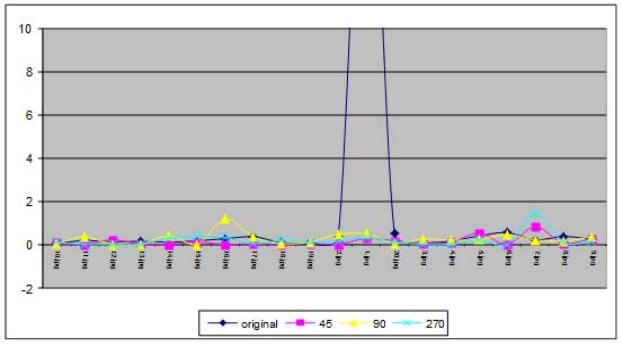


**Figure 3 Rotation Response Curve**

### B. *Perspective View*

Perspective view refers to the image capture situation where the camera plane is not parallel to the object plane. We observe that in some conditions the system has clear peak and is robust in finding the correct image.But in some conditions we have two peaks.As there is only one correct images result,this is a false positive case.

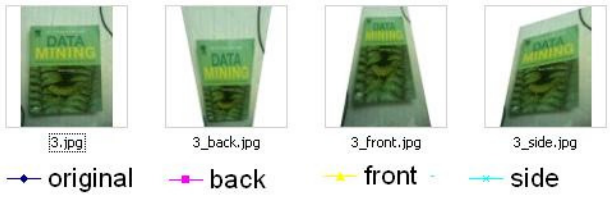


**Figure 4 Example Perspective View**

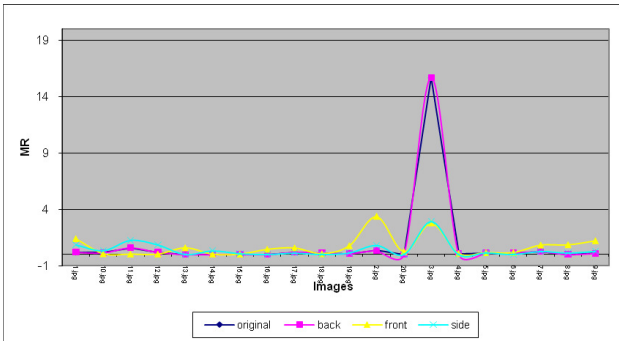


**Figure 5 Perspective Response Curve**

### C. *Light Conditions*

The images stored in the database are taken in a well-illuminated environment. However, in a real life scenario, the light condition can vary and most of the time it is not as

good as the image in the database. For testing the effect of light conditions, we create three different image states with decreasing light intensity.We observer that the system is very robust to the lightning conditions. There are well defined peaks ,which means the discriminatory effect of algorithm is retained.


**Figure 6 Example Illumination**

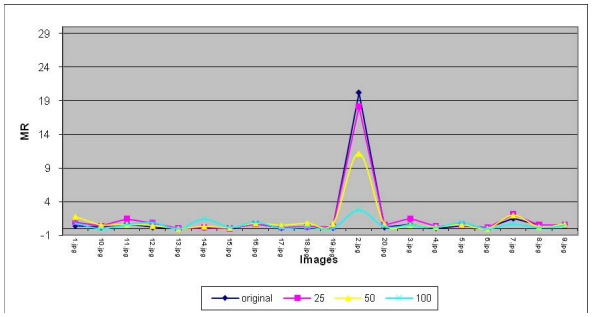


**Figure 7 Illumination Response Curve**

*D. Scale*

Scale refers to the change in dimension of the images while keeping the other geometric properties the same. The scale condition occurs when the user captures the image from a distance different than that present in the database. We then analyze the response curve to study the effect of scaling on discriminative state. We see that there are well defined peaks in different scale condition; hence system is robust in terms of scaling.

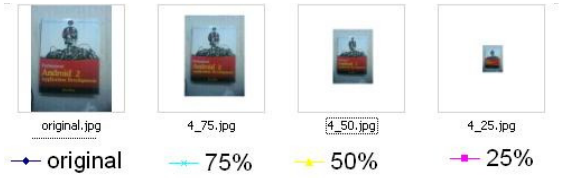


**Figure 8 Example Scale**

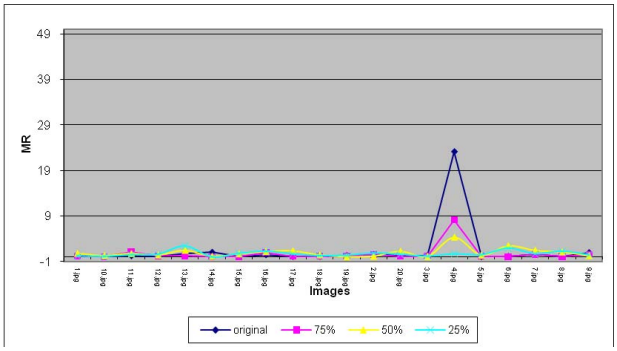


**Figure 9 Scale Response Curve**

## VII. CONCLUSIONS

In this paper, we proposed scalable system architecture for providing object recognition capabilities on Android based mobile devices. The object recognition algorithm used is based on SURF. We ported a Java based implantation of SURF to the Android platform. The architecture is a client server model, which divides the effort between the mobile device and remote server for optimizing results. The client end extracts SURF features and the server end matches the query feature to the image data stored in the backend database. We proposed and implemented a simple matching algorithm based on nearest neighbor search. We provided a case study of implementing this architecture using an Android platform.The case study consisted of a book cover matching application along with a test framework. We further identified the factors that could impact the ideal behavior of the system. Factors such as image rotation, perspective views, scale, and illuminations can cause negative effects on the accuracy of the classifier build in ideal conditions. To study the effects of these factors, we proposed a simple methodology called response curve. With response curve we can visualize the effect of the external factor on the discriminative power of classifier. We used the response curve methodology to analyze the effects of external factors in our case study

## REFERENCES


[1] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision, vol. 60, 2004, pp. 91-110.

[2] H. Bay, A. Ess, T. Tuytelaars, and L.V. Gool, "Surf: Speeded up robust features," Computer Vision and Image Understanding (CVIU), vol. 110, 2008, pp. 346-359.

[3] "Google Goggles." http://www.google.com/mobile/goggles/#text [Accessed: June. 12, 2010].

[4] S. Olsson and P. Akesson, "Distributed mobile computer vision and applications on the android platform," M.S. thesis, Lund University, Lund, Sweden, 2009.

[5] Stromberg, "JOpenSURF The SURF descriptor." Available: http://www.stromberglabs.com/jopensurf/ [Accessed: June. 12, 2010].

[6] "GIMP-The GNU Image Manipulation Program." Available: http://www.gimp.org/ [Accessed: June. 12, 2010].