iPoll

Audience Polling System on iPhone

Amin Heidari, Bita Sadeghlo

April 12, 2011

1 Introduction

1.1 Goal of the project

Audience polling is a very common event that takes place in every effective meetings and learning activity. Depending on the number of people, there are so many methods for performing the process. The simplest one would be counting the number of hands of the audience who are voting. Although this seems simple, it may become very tedious in some cases and that's why there are so many efforts done to automize this process. Some examples are interactive voting pads, audience voting keypads, and clickers. These electronic devices are useful for quickly recording audience member answers, however, they may present some difficulties in both their deployment and use. Some of these difficulties are as follows:

-The per-unit purchase price of devices.

-The maintenance and repair of devices when owned by a central unit or organization.

-The configuration, troubleshooting and support of the related presentation software (which may or may not work well with devices).

-The reliability and performance of the devices under non-optimal conditions of the room in which the devices are used.

Regarding the ability of the iPhone in taking and processing images, we came up with the idea to do something similar on iPhone. This way, there will be no need for installation of complicated wired or wireless systems and the whole audience polling will be handled by a single iPhone.

1.2 How iPoll Works

Using iPoll is very straightforward. All that user needs to do is to take an image using iPhone's camera, or select one from the photo library. After this, the whole audience polling process, including detecting people, finding raised hands and reporting the percentage of the votes will be handled by the application. A screenshot of iPoll's results is shown in Fig. 1.

The procedure of detecting objects (faces and hands here) in the image contains several image processing steps, many of which are done using openCV functions. These steps will be explained in details in the following sections.

2 Overall Design

2.1 Hand detection

Imagine we have the image in Fig. 2 and we want to find all hands at every scale in it. This is done by scanning the whole image at multiple scales and finding the rectangles that contain hands. At every level of scanning, the cropped rectangles are given to a classifier which determines whether they are hands or not. This is shown in Fig. 3.

The classification system shown in Fig. 3 is based on Viola-Jones object detector which is the first object detection framework to provide competitive object detection rates in real-time proposed in 2001 by Paul Viola and Michael Jone. In order to build such a classifier, we need to train it with a large number of hand and non-hand images so that it learns how a hand looks like.

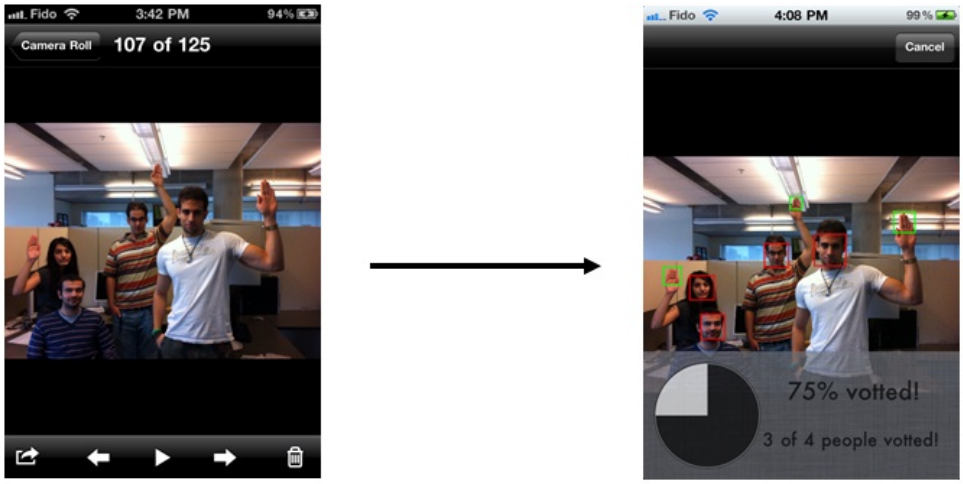

Figure 1: How iPoll works

Figure 2: Sample image used for hand detection

2.2 Training the classifier for hand detection

2.2.1 Collecting Hand images (positive training samples)

As mentioned before, we needed to gather a lot of high quality hand images to be used as positive samples in the training of the classifier. "High quality" means that all unnecessary variance has been removed from the data and they are as similar as possible. Otherwise, the classifier will get confused and will not be able to perform a satisfactory detection.

At the beginning of the project, we thought it would be easy to find some clean hand data set on the internet. Unfortunately, it turned out that despite the huge number of faces, eyes, lips, noses and other image collections available on the web, there are no hand data sets to be used in our work. Therefore, we collected and prepared the hand data set ourselves which took us about two weeks to be done.

First, we took about 350 photos of people while they were holding both their hands up. Some examples of these images are shown in Fig. 4.

Next, the regions containing the hands were cropped and then registered with a reference image through a rigid-body spatial transform. The spatial rigid-body transformation was optimized by aligning four control points in the cropped image and reference image as shown in Fig. 5. This way, all hand images used for the training of the classifier became as similar as possible. This way, around 700 images of both left and right hands were collected. Some examples of these registered training images are shown in Fig. 6.

But we still needed more images to train the classifier. So we thought of generating more images from our current data set. To do so, for each of the current hand images, we generated 12 new images by adjusting the colours of the image, adding very little noise and filtering with a gaussian blurring filter. Then we added these new images to the set of positive samples. This way, the number of positive (hand) samples raised to more than $8,000$. This is also good from the training point of view since it's good for the classifier to know that hands might appear noisy, blurred or with varying brightness.

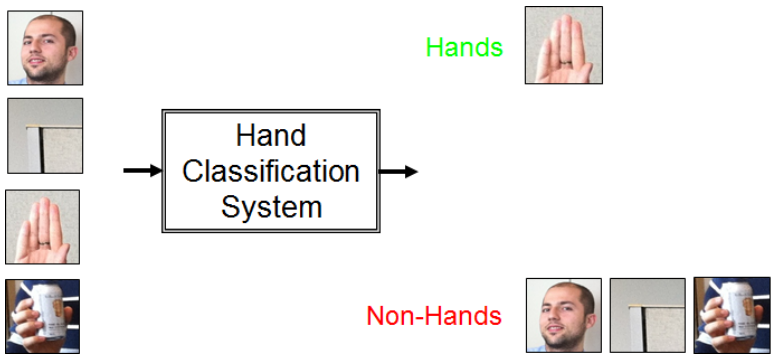

Figure 3: Hand classification system

(a) (b) (c) (d)

Figure 4: Examples of the captured images

2.2.2 Collecting non-Hand images (negative training samples)

According to section 2.1, we also needed to collect non-hand images (negative samples) for training so that the classifier can learn what does not look like hands. Any image that does not contain hands can be turned into a negative sample, but it is best to take the negative images from the same type of data we will test on.

We first collected lots of classroom images from the internet as negative samples. Then, we generated new images by cropping them at random sizes and positions, As shown in Figures 8-9. This way, the number of negative samples raised to more than $30,000$ images.

2.2.3 Putting it all together

After collecting and preparing the positive and negative samples, the classifier was trained with the appropriate training parameters using openCV. This was done multiple times in order to find the best configuration of training parameters. Running each of the trainings took more than 36 hours because of the large number of training samples used.

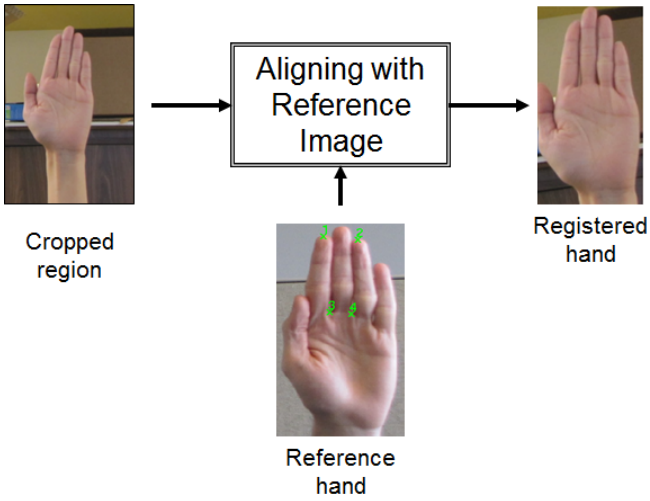

Figure 5: Registering the captured hand images with the reference hand

Therefore, the classifier is ready and it could be used for hand detection on the iPhone.

2.3 Hand detection on the iPhone

Suppose we want to poll the people in the image of Fig. 10 on the iPhone. iPoll performs audience polling in three steps, each of which are described below.

2.3.1 Pre-processing of the image

There is a pre-trained face detector available in openCV libraries. So before starting the processing of image, a fast search is done on the down sampled version of the image in order to find the faces

Figure 6: Some examples of registered hand images for training of hand detector

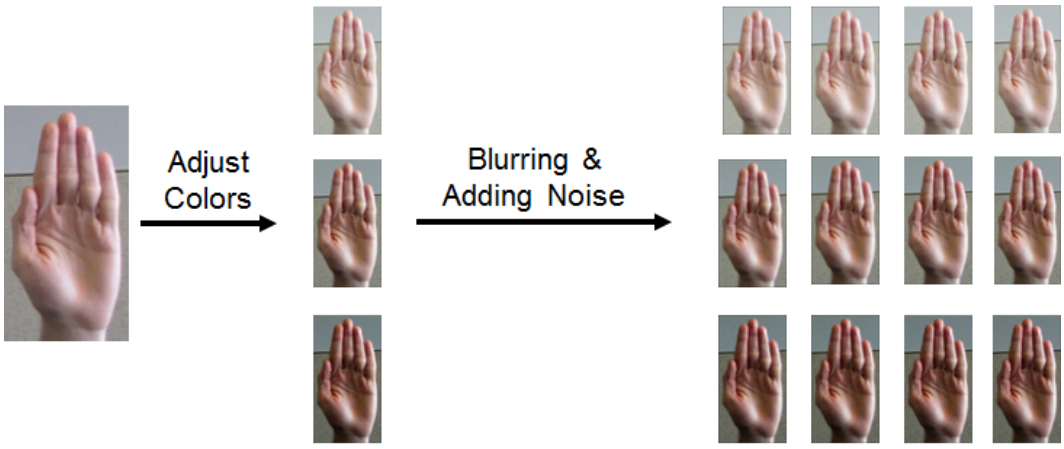

Figure 7: Generating new training images from a hand image

of the people. Next, a region of interest is defined in the image based on the locations of faces as shown in Fig. 11. This region is then passed to the processing step for finding hands. This can save the processing time in cases where we are dealing with a large image with people in a small region of it.

2.3.2 Processing of the image

Using the region of interest defined in the pre-processing step, iPoll then finds the locations of hands in that region using the hand classifier we trained.

2.3.3 Post-processing of the image

After finding hands using hand classifier, iPoll performs some post-processing steps in order to keep only the acceptable hands. These post-processing steps are as follows.

1. The detected hands that are too small or too large regarding the size of their surrounding faces are considered as false positives and rejected.

2. If two or more hands overlap more than a threshold, then only the largest one is retained and the rest are rejected.

The procedure above is shown in Fig. 12 along with final results.

2.4 Overall block diagram

According to the steps mentioned above, the overall block diagram of iPoll is as shown in Fig. 13.

Figure 8: Example of a non-hand (negative) image

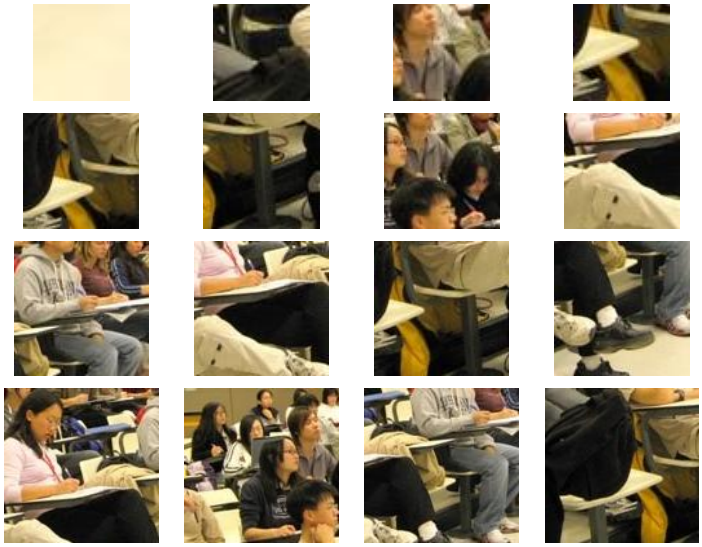

Figure 9: Negative samples generated by randomly cropping a high-resoloution negative image

3 Functionality

3.1 Screen Shots

Figures 14-15 show screen shots of iPoll. As mentioned in the introduction, the user provides the image to the application and after a short time the results appear on the screen.

Figure 10: Example image for audience poling

3.2 Challenges

There are several challenges for iPoll. Some of them are because of the nature of the problem. All computer vision methods have intrinsic properties which makes them to fail in specific situations. The performance of iPoll will therefore be dependent on the quality of the images it is given.

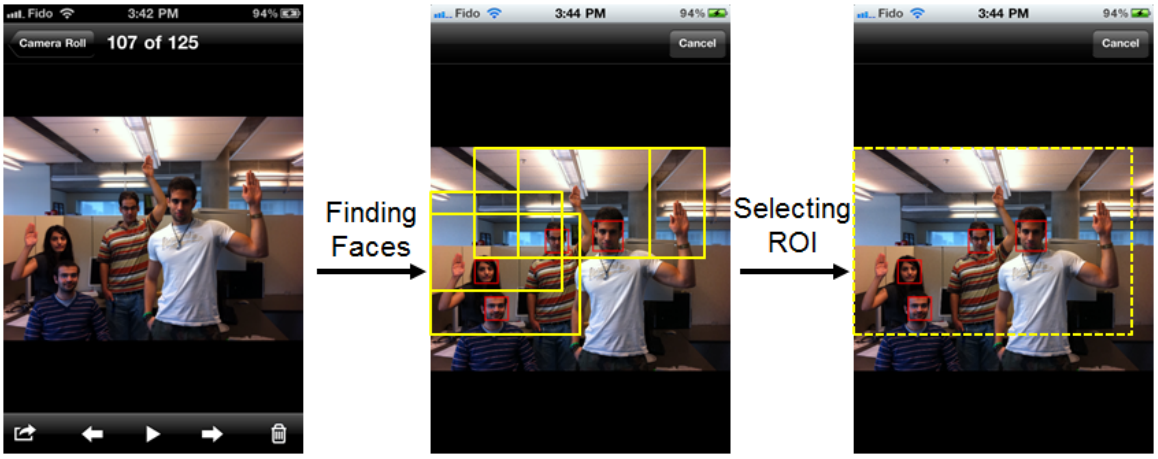

Figure 11: Finding the region of interest for detecting hands

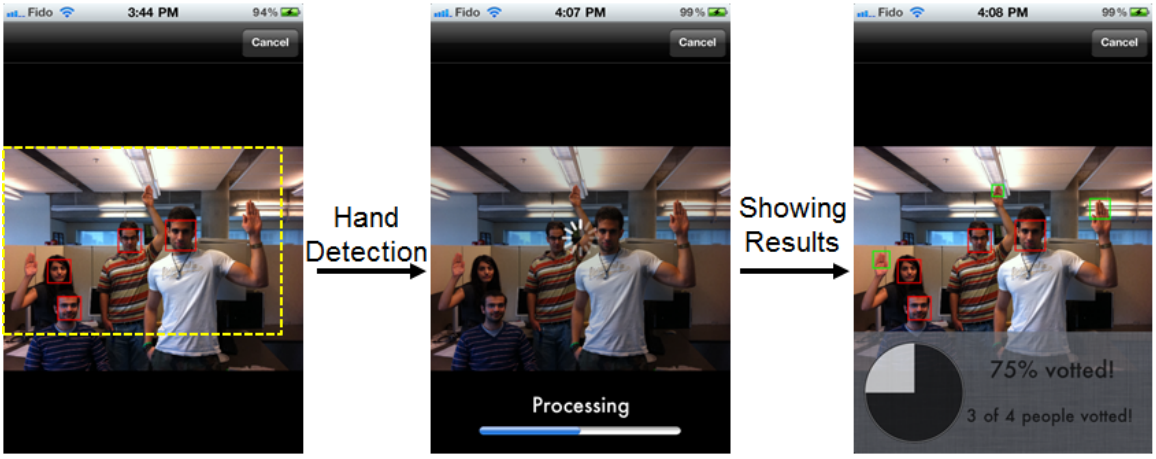

Figure 12: Final results of iPoll

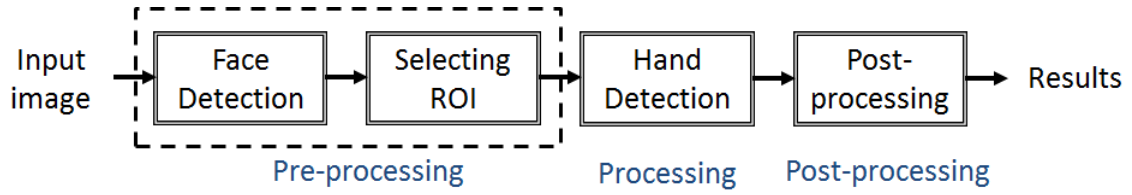

Figure 13: Overall block diagram

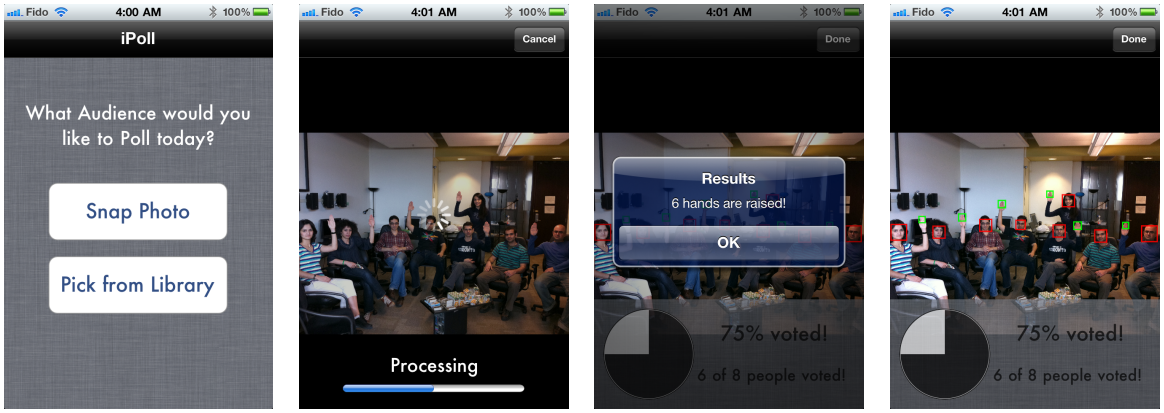

Figure 14: Screen shots of iPoll

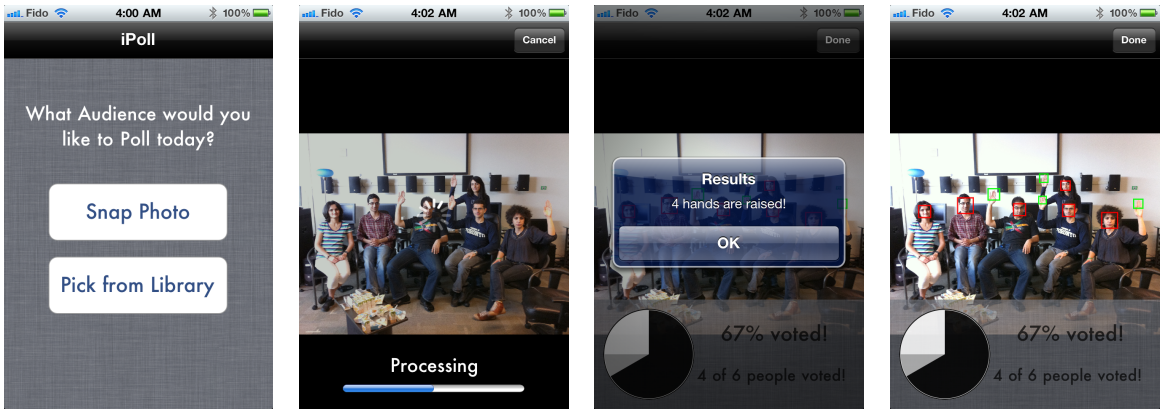

Figure 15: Screen shots of iPoll

For example, Fig. 16 shows a screen shot where iPoll has failed to detect one of the hands. This failure is due to bad illumination conditions at that point which has caused the hand to become significantly different from hand images in the training set. iPoll has also failed to detect one of the faces in this image, which is a result of the bad orientation of the head. Several other failure mode, including rotated hands, hands with different postures, etc could be considered and tested for iPoll. Of course, these kinds of failures could be improved by collecting more images and training a better hand detector.

Another challenge of the iPoll, is the limited computational resources of the iPhone (single core 1GHz processor). In order to have accurate results in an acceptable amount of time, every part of the computations should be optimized. Otherwise, the user will have to wait a long time to get the results.

4 What we learned

4.1 Programming and Scientific Aspects

- Objective C / iPhone programming

  This project provided us the opportunity to improve our programming skills in objective C which is the programming language for iPhone.

- Computer Vision

  Before starting, we were both familiar with computer vision and object detection. This project gave us the opportunity to practice them in more details for real-world applications.

- OpenCV

  OpenCV is one of the best libraries of programming functions for computer vision. During this project, we had the chance of learning how to work with openCV and how to use it on the iPhone.

- General image processing

  As described in Section 2, there are many image processing steps in the preparation of training data. We implemented all of these steps, including registration of images, generating new negative images, generating new positive images, adjusting colours, etc in MATLAB which was a good experience of general digital processing using MATLAB.

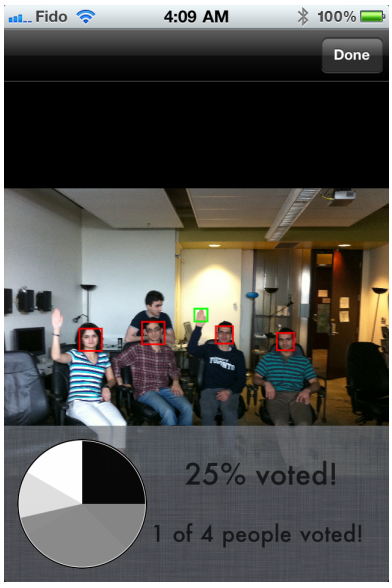

Figure 16: An example where iPoll fails

4.2 Practical Aspects

- Collecting Data-Set

  Since there are no hand data sets for our work available on the web, we had to capture the hand images ourselves. During this, we learned how to communicate with people in order to convince them to cooperate and let us take their photos.

5 Contribution by group members

Regarding that we are both working with the same supervisor, our research interests and skills are so close. So we contributed similarly and did every single part of this project together.

6 Future work

iPoll has the potential to be extended in two ways.

6.1 Research

From the research point of view, if we put more time and improve the quality of training samples, then we will get a better and more reliable performance for iPoll.

iPoll can also be extended in such a way to detect different hand postures so it will be able to poll an audience when they are voting for multiple choices.

6.2 Application

From the application point of view, we can add more fun stuff to the user interface before submitting iPoll to the App. Store in order to have more customers.