Android Augmented Reality Application - Visit Card Real-time Scanne

Anton Prokopov

MSc student of Computer Science Curriculum University of Tartu
Supervised by Assoc. Prof. Dr. Satish Narayana Srirama

Introduction

This poster presents the augmented reality (AR) client-server system developed during the Mobile and Cloud Computing Seminar. General and practical research of topic of AR was conducted and the latest results are described. Aim of this system is to scan visit cards with a camera of an Android device, recognize the text, detect the name of the card owner and search for his picture using Google search engine. The implementation was done firstly on an Android device using OpenCV and Tesseract libraries, but then actual text processing was moved from the device side to the Django server. Later the Django application was moved from local server to an Amazon EC2 instance, which allowed to access the server from every network.

Challenges

In order to accomplish the goals, the following tasks had to be resolved:

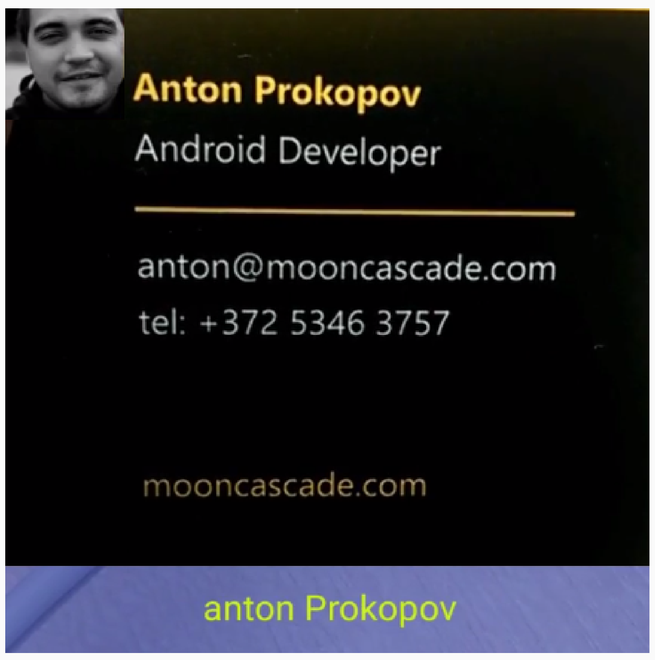
1. Image, obtained from camera of an Android device that contains the typed text, has to be translated into actual text. Popular technique for this task is to use optical character recognition (OCR) to recognize text from an image. It can be applied by using open-source library called Tesseract. Even though OCR is already implemented, tasks of optimizing the algorithm for Android and increasing the accuracy are still challenging.
2. When OCR is performed and the text is stored in the memory, it is really hard to understand, what text corresponds to the name and forename of a visit card owner. Extracting only desired text from the whole scanned text is not a trivial task.
3. In the latest version of the system, the whole processing is done on the server side, which means data obtained from camera has to be transferred somehow to the server and the server should respond with the desired result, which is a picture of a card owner. Goal of the server processing is to speed up the algorithm, so it is really important, how to compress and transmit data the most optimal way.
4. Real-time scanning means not only fast text recognition and optimal data transfer between the server and Android application, but also smooth experience for the user. If software is lagging, the user should be notified about the reasons of the lags: poor Internet connection, camera movements are too fast, so there is no focus, too much text in front of the camera and so on.

Results

On-device recognition, was too slow, because computationally heavy OCR algorithm was running on a smartphone's CPUs. As tests have shown, the average processing speed of each frame, while running the recognition on device, was about 6 seconds. Some frames were processed within 3 seconds, some of them took up to 10 seconds.

Introducing server to run the computationally heavy operations could possibly solve the problem, as server has much more powerful CPUs. Was server a good idea in order to optimize the process of text recognition? Yes. Using the client-server architecture it was possible to reduce the average time to 3 seconds per frame. Server is also more stable, as the time varies here between 2.5 and 3.5 seconds per frame.

After transferring the Django application to Amazon EC2 instance, the latency of data transfer (including processing) has increased to 4-5 seconds per frame. Using Europe server instead of US should decrease the time required for data transferring. Using more powerful instance should decrease the time for processing.



How it works?

First, camera frame is received. Then the processing of the frame can be started. By processing the frame following steps are meant: sending data to server, extracting the text, searching for an image, returning the response.

To optimize the algorithm, the text processing is not triggered on every frame, but on every 10-th frame, which means the processing is launched 3 times in a second - this is still very fast, so the real-time processing could be achieved. Saving the time is critical, so every time the processing is launched, a new thread for it is created. There is also a buffer of maximum 4 threads being launched at a time to ensure the application does not crash with creating infinite amount of threads. If the buffer is full, no new thread is created, therefore no processing for new frames is launched. After a thread has finished and the result is received back from server, the thread is removed from the buffer.

Sending data to server is realised by encoding an image as a base64 string and transmitting this string inside a JSON. After JSON is received by the server the image from it gets decoded. Obtained image is then formatted appropriately.

After image is in correct format, OCR can be applied for text recognition. Tesseract library for python is used in order to get the scanned text on the server side. When the whole text is received, the name and forename should be extracted. The implementation of name and surname extraction from a string is pretty naive and requires a few assumptions:

1. on every visit card there is an email address
2. every email address either contains name before the '@' symbol or dot-separated name and surname
3. on every visit card a surname always follows the name and they are separated using whitespace symbol

When aforementioned requirements are met, the name and surname can be successfully extracted. Now it is possible to use extracted name and surname to search for a person's photo using Google Custom Search. The first image from the response is considered to be the photo of a visit card owner. This image with the extracted name and surname is then given back to the Android application encoded as a base64 string.

Conclusion

The results of the Visit Card Scanner application are satisfactory. Three seconds per frame are still a too much, but as camera is running at the same time in another thread on Android device and the result gets updated on the fly, the latency does not affect the user so much - he always can see that the app is doing something. It is not a real-time processing, but it seems like one. Further optimisation is desirable nonetheless.

Accuracy of the text recognition is sufficient, but in case of camera movement, insufficient focus or poor light conditions, the accuracy drops. This can be further improved by adding preprocessing, while doing text recognition. Before performing OCR, it is possible to prepare the image for better text extraction. Adding preprocessing to on-device application would increase the processing time dramatically, however using powerful server for processing means the processing time is basically the same time that is required for data transmission.

Info

More information at **http://kodu.ut.ee/ prokopov/ardemo** Contact: **prokopov@hotmail.com**