18-551 PROJECT PROPOSAL, SPRING 2012:

TEXT RECOGNITION AND TRANSLATION ON THE GO

GROUP #1

ALEX BARAN	ALEX.V.BARAN@GMAIL.COM
JAMES CHUN	JTCHUN@ANDREW.CMU.EDU
TOM KEAGLE	TKEAGLE@ANDREW.CMU.EDU

INTRODUCTION

With the rise in use of smartphones and tablets, the informational world has opened up tremendously. This allows for a wide variety of data exchanges, one of the largest consumer interests being travel. However, not every tourist knows their destination's native language, and travel spots can't translate everything in advance. In an effort to grant a versatile solution to this issue, our group proposes to develop an Android application that can convert captured images of foreign languages and translate them to another desired language. Using a combination of tools and resources ranging from past 18-551 projects and the concepts behind one of the newer additions to Google Goggles, we plan to not only implement a basic functionality to address the aforementioned problem, but also to add as much versatility as possible. These enhancements will allow the user to access more difficult-to-translate signs, such as those at abnormal angles, and possibly, even live, without needing to take a picture at all.

We plan to use OpenCV with the Android platform to collect input data and adjust the images; thresholding and image isolation to select the desired image text; Optical Character Recognition (OCR) to segment, train, recognize, and convert character images to ASCII; and the Google Translate API to translate the results to a more desirable output language. Depending on progress, we will implement as many additions as possible, such as deskewing, color adjusting, and language detection, among other possibilities.

THE PROBLEM

BACKGROUND

The problem we are addressing is that of text recognition and translation using computer vision – specifically, mobile embedded vision – and the internet. As mobile devices gain increasingly powerful processors and access to quicker networks, data input becomes the bottleneck. A typical user on a smartphone virtual keyboard has an input rate of 15 bits/s. The camera, found in virtually every mobile device today, is the highest-bandwidth input device; there, image extraction and recognition becomes crucial. The application we plan to address deals with image processing, and more specifically, text recognition.

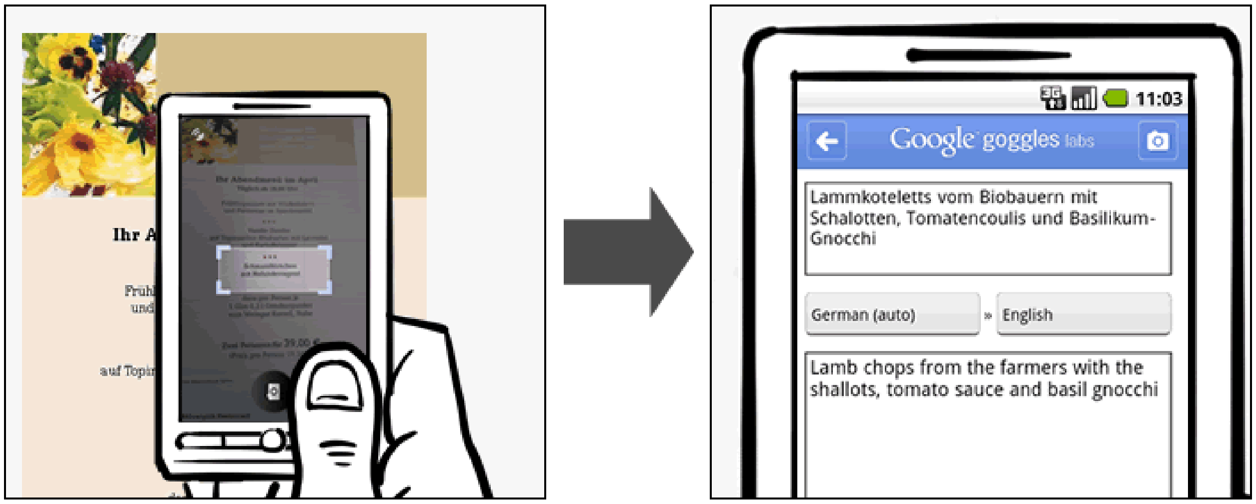
SPECIFICS

Today's world invites consumers to travel internationally and experience cultures other than their own. However, most tourists don't have the time or background required to understand the languages native to their destinations. While many famous sights abroad cater to tourists with multi-lingual instructions and information panels, the majority of the rest of the city generally only accommodates their own citizens. This restricts the amount of the culture the tourist can absorb without paying for a guide or learning the language themselves.

THE SOLUTION

BACKGROUND

Currently, Google Goggles allows users to translate text from captured images. The application works by taking a picture of a user-defined cropped area as input; depending on the quality of the picture, the application may or may not recognize the text. From there, the user may select an input and output language for translation.



Lammkoteletts vom Biobauern mit Schalotten, Tomatencoulis und Basilikum-Gnocchi

Lamb chops from the farmers with the shallots, tomato sauce and basil gnocchi

Figure 1: Google Goggles results for a translation scan.

NOVELTY

To start, our team proposes to develop an Android application that will recognize text and translate it, if necessary, using the Google Translate API in a similar fashion to that of Google Goggles. Once the application runs properly on basic inputs, we intend to improve the system by adding versatility. Innovative additions include but may not be limited to: a process to deskew input images captured at odd angles, the acceptance of multiple font/background color combinations, and, more ambitiously, the option for live translation using a video feed.

PREVIOUSLY IN 18-551

So far, no previous 18-551 group has tackled the same project we propose. However, there have been a few past projects that implemented systems similar to the ones we will need to complete pieces of our application.

SPRING 2011 GROUP 6

The Automatic LP Digitalization group worked with OCR to convert album labels in their project. While our project focuses more heavily on this area, we may be able to use some of their experimental findings to anticipate some of the problems that might arise during the process (i.e. “EX” being read as one letter).

FALL 2007 GROUP 3

The Cursive Handwriting Segmentation and Character Recognition group had several steps that we can use when developing the majority of our project. As their project involved cursive input, their word and character segmentation steps will be different than ours; however, we may be able to use their word segmentation process to separate characters in our own process. The main detail we will look at in their project, though, will be the character recognition tool they implemented: Support Vector Machines. SVM, as they describe it, has a good deal in common with PCA, so we may decide to look into it further during the pattern recognition step of our project.

This group also used thresholding to convert input images to grayscale, and normalization to eliminate slanting and skew from different inputs; we will need to account for both of these steps when implementing the respective additions to our project. They implemented a few processes to improve efficiency as well, including zero padding, a helpful detail to restrict pixel accessing during the preprocessing stage, and thinning, a process to allow for more effective segmentation and recognition for later steps. These processes may or may not be entirely pertinent to our project, but the theory behind them might ease our progress along.

OTHER HANDWRITING PROJECTS

Group 9 of Spring 2005 converted handwriting to ASCII characters, they focused more on the handwriting side of the process, allowing strokes to be part of the input. Since our project will not have that data, only pieces of their project will be relevant, and the majority of these parts are already mentioned in Fall 2007 Group 3's report.

Group 5 of Spring 2003 used methods similar to group 3 of Fall 2007; they have some notes on character segmentation that may help use recognize 'i' and 'j' as single characters, rather than two due to the dot.

Group 8 of Spring 2002 also worked on converting handwriting to ASCII text; their project is very similar to that of Group 5 in Spring 2003.

LICENSE PLATE RECOGNITION PROJECTS

Group 8 in Spring 2003 worked on reading and recognizing license plates. While this is clearly different from our project, their character recognition system implemented a technique called Adaptive Binarization to enhance text and background contrast, which may prove useful in our project. They also seemed to use a different character segmentation method than the handwriting groups, and we will need to weigh the pros and cons of both methods before choosing one to implement in our own application.

Group 18 in Spring 2000 also worked on license plate recognition; their methods were very similar to that of the above group, but they implemented a Histogram Equalization that may prove useful for locating and contrasting the words we wish to translate within the entire image field.

WHAT WE'LL DO

DATABASE

To begin, we will continue our in-depth research on relevant applications already in place, such as Google Goggles and a range of online OCR tools. From the documentation available, we will gain a better understanding of what can be used directly, what can be translated from one programming language to another, and what we will need to create on our own, from scratch.

Initially, we plan to get the entire process to work using the simplest input available: a direct camera image of English words in a standard, capitalized font, with black text on a white background. We expect to find documentation on similar systems that will help this process, and while a direct implementation may not be available, we will be able to combine various parts of already-developed programs in order to construct the framework of our project. These pieces will all affect one or more of our tasks; the more parts we can implement, the more versatility we can add to the application once the initial case works properly.

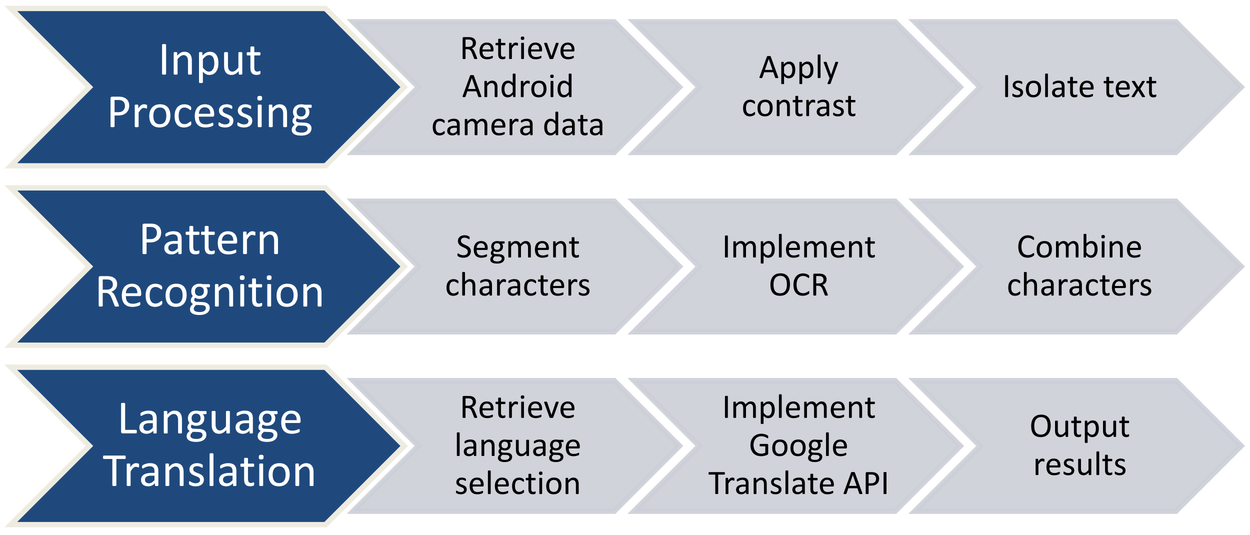


Figure 2: Step-by-step list showing the intended steps our application will perform.

LIST OF TASKS

- Basic framework
  - Implement OCR for text recognition
    - Capturing input and segmenting images: camera data → character images
      - Retrieve camera data from the tablet
      - Isolate image text using thresholding and connected component analysis
      - Apply segmentation and image scissoring to isolate character images
    - Pattern recognition: character images → ASCII characters (and words)
      - Implement OCR Engine and train program to get ASCII representations of the images
      - Combine characters to form the original text words
      - Confirm that the results match the input with a low error rate (<10%)
  - Implement translation: input language → output language
    - Create menu for language selection
    - Establish Google Translate API plugin
    - Confirm translation success with low error rate (<10%)
- Additions for versatility (dependent on progress)
  - Add capability to recognize lowercase letters and multiple fonts
  - Implement a process to deskew camera input for indirect capturing angles
  - Train OCR to work regardless of text/background colors
  - Allow application to work for multiple input (and output) languages
  - Make language menu optional, implementing the Google Translate language detection
  - Allow application to run on a video feed
    - Continuous input without needing to capture an image
    - Dynamic output generated and posted live

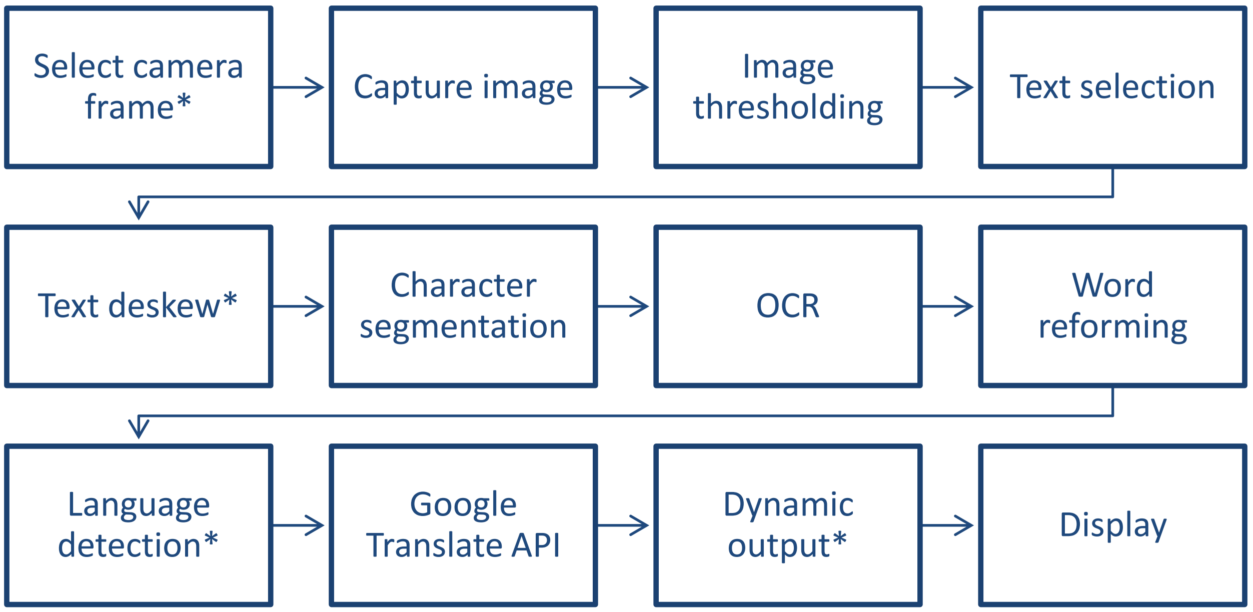


Figure 3: Flow graph showing the path through our system, including possible additions (marked with *).

DEMO

The final demonstration should at least show that our application can translate a simple word from a camera image. Depending on our progress, the demo will also display any and all additions we were able to implement to make the application more versatile.

HARDWARE/PURCHASES

As we intend to use only the ZOOM tablet, we do not anticipate requiring any additional hardware. Depending on our progress, though, we may need to evaluate the quality of the camera input; if the resolution is simply too difficult to work with, we will need to consider using a better camera to capture the input images.

We will, however, need to access the Google Translate API for the text translation end of the project. As we do not expect our project to require a large number of translations over the course of the semester, the usage fee should not go over $20.

Also, depending on our progress, we may decide to incorporate a language-detection feature to the menu. For this, we would implement the language detection feature of the Google Translate API, which, under similar constraints as the translation usage, will not go over $20.

SCHEDULE

We plan to all work on each part of the project, though each group member will focus mainly on one area of the basic framework to research (as referenced in the list of tasks):

Alex – Google Translate API / Google Goggles
James – Image Isolation & Segmentation
Tom – Pattern Recognition

Week	*Task(s)*
7	Image Isolation
8	Image Segmentation
9	Character Image → Character (Pattern Recognition)
10	Prepare for mid-project update, finish Pattern Recognition
11	Implement Google Translate API, create language menu
12	Finalize GUI, begin to add versatility (lowercase letters, additional fonts)
13	Add deskewing functionality
14	Prepare for demonstration, add languages, finish other tasks, add coloring functionality

REFERENCES

BASIC OCR SYSTEM RESOURCES:

Basic OCR Overview with code: http://blog.damiles.com/2008/11/basic-ocr-in-opencv/
Random OpenCV Instructions for Android phone: http://www.cs.bgu.ac.il/~tcv121/wiki.files/Instructions.pdf
iOS Computer Vision: http://aptogo.co.uk/2012/01/jump-into-our-new-tech-series-computer-vision/
Overview of Mobile CV: http://www.bdti.com/MyBDTI/pubs/201104_compvision.pdf
Capstone Ideas: http://blog.bumblebeelabs.com/the-killer-app-for-iphone-3gs/
Automatic License Plate Recognition using Pythonand OpenCV Paper: http://sajjad.in/content/ALPR_paper.pdf