*Project Proposal:* **Receipt Analyzer**

*due:*  3/26/2015

*authors:* Nina Baculinao (nb2406) & Melanie Hsu (mlh2197)



Source: "The bill" by Peter Merholz from Berkeley, CA, United States - the bill -- thank you adaptive path!. Licensed under CC BY-SA 2.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:The\_bill.jpg#/media/File:The\_bill.jpg

**Description**

For our final project, we plan to analyze restaurant receipts by splitting them into blocks of text and numbers, in order to ignore the text and extract the individual item price values. From there, we hope to give the user the ability to color code the different price values so they correlate with the person who ordered that dish, and use a simple calculator function to determine how much each person owes toward paying off the entire bill. The target software for this project is the latest version of iOS with OpenCV.

The program of analyzing a paper receipt constitutes a visual interface because it requires a system to accept and manipulate visual input from a real time camera in order to direct further computer processing. This system shall enable an iOS device to interpret numerical receipts and come up with a final deicision, thereby helping humans with the mental math of floating division and accounting for tax and tip.

**Limits**

In order to make our investigation more manageable, we have restricted our genre of OCR processing to restaurant receipts and reading number digits (typed, not handwritten). We will begin with a static training set of sample images. After we have achieved satisfactory results with our numerical and positional classifiers for the sample itemized receipts, then we will test images taken on the spot with our mobile application.

Lighting, camera resolution and focus, skew, and processing power all contribute to the difficulties in developing effective OCR applications using mobile phones. Therefore, we have also placed other constraints, mostly in the area of domain engineering. The image should be taken by an iOS rear-facing camera at a bird’s eye view, while the receipt is laid flat and as close to possible at a 0 degrees orientation on the table. The device limitation is not inconvenient for us, as we are both iPhone 6 users. There should be high visibility lighting during the photo taking, so the black print of the receipt is easily distinguishable from the white paper. Ideally, the background should be dark, in order to contrast with the white receipt. Glare and shadows should be avoided at all costs as it can hide some of the print details, even if flash is needed. The receipt should be rectangular, not ripped or crumpled, and not covered by any obstructions. The photo should be crisp and taken with a firm and unshaky hand.

**Methods & Results**

1. **Prepare training data**
   1. Take (or find a database of) 25 photos that follow the domain engineering constraints described above.
2. **Perform pre-processing (signal acquisition and filtering)**
   1. Convert to grayscale.
   2. Detect edges.
   3. Use the edges in the image to find the contour (outline) representing the piece of paper being scanned.
   4. Apply a perspective transform to fix the skew and obtain the normalized top-down view of the receipt with the background removed.
3. **Train our classifier and extract features**
   1. Convert pre-processed image to a characteristic vector of features (the relevant numbers we want from the restaurant receipt), either using the pixels matrix or OpenCV library’s contours funcitons
   2. There also exist open source OCR engines like Tesseract (backed by Google) that are supposed to be quite accurate – we can compare results, or just use this, if we are pressed for time and understand how the algorithms work
   3. Find the regions (positions) for each itemized price, the tax, and total price.
   4. Use OCR (either Tesseract or our own system) to read the numerical digits
4. **Crunch the numbers**
   1. Allow users to identify via touch which items are bought by different individuals, and which items are bought by the group and should be evenly divided
      1. For a simpler user interface, consider not attributing items to people’s names, but rather shapes and colors, i.e. green ring for group purchase, different color circles for different individuals
   2. Allow users to decide the tip amount
   3. Return to users the amount each individual should be paying in terms of dish, tax and tip so that all the individual subtotals equal an appropriate total
5. **Evaluate the system using test data**
   1. Count the TP, TN, FP, FN cases on a receipt-by-receipt results, as well as a character-by-character basis

**Evaluation Metric**

We hope to run 25 test trials, and achieve at least 7 True-Positive/True-Negative cases, as well as 4 False-Positive/False-Negative that we can use as a baseline for areas of improvement. For each number-related digit that appears in the receipt ($12.99 is considered 6 characters), we will count the number of TP, TN, FP, FN and use Precision, Recall and the F1 Harmonic Mean to evaluate whether this is considered good performance. If we can achieve above 60% overall performance in all of these areas, we would consider this project a great success. We also hope to learn from our failures how we can improve our classifier and preprocessing to make this a more robust mobile application that can potentially be of use to any group of restaurant-goers.

**Management Plan**

1. **Prepare training data:** We will make efforts to find/obtain/produce training data together for Steps 1 and 5. Since it may be unfeasible for us to eat out a total of 50 times by May, we may have to look for online databases. Melanie has worked more on capturing and saving images with the camera interface in the iPhone, she can set up the image picker controller class.
2. **Perform pre-processing:** As for Step 2, Nina will handle the image pre-processing and locating of the receipt on a dark background.
3. **Train classifier and extract features:** Together, Nina and Melanie will research the best way to extract features for the OCR of the numbers in Step 3, either using the Tesseract engine or our own as this is the most critical and theory-rich step. Depending on time, we may perform two separate implementations for comparison.
4. **Crunch the numbers:** This task can be divided between Storyboard implementations of the buttons for user interaction (Melanie), and the function calculations (Nina).
5. **Evaluate the system using test data:** Nina can refactor her Python code from Assignment 2 to evaluate the Precision and Recall for this data set if we can put the correct answers and system answers in CSV files, as this evaluation step does not need to be built into the mobile app itself and exists solely for development and assessment purposes.

**References**

Barczak, A. L. C., Johnson, M. J., & Messom, C. H. (2007). Revisiting Moment Invariants: Rapid Feature Extraction and Classification for Handwritten Digits. *Computing*, 137–142. Retrieved from http://digital.liby.waikato.ac.nz/conferences/ivcnz07/papers/ivcnz07-paper26.pdf

Beitzel, S. M., Jensen, E. C., & Grossman, D. A. (2003). Retrieving OCR Text. In *Symposium on Document Image Understanding Technologies (SDUIT)*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.3743

Bissacco, A., Cummins, M., Netzer, Y., & Neven, H. (2013). PhotoOCR: Reading text in uncontrolled conditions. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 785–792). Institute of Electrical and Electronics Engineers Inc.

Chen, X. C. X., & Yuille, A. L. (2004). Detecting and reading text in natural scenes. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, *2*.

Garain, U., Jain, A., Maity, A., & Chanda, B. (2008). Machine reading of camera-held low quality text images: An ICA-based image enhancement approach for improving OCR accuracy. *2008 19th International Conference on Pattern Recognition*.

Heikkonen, J., & Mäntynen, M. (1996). A computer vision approach to digit recognition on pulp bales. *Pattern Recognition Letters*, *17*(4 SPEC. ISS.), 413–419.

Hymes, K., & Lewin, J. (2009). OCR for Mobile Phones. *Stanford Feb*, 1–5. doi:10.1.1.217.5652

Laine, M., & Nevalainen, O. S. (2006). A standalone OCR system for mobile cameraphones. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*.

Netzer, Y., & Wang, T. (2011). Reading digits in natural images with unsupervised feature learning. *… Unsupervised Feature …*, 1–9. Retrieved from http://research.google.com/pubs/archive/37648.pdf

Neumann, L., & Matas, J. (2012). Real-time scene text localization and recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 3538–3545).

Sandberg, J. S. (1995). Counting {OCR} errors in typeset text. *Proceedings of SPIE --- The International Society for Optical Engineering*, *2422*, 184–195.

Seeger, M., & Dance, C. (2001). Binarising camera images for OCR. *Proceedings of Sixth International Conference on Document Analysis and Recognition*.

Smith, R. (2007). An overview of the tesseract OCR engine. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* (Vol. 2, pp. 629–633).

Sun, W., Liu, L.-M., Zhang, W., & Comfort, J. C. (1992). Intelligent OCR processing. *Journal of the American Society for Information Science*, *43*(6), 422–431. Retrieved from http://www3.interscience.wiley.com/cgi-bin/jtoc/27981/

Xu, Y. (1999). Prototype extraction and adaptive OCR. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(12), 1280–1296.

Yamazaki, S., Furukori, F., Zhao, Q., Shirai, K., & Okamoto, M. (2011). Embedding a mathematical OCR module into OCRopus. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* (pp. 880–884).

**Links & Tutorials (Self Reference)**

* 2014: <http://www.intorobotics.com/tutorials-setup-opencv-for-android-ios-linux-and-windows/>
* 2014: https://github.com/Itseez/opencv\_contrib/tree/master/modules/text/samples
* 2013: <http://docs.opencv.org/trunk/doc/tutorials/ios/table_of_content_ios/table_of_content_ios.html>
* 2013: <http://stackoverflow.com/questions/9413216/simple-digit-recognition-ocr-in-opencv-python>
* 2008: <http://blog.damiles.com/2008/11/basic-ocr-in-opencv/>

<http://www.raywenderlich.com/59602/make-augmented-reality-target-shooter-game-opencv-part-1>

<https://www.indiegogo.com/projects/bistro-a-smart-feeder-recognizes-your-cat-s-face>

<https://github.com/harthur/kittydar>

Flask Image Search: <http://www.pyimagesearch.com/2014/12/08/adding-web-interface-image-search-engine-flask/>

Document scanner (Python): http://www.pyimagesearch.com/2014/09/01/build-kick-ass-mobile-document-scanner-just-5-minutes/

**REQUIREMENTS**

**COURSE INTRO:**

*This course is a survey of existing research systems that allow computers to accept and manipulate visual input, usually from real time cameras, in order to direct further computer processing. These systems presently enable machines to: recognize and interpret human hand and body gestures, generate natural language descriptions of medical or map imagery, index into a database of pictures to retrieve related images, analyze imagery such as fingerprint or iris patterns for security data, monitor large outdoor areas for types of activity, steer automobiles automatically, summarize video sequences, and execute other and more exotic tasks.*

**PROPOSAL:** *The phenomena or program to be investigated, and why it should be considered to be an example of a visual interface. If you are interested in doing something that deals with the processing of visual input, but does not lead to a decision (for example, you want to investigate one of the many "black boxes" of computer vision that transform one image into another kind of image), please feel free to suggest it in the proposal; maybe we can work something out. But please do not recycle prior systems or papers. If you are incorporating work you have already done or are doing for this or another class, you must declare it up front. In every case, your grade will depend on how much you have added to an existing 4 baseline, even if that baseline is your own prior work.*

**LIMITS:** *A description of the limits you have placed on the investigation. This is most important for programs, but even papers can get out of hand. In past experience, most initial project proposals are approximately ten times too ambitious. For a project, please also note whatever environmental restrictions you expect to be met, if it is to be demonstrated in real time, so we can anticipate a satisfactory place for the demo. My office, my lab, or your dorm room (with advance notice) are acceptable; if it is a mobile app, we need to know in advance how mobile it will be. In this section you must indicate any special help that may be required, such as exotic references, equipment, or environmental circumstances. If such help is necessary but unattainable, the proposal will have to be denied.*

**Methods & Results:** *Two page sketch of anticipated methods and results. How you will measure program performance. You are not held to this as a promise, but past experience has shown that thinking about how to get to a final conclusion will help you decide among your topics and approaches.*

**Evaluation Metric:** *You must state what you consider acceptable performance, how you will measure it, how many trials you will run, and what you expect to learn from any failures.*

**MANAGEMENT PLAN:** *If you are proposing a team, you must indicate clearly a management plan: who will be responsible for what aspect of the paper of projects, and any timing dependences. Do not say that all aspects will be developed jointly throughout; in practice this never happens. Please think this through ahead of time. Teams are limited to two people only, except with advance permission; in the past, teams of three or more have rarely worked well.*

**REFERENCES:**

*At least 4 major references: either textbooks, journal articles, or (possibly) computer code. Wikipedia articles or equivalents do not count! This must be a part of the proposal: start now.*