COMS 6998 Computational Photography Spring 2009

BREAKING AN IMAGE BASED CAPTCHA

INTERMEDIATE REPORT

Michele Merler(mm3233)
Jacquilene Jacob (jj2442)

First Step: Data acquisition

We wrote a Perl script to download 200 Vidoop challenges from their website. The images can be found from our project page, together with .txt files containing the correspondent categories required by the challenge and manually annotated ground truth letters that actually solve the test, and the results of the split and letter detection algorithm,.

We discovered that only 26 categories are used in the challenges. Their distribution can be observed in the graph of Figure 1.

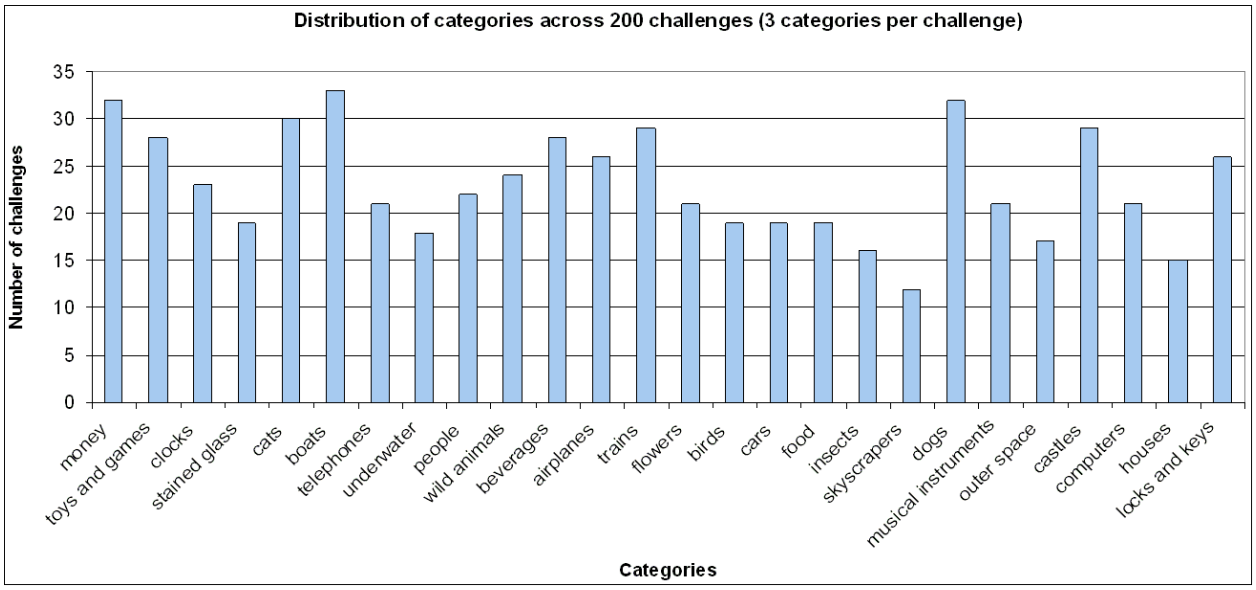

Figure 1 : Distribution of 26 categories across 600 requests in 200 Vidoop challenges

We also wrote another Perl script to download images from Flickr for every category, in order to use them as training data. In this context, we decided to download 500 images per concept, a number large enough to train a fairly robust classifier, but small enough to prevent too many noisy examples to be in the training set. In fact, downloading images from Flickr allows to automatically obtain a large scale of data, but many examples might not be relevant to the given query. Flickr's query system relies on users tagging or other text labeling of the images, rather than on their actual content, therefore mislabeling by users can lead to errors, which increase as we proceed to lower rankings in the returned list of results.

Test images preprocessing

The goal of this step is to split each challenge image into the correct subimages, and then localize and extract the circular region containing the character within each subimage. The split algorithm we use is based on localizing vertical and horizontal lines containing the

maximum number of edges in the edge image obtained by applying a Laplacian of Gaussian filter to the original challenge image. Once the image had been split into the subimages, a generalized Hough transform (we found the code here[1]) is computed on each subimage to detect circular regions. The circular region which is detected in most of the subimages in approximately the same position and with same radius is kept to be the character's region. Finally, the rectangle with equal sides of length l = r/sqrt(2) inscribed in the localized circle of radius r is the final character region, which is thresholded into a binary representation. The algorithm, while being simple and a little bit as hoc, is quite effective. In fact, it splits and segments subimages and text regions with 100% accuracy. Figure 2 presents an example of the processing chain for Challenge1.

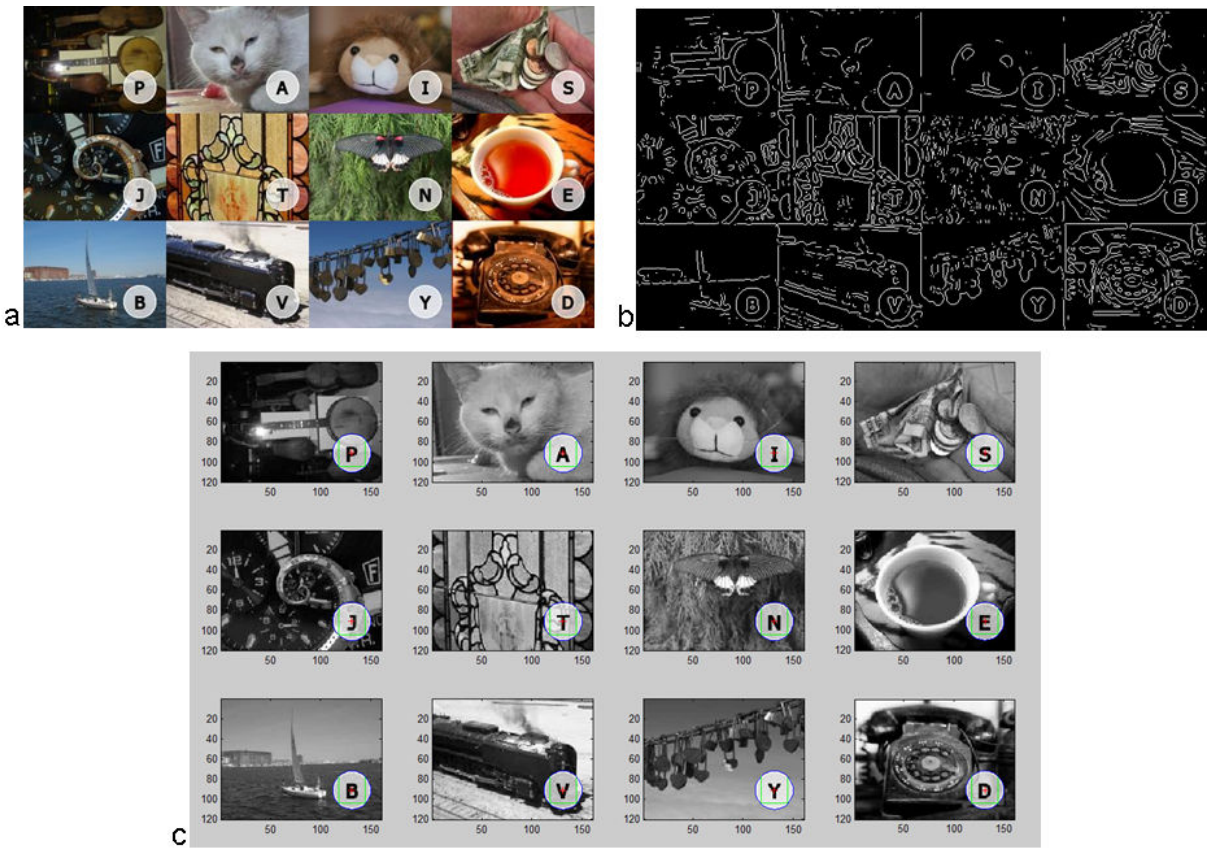

Figure 2 : Preprocessing chain: a) original test image, b) LoG based egde image, c) split and circle detection result.

[1] http://www.mathworks.com/matlabcentral/fileexchange/9168

Features extraction

We are extracting color histogram, edge histogram and color moments features to train and test classifers as in Assignment 2. We are still in the process of extracting and testing the results, which will be uploaded soon.

Next Steps

- Besides extracting and testing the proposed features, we will try to explore more elaborate features (maybe SIFT in a visual codebook fashion)
- We need to build the character recognition classifier. The options are using an existing OCR system (for example Tesseract[2]) or build our simple one.
- Once the offline evaluations are done, we'll implement the online version of the system

[2] http://code.google.com/p/tesseract-ocr/