Project Proposal

Scanning Text from Phone Captured Image

Yuchen Tong 1003534669
Junming Zhang 1003988982

1. **What is the problem exactly and what to achieve?**
   Optical Character Recognition (OSR) is the electronic or mechanical conversion of images of typed, handwritten, or printed text into machine-encoded text from any source. (https://en.wikipedia.org/wiki/Optical_character_recognition)
   What we try to solve here is the sub-problem of OSR, where we mainly focus on recognizing typed text from the captured image of any form of document from a phone. Our goal is to design an algorithm such that given a well-conditioned image of a document (or a distorted image) as an input, it precisely detects the text in the image (or automatically corrects main errors) and outputs the text correctly.

2. **How is the problem relevant to this course?**
   Since the inputs are images, in order to extract important information from the image, image processing techniques (some are introduced in this lecture) listed below are required:
   a. Denoise the image (Gaussian Smoothing)
   b. Image Filtering (Apply filters to make the characters stand out)
   c. Binarize an image (Get the binary image using thresholding and grey scaling)
   d. Contour Detection
   e. De-skew (Warp the image using Homography based on the capturing angle)

   Some other general processing techniques such as matrix matching (pattern matching, image correlation), feature extraction (extract features from input), image segmentation, feature detection (using KNN, CNN) are also relevant.

3. **What have others tried to solve this problem?**
   Existing official implementations such as **CuneiForm**, **Tesseract**, and **OCRopus** can achieve high accuracy in general OSR including the problem we proposed to solve. Where:
   - **CuneiForm** and **Tesseract** use the "two-pass" approach on character recognition, which focus on the shape of the character and succeed on distorted characters (Ray Smith (2007). "An Overview of the Tesseract OCR Engine").
   - **Tesseract** and **OCRopus** use a trained neural network to recognize the whole text instead of character by character (https://en.wikipedia.org/wiki/Optical_character_recognition)

4. **What approach are we going to try and why do we think it will work well?**
   Our Approach (preprocessing and text recognition):
   a. Image preprocessing (Denoising, De-skew, Binarize)
   b. Characters, digits, symbols patch extraction (Contour Detection)
   c. Image segmentation (Using the bounding box to cut each feature patch out)
   d. Create training data (labeled feature image patch)

e. Feature extraction and detection (Training and Testing ML models, ex., KNN), this can train model to recognize large amount of text and new fonts.
f. Text detection (by matrix matching, comparing to stored image pixel by pixel, used to recognize one character by one character) to improve and check the result. This works well for typewritten characters but poor on new fonts.
g. Output sentences.

We think it will work well since:

a. We preprocess the image to make text detection easier.
b. ML method (like KNN) helps to find the closest match from the stored labels. We can tune the hyperparameters to improve the performance.
c. Doing feature recognition using ML models is more generalized and reliable.
d. We work at the character level which has relatively low variance, so the detection could be more accurate.
e. The amount of English characters, digits, and symbols is relatively small, which makes building training data set easier.
f. The character by character check (text detection) helps to tune the ML model, which improves the accuracy of prediction for the model and may contribute to enlarge the training set.

5. What steps (task list) are required?

- **Image preprocessing:**
  a. Get grayscale image
  b. Denoising image using Gaussian smoothing.
  c. De-skew image using homography
  d. Binarize the image using adaptive thresholding.
- **Feature patch extraction:**
  a. Calculate contours
  b. Threshold the contours to get clear text
  c. Using bounding box to bound each character
  d. Cut out the bounding box patches
  e. Resize the patch into same resolution
- **Create training data sets using steps 1 and 2 above:**
  a. Label the feature patch manually
  b. Train the ML model (i.e. KNN or Deep Neural Network)
  c. Test the model using the newly captured image (recognize characters by the ML model)
  d. Use the matrix matching to check the prediction
  e. Generate words and sentences based on the distance of each bounding box calculated from step 2
  f. Output the texts