Ministry Category : Department of Space (ISRO)
Problem Statement : Text Extractor **Problem Code :** #ISR3
Team Leader Name : Chaitanya Tejaswi **College Code :** #2451

Idea Description

The user will be able to detect, extract and upload textual information from a set of *community assets* images, using a GUI application on a PC.
The 6 main stages are described below –

1. **Image Capture** - capturing the image on an Android smartphone
- We are using an android app - Camera FV-5 Lite for capturing the image. This is the free distribution of the Camera FV-5 app.
- We are using it to obtain:

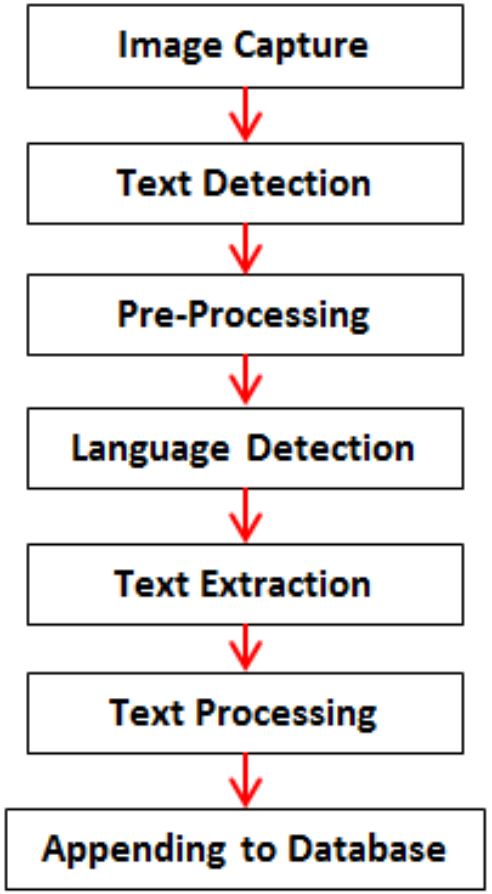
1. PNG images: For OCR, PNGs offer a much better text output compared to JPEGs. This is because PNGs are a lossless encoding format, in contrast to JPEGs, which are lossy.
However, JPEG files, if selected, will also be processed.
2. Geo-tagging information: Setting the "Geo-Tagging" option allows the user to directly obtain location coordinates (along with other metadata) in an EXIF/XMP file, instead of entering them manually.
We require both the image file (PNG or JPEG) and the metafile (EXIF/XMP) for further use.
3. Rotation Correction: Store image data according to correct orientation (Portrait/Landscape).

2. **Text Detection** - determining if there is text in the image
- Using the Extremal-Region-Filter Algorithm, as described here, a custom implementation in Python will be made; sample given here.
- For testing, we have used the Street View Text (SVT) dataset.
- If text is detected, this stage will also store the first 3 characters from the image. This will be used later for "Language Detection".

*[We furnish some results for the dataset on your website]

3. **Pre-processing** - making the image more suitable for character recognition.
- We are using the opencv-python module for cropping images, and image enhancement of textual regions.

Flow Diagram



4. Language Detection

- From the pre-processed image, 3 characters will be tested for language detection. We'll use Python's langdetect module for detection of regional language - en,hi,gu (English,Hindi,Gujarati).
- Once the language is determined, the actual OCR will take place.

5. Text Extraction (OCR) - extracting text from image

- This stage is implemented using the Tesseract-OCR API.
- Using a Python wrapper, the segmented image will be sent for evaluation by the libtesseract OCR engine, where it will be converted to text.
- The OCR engine will be trained on the provided dataset (for recognising Hindi & Gujarati text) using the training sets (for Hindi & Gujarati) provided by the developers.

5. Text Processing - making sense out of obtained text

- Use of Indic Language NLP library will be made in an attempt to free the extracted text from errors.
- The output can be stored in native text form, as well as Romanized text, as required. However, the text-string to be fed into the database will be Romanized text.
- This is done to save space and avoid unwanted deletion of data.
- If the user desires to view the text in native script only, it can be done using the GUI option - "See Original".

6. Appending to Database (along with geographic co-ordinates)

- It is assumed that the geo-tagging information is already provided to the GUI. It may be validated using the option - "Verify Co-ordinates".
- This option can be used to look up the entered location using the Google Maps API on a Web Browser.
- The option - "Submit" completes the process by adding an entry to the database.

Once all the entries have been made, the records can be uploaded to the server (preferably in CSV file format).

Technology Stack

- PyQt4 module *(for GUI application)*
- Camera FV-5 Lite Android App *(for image capture & geo-tagging)*
- numpy, opencv-python modules *(for image processing)*
- langdetect module *(for language detection)*
- tesseract-ocr API *(for OCR)*
- Indic Language NLP module *(for text processing)*
- MySQL API *(for managing image database)*
- Notepad++ Editor

Indic Script & Romanized Script Example

```
from indicnlp.transliterate.unicode_transliterate import ItransTransliterator

input_text=u'राजस्थान'
lang='hi'


print ItransTransliterator.to_itrans(input_text,lang)
```

rAjasthAna ← Indic_Text-to-Romanized_Text

```
from indicnlp.transliterate.unicode_transliterate import ItransTransliterator


# input_text=u'rajasthAna'
input_text=u'pitL^In'
lang='hi'
x=ItransTransliterator.from_itrans(input_text,lang)
```

पितॄन् ← Romanized_Text-to-Indic_Text

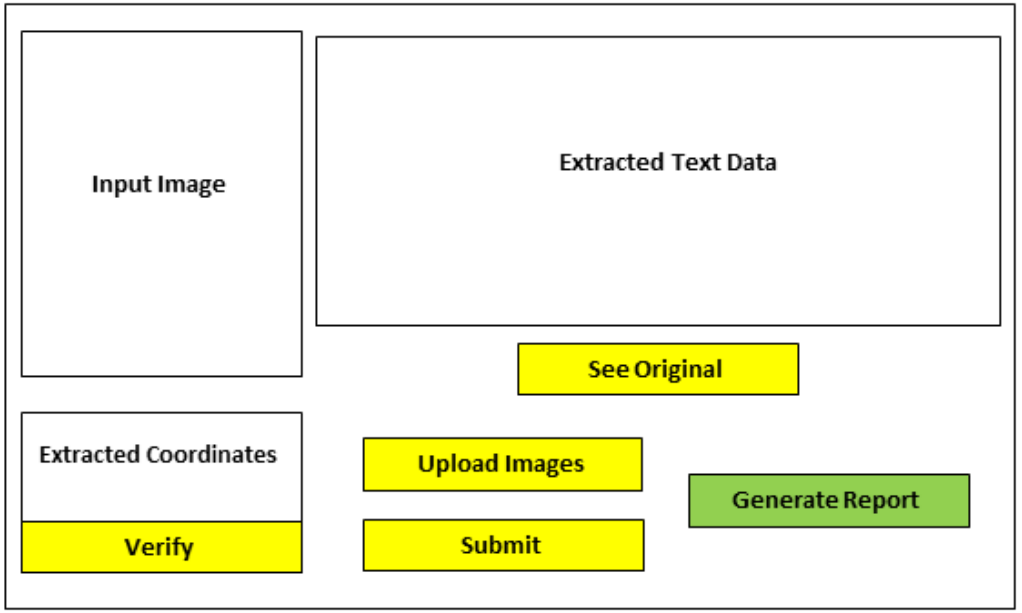
Users

Office Staff, who will upload the images for geo-tagging.

Use-Case

- The User will specify the set of images & also provide geo-tagging information (either manually, or through an EXIF/XMP file).
  (See the GUI application template to the side)
- If only the image file is provided, the “Extracted Coordinates ” would be blank, and the user will be asked to explicitly specify the co-ordinates.
- If the user clicks on “Verify”, a browser window opens up, and he/she can check for authenticity of the location. If they find it to not be true, they can manually re-enter the co-ordinates. But for every such manual entry, the report will attach an “EDITED” label to it.
- After this, the user will click “Submit”.
- Once all the images have been processed, the “Generate Report” button becomes active. On pressing it, a custom report will be generated which logs all the extracted data in a CSV file.

User Application Window



Dependencies

- Most of our application would be in Python, and the primary system dependency for it would be a Windows 7 system, with at least 1 GB of RAM.

Showstoppers

JPEG images

- As mentioned earlier, JPEG images are not suitable for optical character recognition. And that they are obtained from a mobile device does not help. But we do acknowledge the fact that most existing community asset images are JPEGs. Hence, we have chosen an approach that is more training dependent. And as expected, we would need a larger dataset of images for training the system before we could produce results of significant accuracy.

Text Detection

- We list 3 examples each of Hindi, Tamil & Telugu. All the images are JPEGs (as provided on the website).
- As can be seen, our current program does decent at isolating regions containing text, but character-bound regions are identified in case of English, Tamil & Telugu. Hindi text regions are not identified initially, which indicates that the final output of OCR engine in such cases is more prone to error.
- We are currently working on fixing this issue.