TEXT EXTRACTOR

Ministry Category: Department of Space (ISRO)

Problem Code:#ISR3

College Code:1-3324256325

Team Name: lazarus90

Team Leader: Sauradip Nag

SOLUTION APPROACH

- The Text Extractor Software will be Useful for any kinds of Environment (Both Offline and Online).The Offline Version will be slightly Less Accurate during Geo Tagging .The Online option is kept in case of any Important Location is needed for Security Purpose as Geo Tagging is Very Accurate. This can work both for Android Devices (KitKat and Above) and PC. User can Save the Data in Local Database or he will have an Option to Upload it on Cloud if it is Required by Government Of India
- **Step 1: File Selection**
- It requires the user to choose a Folder of his/her choice and select a Image/ Group of Image as shown in Fig. 1.(a) and 1.(b). User has the flexibility to choose either of 2 options Online / Offline as depicted in Fig. 2.The Online Version requires Internet to Locate Accurately the Geographical Position and also for translating complex Indic Language. The offline version will contain Prespecified Indian Locations and Regional Languages Manually stored in Database.The Offline Data is collected from Government Records in Official Sites and Open Source Datasets.
- **Step 2: Text Region Extraction (ROI)**
- After Taking the Image we extract the Metadata Available from the Image and we store it in a Temporary Location . If metadata contains Latitude and Longitude we perform offline Reverse Geocoding using the following package available at (https://github.com/thampiman/reverse-geocoder) .We then proceed to extract Region Location .We check if there are any Information Regarding Location, if found then we neglect using Geotagging with Online /Offline Mode. Next , For each Image we also found out Sobel and Canny Images .

Next we perform Text Region Identification from the Image .We grayscale the image and find out 2 features namely

1. *Magnitude Gradient(MG) :* Magnitude Gradient Feature enhances the Magnitude of Gradient and also increases the interclass distance between Strong and Weak edges as shown in Fig 3.(a) .

2. *Directional Coherence(DC)*. Directional Coherence calculates coherence of Dominant Gradient within Patch as in Fig. 3.(b)

We convert these MG and DC into Sparse Matrix and then fuse them. Next we perform k means clustering on used Image to identify text pixels. Now we extract 4 more Features to Find our ROI namely.

3. Histogram of Gradient(*HOG) :* After observing histogram of MG sliding window we calculate HOG Feature as shown in Fig .4.(a) to eliminate Non Text Pixels.

4. *Gradient Vector Flow Feature (GVF) :* For each pixel passing HOG step we use the Canny Image and extract GVF. Now we can distinguish slowly Text and NonText as Illustrated in Fig 4(b) and Fig 4(c).

5. *Stroke Width Transform(SWT) :* To get more clarity on the Region of Interest (ROI) we use this . SWT works well in natural Scene Images as is the case in this problemset.With atleast one Text Pixel we can Grow a Complete Character using SWT . So after this Step we get Partial Text Character / Full Character .as in Fig. 5 .We still get False Positives Here.

6. *Principal Component Analysis(PCA) :* We use this feature to Restore the lost Characters / Lost Boundary of Text component and Remove the False Positives we try to recover it and eliminate False Positives.

After this feature almost **95%** of False Positives are eliminated and we get only Text Region as in Fig 6.(a) .We then draw bounding box on remaining component and Crop out ROI.Traditional Existing Methods simply run OCR on whole image to detect text/mumbers , but in many Difficult cases like Scene Text , the nontext background may represent some character which will be misinterprated by OCR as character. With our proposed method we run OCR on ROI which is smaller hence lesser Time complexity and Higher Accuracy. Our Both Desktop and Android Version will use this Model to Extract ROI from any Image , thus Computationally Less Expensive.

Step 3 :Text Recognition and Translation

In this Step we will First use Gaussian Blur on the Image from Step 2 and we removed Noise from the Image .

We then have 2 choices for OCR andTranslator as given by user in Step 1 .If user chooses Online Mode ,we will use Open Source OCR in the form of API and GoogleTranslator . If user Chooses Offline Mode, we will use the Manually Trained Open Source Tesseract OCR Model using Training Datasets for Hindi and Regional language using Models available at (https://indic-ocr.github.io/tessdata/)and pretrained Translator using .csv files of publicly available dataset (https://github.com/umpirsky/language-list/tree/master/data) .After this step we will be having an array of words and numbers (words[i]) of any language translated to English identified from the ROI Image of Step 2 as Illustrated in Fig 6.(a) and 6.(b). Currently we trained Translator and OCR Model for Hindi Language Only .

For Language Translation in Android Device , we can use the Opensource NDK available at (https://github.com/artetxem/mitzuli).

The Drawbacks in this Step is Training of Language for OCR is not accurate enough in identifying text in Low Light Twisted Images.

Step 4 : Location and Regional Language Identification

- In this step we find out the Geographical Location Present in the Image. We can find a location by Geocoding information like Street Name , City Name , Pin Codes etc from the Image . For each image we have Location Target 3 Sources :
- a) Meta Data of Image (Offline Mode) : If Image MetaData contains Location we proceed to find Region using Reverse Geocoding and Find Regional Language from Regional Language Template (a Database Containing Predefined Regional Languages and Their Location)
- b) Using Offline Location Database (Offline Mode): This is possible using the official Datasets in Excel Format(.xls) publicly available at https://data.gov.in/catalog/all-india-pincode-directory which represents the GeoLocation Coordinates of Nearest Postoffice. Using this we can populate data into MYSQL Database as CITY AND STATES DB (CSDB) and also use the publicly available database as LONGLAT DB(LLDB) at https://gist.github.com/gsivaprabu/5336570 to map the Longitude and Latitude with CSDB. Hence if we have name of a Village in Image , it searches the CSDB and maps the State , City ,Town the village name is part of and then it query those in LLDB , so we can get the Closest PostOffice Location even if we do not find the Exact Village.
- c) Using GeoLocator (Online Mode) :We can find the Geographic Location , coordinates directly using Geopy Module which
internally uses Google's Location Api providing Latitude and Longitude , which is very Accurate
- Traditional Location Tracking make use of GeoLocation Api ,with our software we made an attempt to do this process Offline so as to save the Resources.Offline Resources Includes downloading Maps Offline but that is possible only for Android Device Currently
- For Android Device we can get Offline Location Information by name using the library available here (https://github.com/mapsme/api-android)

- For Regional Language Detection we can use 2 methods :
- a) Storing Regional Languages Template (Offline) :With the help of .CSV/.XML File available publicly at https://github.com/umpirsky/language-list and we can manually populate a Relational (SQL) / Non Relation (MONGODB) for the Regional Language as RLDB with [Columns like Language , ISO-Standard , Regions Spoken] and Embed it in Software
- b) Regional Language Api (Online) :We can use the same Pythons GeoPy Module to get Regional efficiently for a particular region detected by geopy module from the input image.
- Till Date , there are almost no collections of Indic Languages and Regions Spoken and with our .CSV/.XML File we can keep track of Regional Language for Future Use.
- After this stage we get Geo Location and Regional Language of Particular Location.
- **Step 5 : Result Representation and Storing**
- In this stage we represent the list of Inventories in the following ways
- 1) *GeoTagging the Images on Map(Requires Internet) :* In this method we simply represent the Image , Location Detected and Regional Language and tag this information at the Longitude and latitude given by Step 4 on Google Map as represented in Fig.7.(a)
- *2) GeoTagging the Images(Offline) :* In this method , we can Represent the Inventories on Image Itself and a url to the google maps location using ImageMap property.
- *3) Stuffing Metadata of Image(Offline) :* In this method , we modify the Meta data of the Image and Stuff Information like Location , and Tags like Regional language , Useful Texts etc
- *4) Storing in Database(Offline) :* We store the information in MONGODB due to its BSON Like Format of Storing Data

- We can store the Data in MongoDB in the following Format :
- **var** data = {
- _id: ObjectId("5099803df3f4948bd2f98391"),
- name: {doc :“ Hindi4.”},
- Extension :{ ext :“ jpg/jpeg” }
- date: **new** Date(‘December 31 , 2017’),
- Map: {Latitude :“26.71", Longitude:“108.56" },
- Regional Language : [“Rajasthani ”, “Hindi”],

  Regions Spoken : [{

  - Country :“India”
  - State :“ Rajasthan“
  - District :“Jodhpur”
  - Village/Other :“Luni ”
  - }],
- GeoTagged : { “yes” }
- }
- Representation is in Fig.7.(b)

We can Query from the LLDB and CSDB for GeoLocation in SQL as follows :

Select t1.latitude,t1.longitude,t2.pincode from LLDB t1 Inner join CSDB t2 on t2.divisionname = t1.city_name where t2.Taluk = ‘ “ ‘+words[i]+’ ” ‘ limit 1;

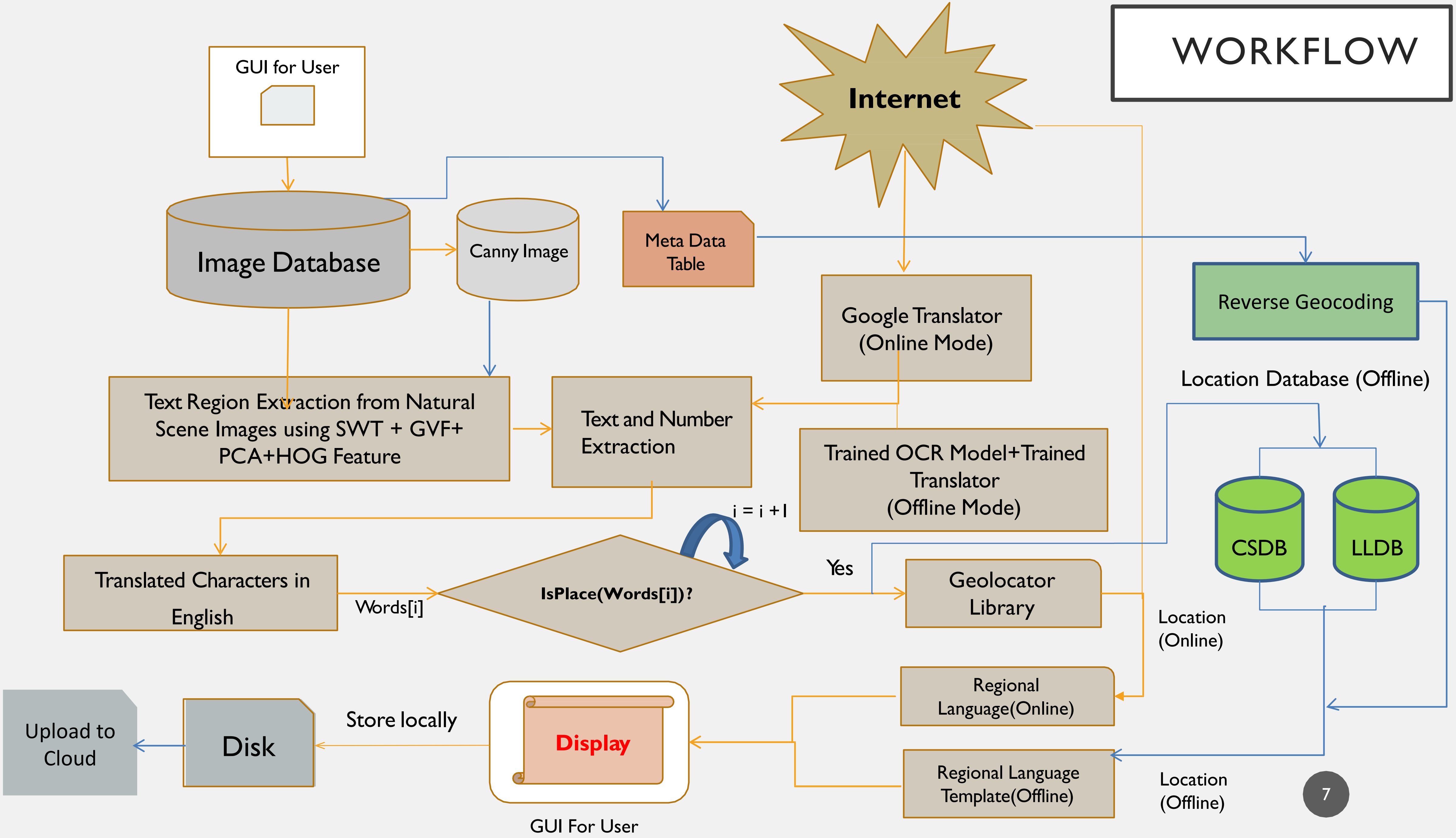
If words[i] = Luni then this above Query returns

Latitude	Longitude	Pincode
26.29	73.02	342802

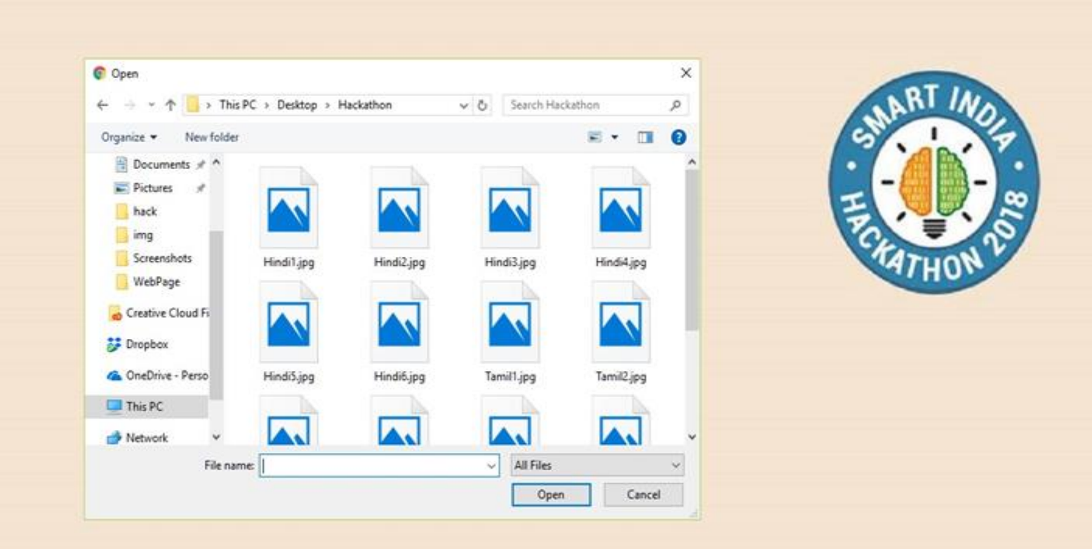
This gives us Location Accuracy upto +/- 5 Kms

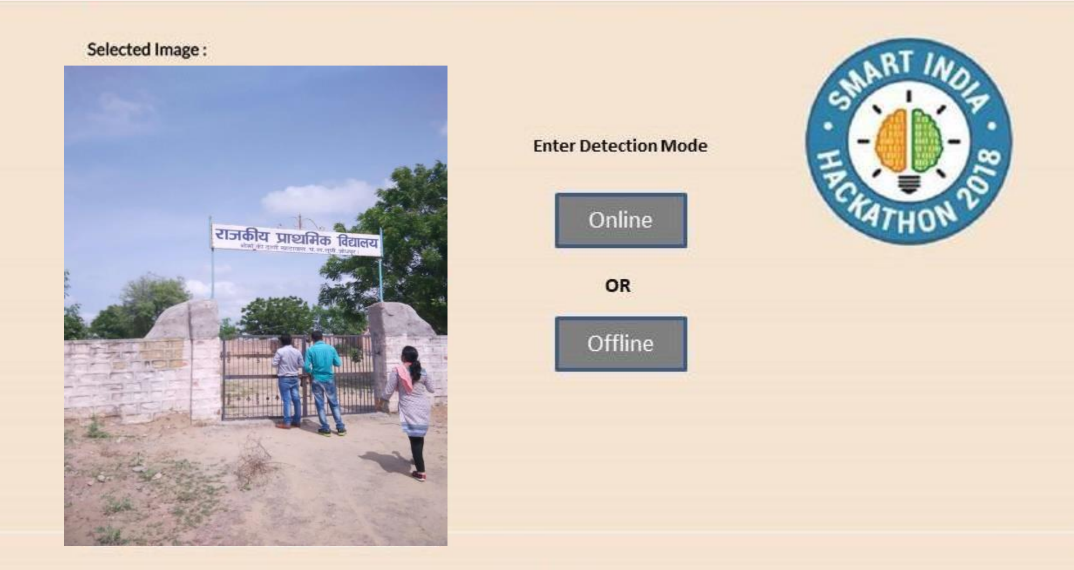
Using this Information we will Query in RLDB to get the Regional Language for a Particular Region

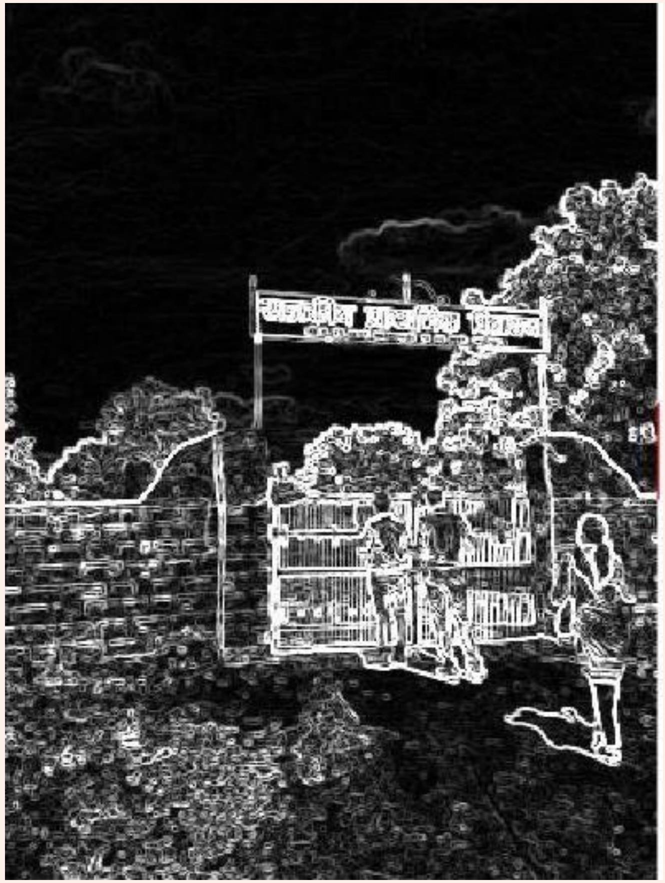
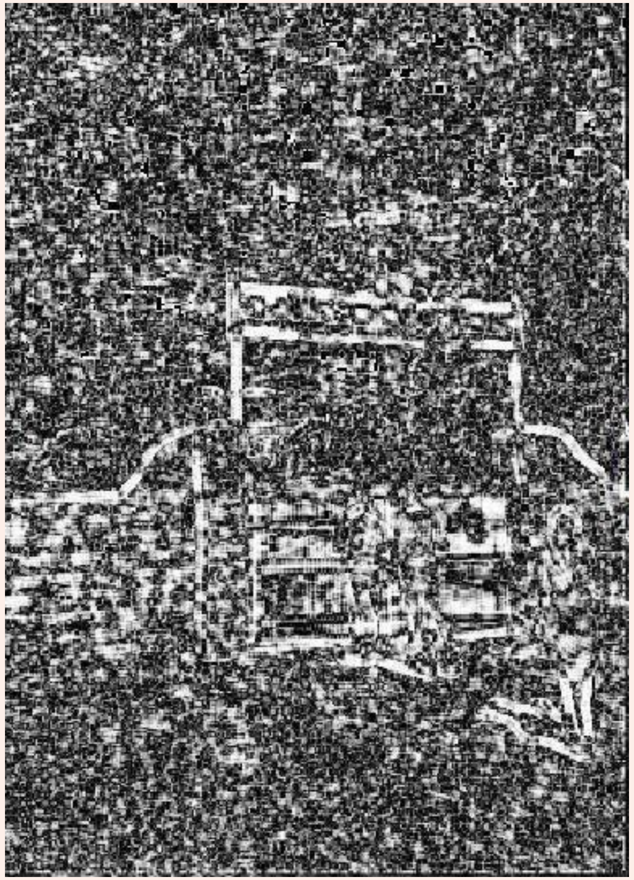
WORKFLOW

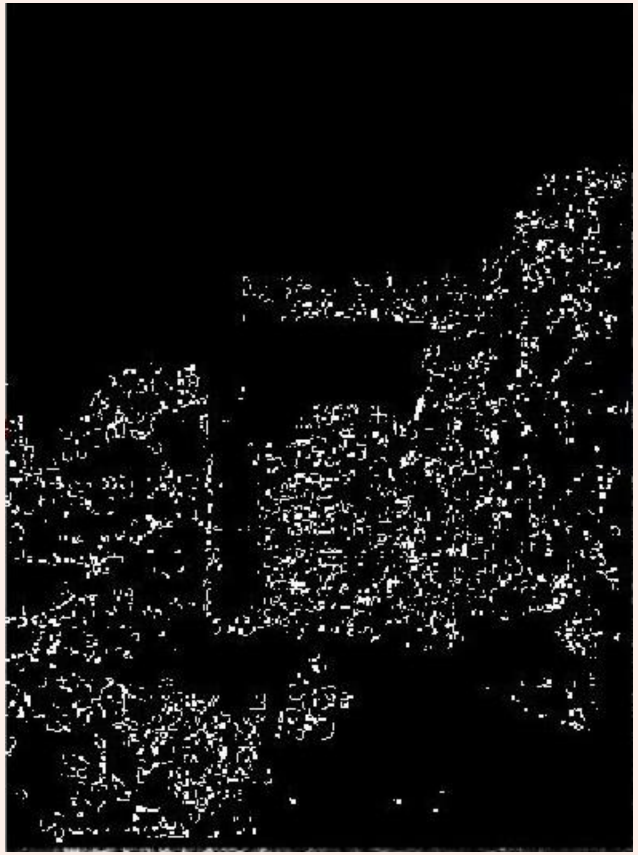
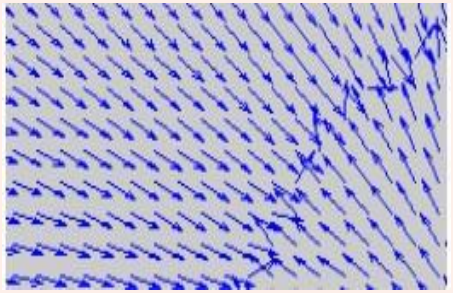
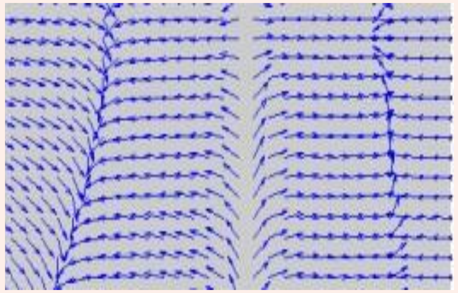

GUI for User
Image Database
Canny Image
Meta Data Table
Internet
Google Translator (Online Mode)
Reverse Geocoding
Location Database (Offline)
Text Region Extraction from Natural Scene Images using SWT + GVF+ PCA+HOG Feature
Text and Number Extraction
Trained OCR Model+Trained Translator (Offline Mode)
CSDB
LLDB
i = i +1
Translated Characters in English
Words[i]
IsPlace(Words[i])?
Yes
Geolocator Library
Location (Online)
Regional Language(Online)
Regional Language Template(Offline)
Location (Offline)
Upload to Cloud
Disk
Store locally
Display
GUI For User

IMPLEMENTATION

1. File Selection	1. The user selects the images from the folder	

1. File Selection	2. The user selects whether to run the app online or offline	 Selected Image : राजकीय प्राथमिक विद्यालय Enter Detection Mode Online OR Offline SMART INDIA HACKATHON 2018

2. Text Region Extraction:	Magnitude Gradients and Directional Coherence of selected images	 Fig 2.(a) MG of Input  Fig 2.(b) DC of Input

2. Text Region Extraction:	Histogram of Gradients of the image	 Fig 4.(a) HOG of image	 Fig 4.(b) GVF for text  Fig 4.(c) GVF for non-text

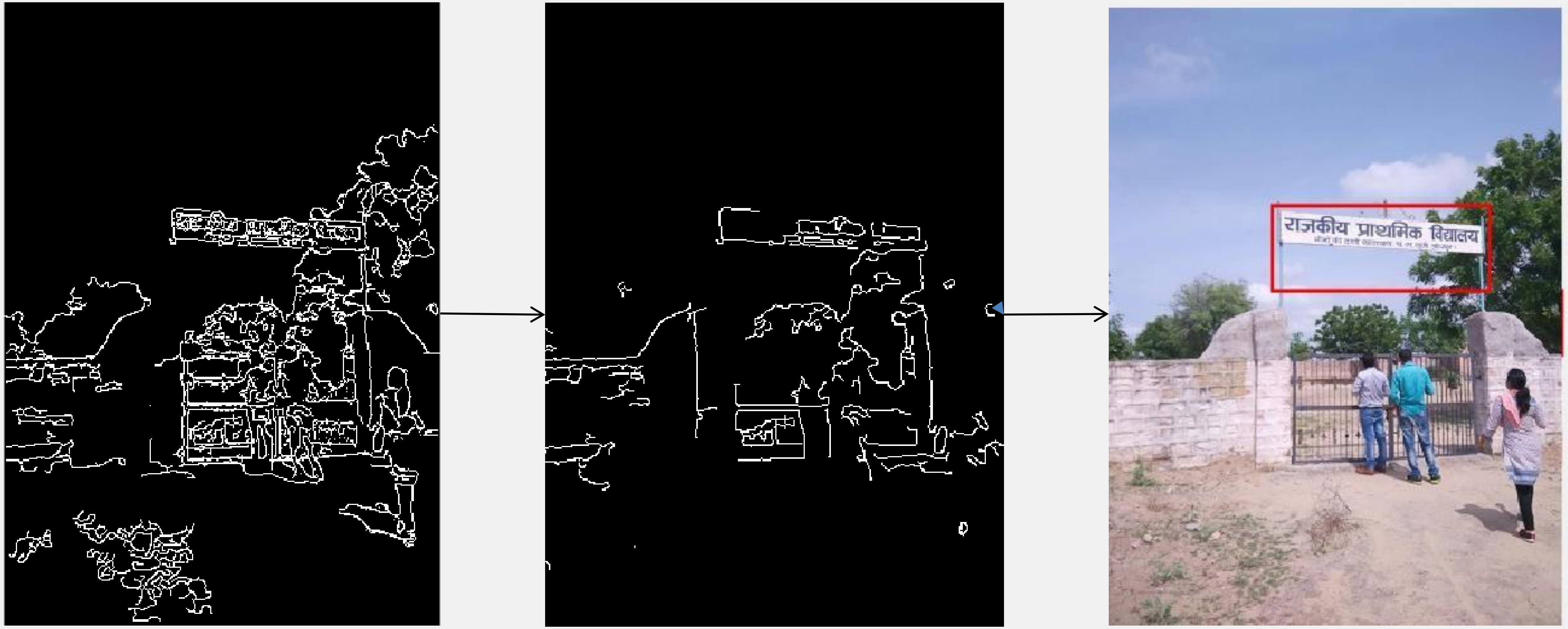
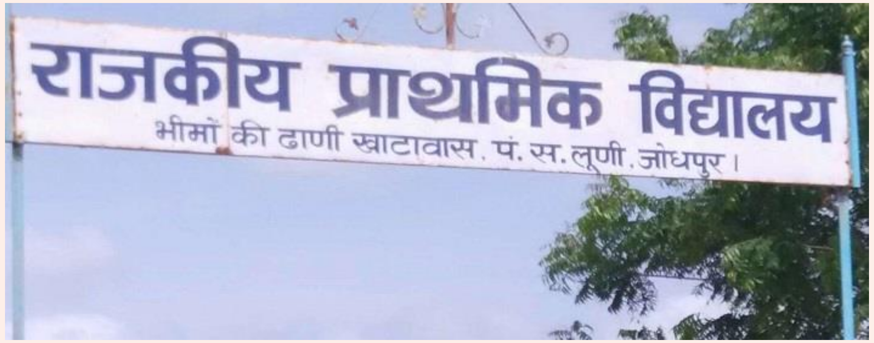
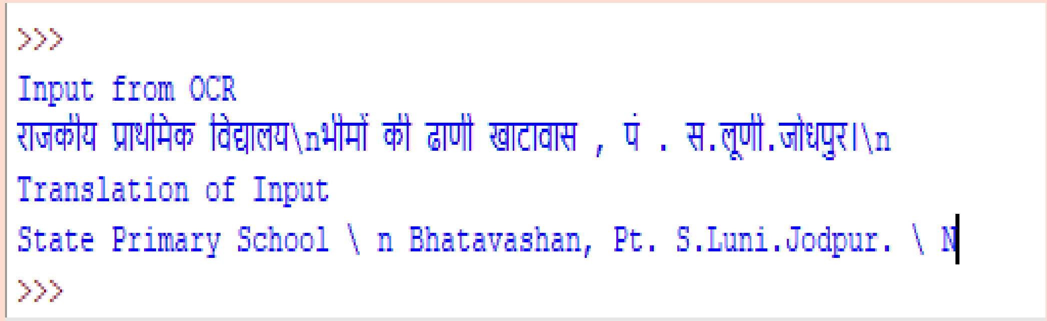


Fig. 5: Stages in detection of text region

2. Text Region Extraction	The final extracted text region from the image	 राजकीय प्राथमिक विद्यालय भीमों की ढाणी खाटावास, पं. स. लूणी, जोधपुर। Fig 6.(a) The Cropped Out ROI
3. Text Recognition and translation	OCR is run on the Region of Interest	 >>> Input from OCR राजकीय प्राथमिक विद्यालय\nभीमों की ढाणी खाटावास , पं . स.लूणी.जोधपुर।\n Translation of Input State Primary School \ n Bhatavashan, Pt. S.Luni.Jodpur. \ N >>> Fig 6.(b) Translated Output of OCR on ROI

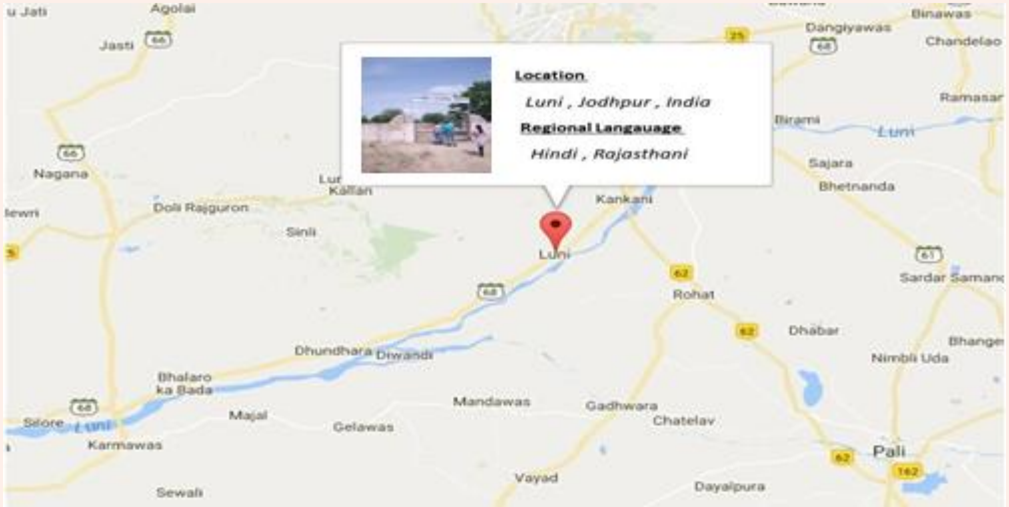
<table>
<tr>
<td>4. GeoTagging the Images on Map</td>
<td>This requires an internet connection. The detected language, and region are shown on Google maps</td>
<td>


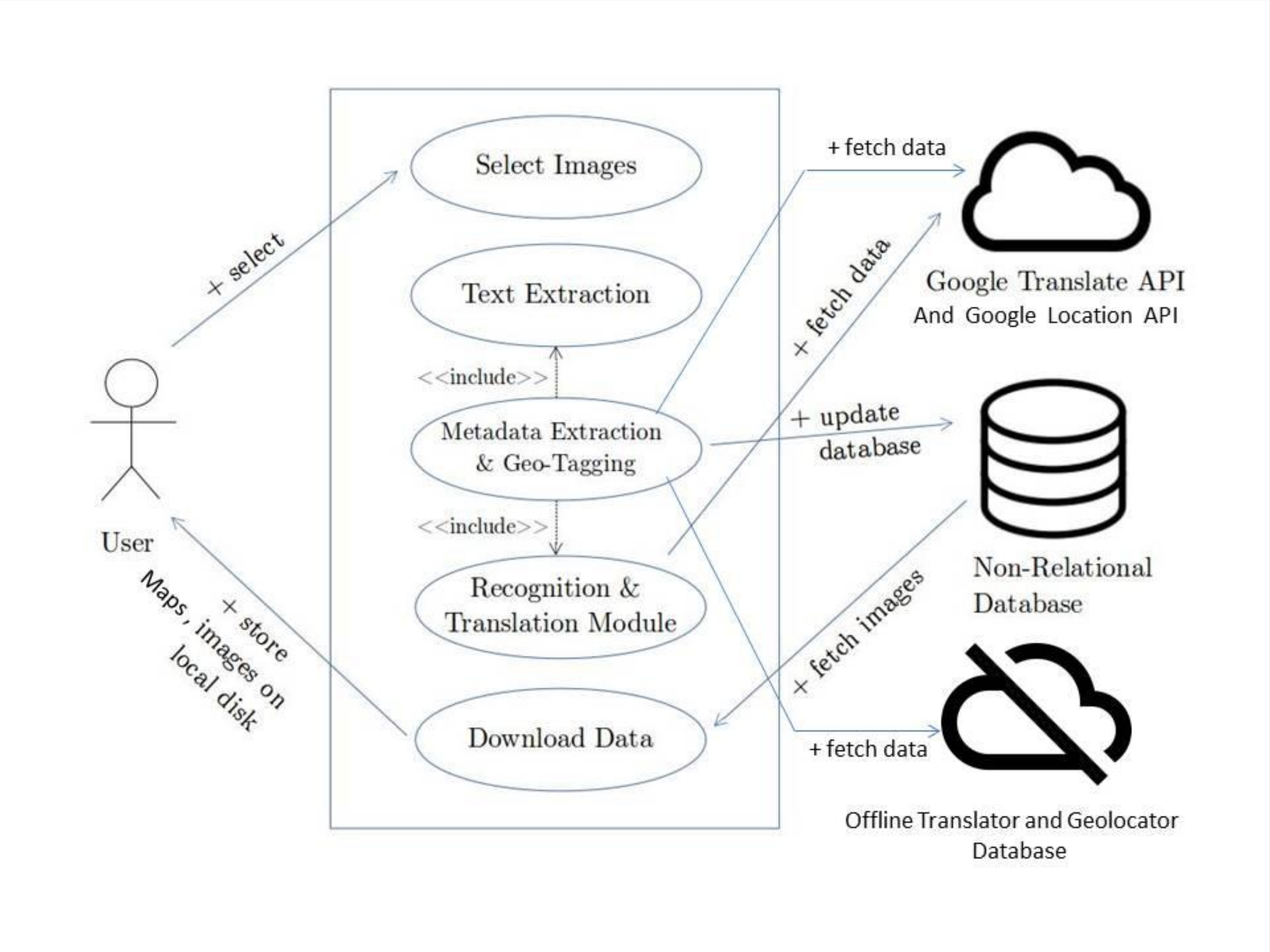
Fig 7(a). Representation of Result on map
</td>
</tr>
<tr>
<td>5. Storing in Database</td>
<td>The same information is stored in a database against the selected image</td>
<td>
<table>
<tr><th>File</th><th colspan="4">Location</th><th>Geotag</th><th>Regional Language</th></tr>
<tr><td></td><td>Lat</td><td>Long</td><td>Town</td><td>Pincode</td><td></td><td></td></tr>
<tr><td>Hindi4.jpg</td><td>26deg 0min 6sec</td><td>73deg 0min 8sec</td><td>Luni</td><td>342802</td><td>Yes</td><td>Hindi, Rajasthani</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
Fig 7(b). Representation of Result in Final database
</td>
</tr>
</table>

TECHNOLOGY STACK

- Python 3.5
- OpenCV 3.1.0
- Pillow (PIL fork)
- PyAutoGUI for building GUI
- Py2exe for making .exe file
- PyTesseract - OCR
- GeoLocator
- Tensorflow

- For relational DB: MySQL
- For non-relational DB: MongoDB
- AndroidDB : SQLite
- AndroidIDE : Android Studio

USE-CASE DIAGRAM



Select Images
Text Extraction
<<include>>
Metadata Extraction & Geo-Tagging
<<include>>
Recognition & Translation Module
Download Data
User
+ select
+ store Maps , images on local disk
+ fetch data
+ fetch data
+ update database
+ fetch images
+ fetch data
Google Translate API And Google Location API
Non-Relational Database
Offline Translator and Geolocator Database

DEPENDENCIES

- Numpy (for OpenCV 3.1.0)
- Scipy (for Tesseract OCR)
- Gradle (for Android Studio)
- Firebase(for Android Studio)
- NDK (for Android Studio)