Team **ARCSix**

Lec01 (Problems)

- ***#CSIR6*** [Anti-pilferage & Anti-adulteration system for fuel road tankers]
- ***#ISR3*** [Text-Editor]
- ***#MOD5*** [Real-Time Image Processing & Forensic Verification of Documents]

#CSIR6

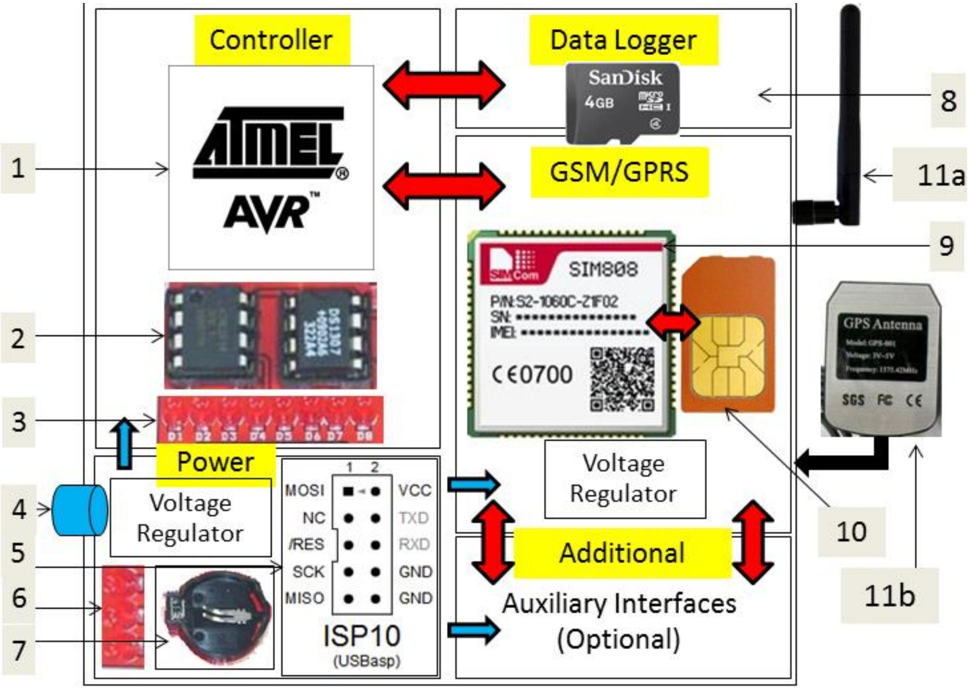
Ministry Category : Council of Scientific and Industrial Research (CSIR)
Problem Statement : Anti-pilferage & Anti-adulteration system for fuel road tankers **Problem Code :** #CSIR6
Team Leader Name : Chaitanya Tejaswi **College Code :** #2451

Prototype Description

- The user will be able to track a fleet of vehicles custom-fit with tracking hardware, in real-time, using a Web Browser plugin.
- At the end of a vehicle's journey, a customized report will be generated, describing the route taken, stops made & openings of the drain valve. This will be helpful to any organization to study the on-route behavior while making sure no pilferage/adulteration of fuel takes place.
- It consists of two modules - ***Main Board*** & ***Sensor Extension***.

Main Board consists of 5 segments:

1. Power - Supplies power to the circuitry.
   - It consists of - main & on-board supplies. Main supply would be 12 V, on-board supply would be 3V using CR032 cell (for RTC). In-system programming (ISP) can be done using USBasp.
2. Controller
   - An 8-bit AVRμc controls the operation of GSM/GPRS module & ensures consistent logging of data. Special entries will be made to the data-logger every time the main power is cutoff (indicating stoppage of vehicle). This information is also conveyed to the Server in real-time.
3. Data logger
   - It's a MicroSD card storing travel-time information as 128-bit frames. This consists of real-time co-ordinates (recorded every 5 seconds).
   - Text formatting (using descriptors) will be done for these frames to allow easy interpretation of the records, in case user needs to verify.
4. GSM/GPRS module
   - Currently SIM808 is being used to provide data-communication using GPRS, and GPS tracking using L1 frequency (1575.42 MHz) receiver.
5. Additional module(s) – the **Sensor Extension**
   - 10 pins would be drawn out for adding SPI/USART compatible modules. These will be used to interface compatible sensors for recording the OPEN duration of drain valves/lids.



Technology Stack

# Hardware

To save space, hardware modules have been referenced in the descriptors list.

# Software

for hardware programming

Atmel AVR Studio 6
GNU C-Compiler (GCC)
AVRdude (for ISP)

for server deployment

Google Maps Web Services API
Google Maps JavaScript Client API
Apache HTTP Server
Python-Django Framework
Xampp package
Notepad++ Editor

Hardware Descriptors

1. AVR 8-bit microcontroller
2. RTC & EEPROM
3. Status LEDs
4. Input Power (main) – 9V
5. ISP interface
6. Power indicators
7. CR2032 (on-board power-supply)
8. MicroSD card
9. SIM808 module
10. SIM card
11(a,b). GSM, GPS Antenna

Users

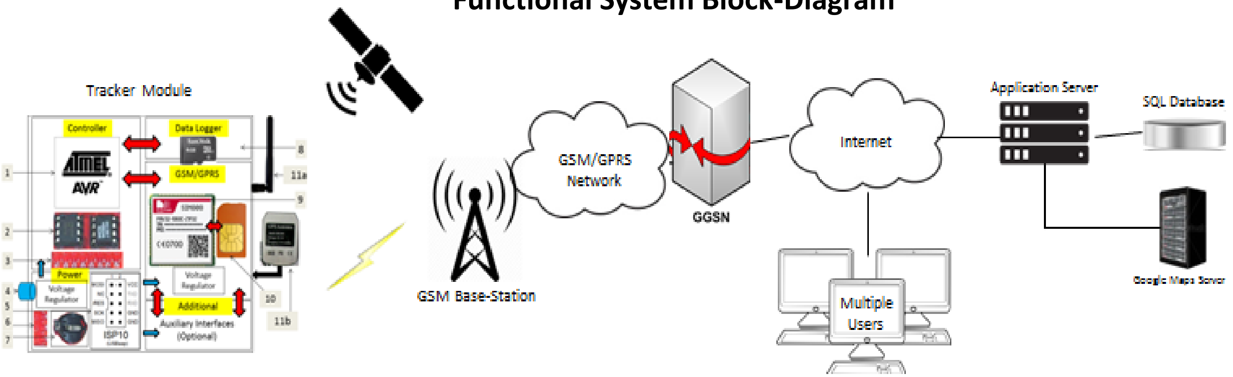
1. Maintenance operators, who will monitor the tanker's movement in real-time.
2. Operators at start/stop points, who ensure loading/unloading of fuel-tanks.
3. Truck-drivers, whose journey will be recorded by the hardware unit.

Use-Case

The travel-route information will be provided to an application server, as described before. The 'additional module(s)' will send alerts if drain valve is opened.

(See the functional-diagram to the side)

Functional System Block-Diagram



The application server will ensure proper delivery of tracking-information to the user, who will see it using his/her browser in real-time.

Dependencies

1. Active HTTP Application Server.
2. Active 2G/3G GSM Service in the area.
3. 'Additional Module(s)' will be decided, based on the mechanism of drain valve operation.

Showstoppers

1. **Google Maps API Pricing (for commercial applications)**

- Google Maps Web Services API provides a collection of 8 APIs for diverse needs. Especially useful to our solution are the Roads & Distance Matrix APIs.
- However, for commercial use, they must be licensed.

2. **Improper installation of hardware module**

- The hardware must be installed in a secure location, where it can be provided with a proper power input. A study of RF behavior near the installation will be helpful to ensure proper system behavior.

Notes

GPS-compatible modules (like SIM808) offer much more functionality than we require for this problem.

Hence, it seems useful to first implement our solution using a SIM808 development board. This allows us to quickly verify our AT-command sentences using an RS232 interface.

Once all the parameters are fixed, a SIM808 SMD-chip will be fixed onto the PCB, and can be programmed only through the microcontroller.

Future Use

Cargo Ship Tracking

Using the OpenCPN stack, interface can be provided for tracking fuel-containers on-board cargo ships.

#ISR3

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 7 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

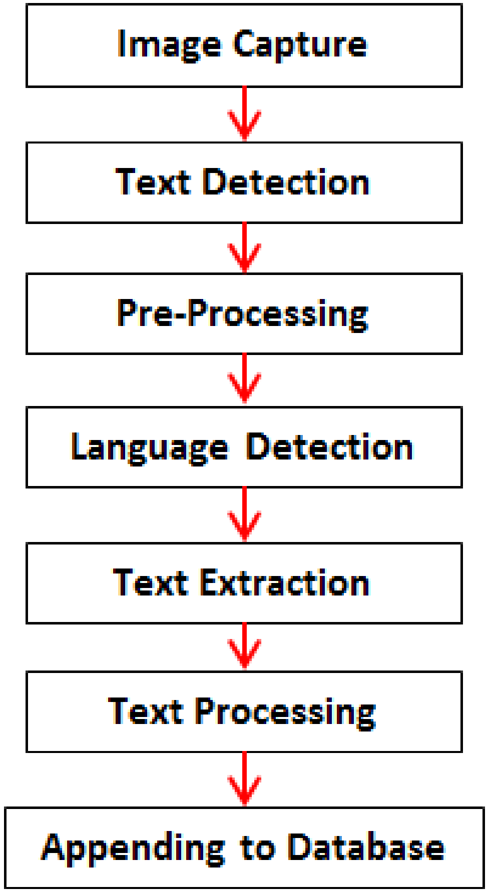


Image Capture

Text Detection

Pre-Processing

Language Detection

Text Extraction

Text Processing

Appending to Database

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.

6. 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".

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

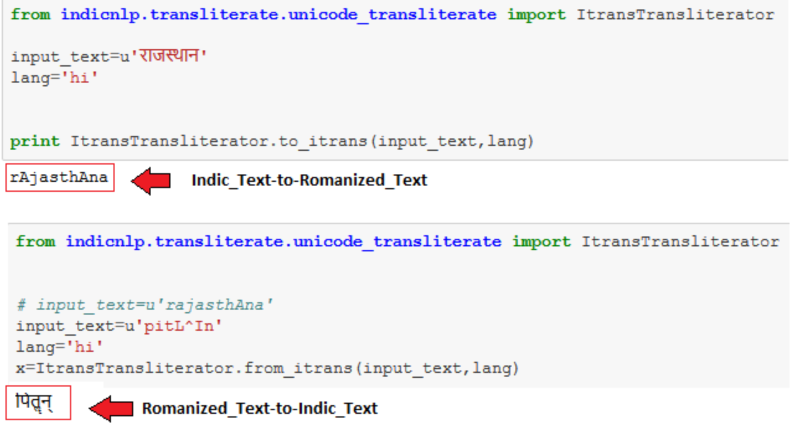
- It is assumed that the geo-tagging information is already provided to the GUI. It may be validated using the option - "Verify Co-ordinates".
- This option can be used to look up the entered location using the Google Maps API on a Web Browser.
- The option - "Submit" completes the process by adding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

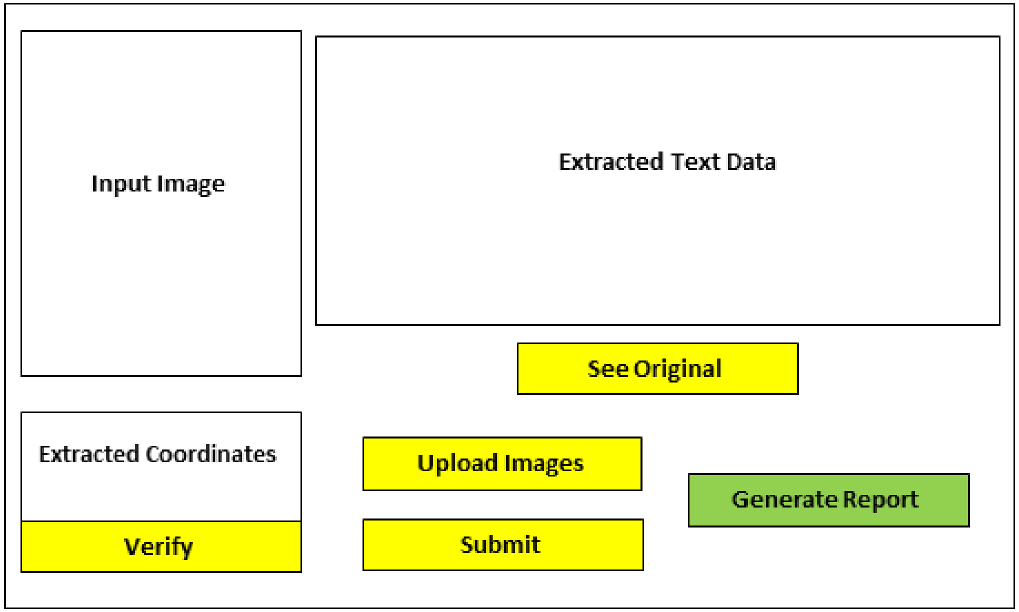
Users

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.

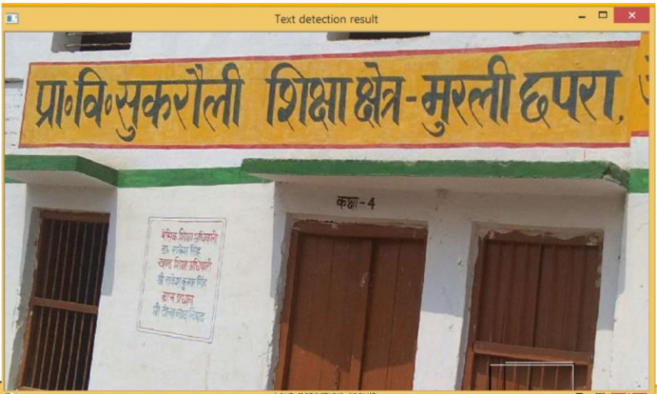
Showstoppers

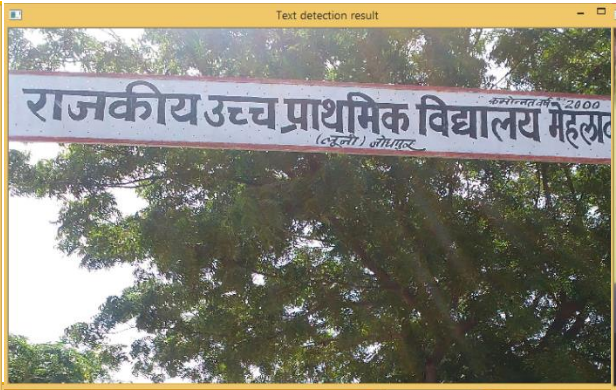
JPEG images

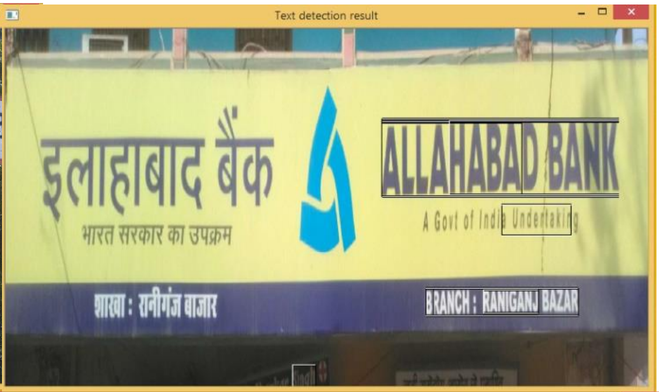
- As mentioned earlier, JPEG images are not suitable for optical character recognition. And that they are obtained from a mobile device does not help. But we do acknowledge the fact that most existing community asset images are JPEGs. Hence, we have chosen an approach that is more training dependent.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

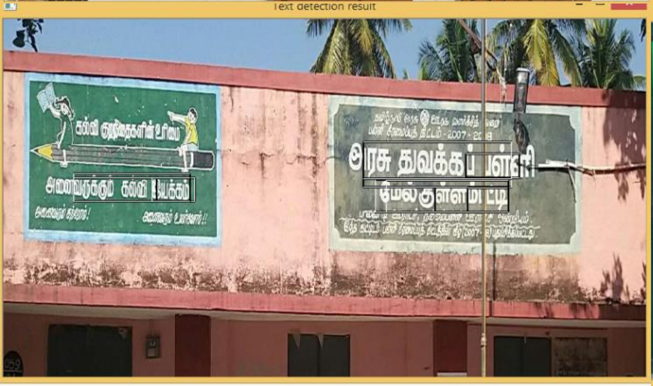
Text Detection

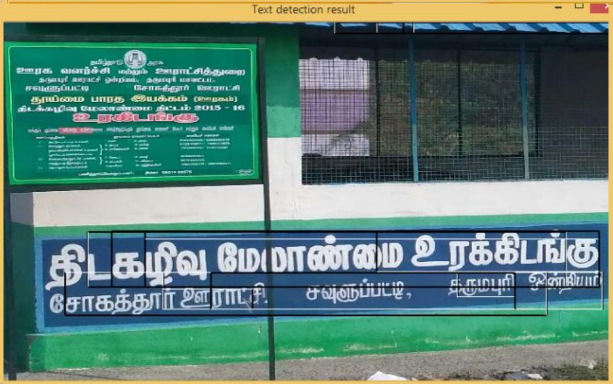
- We list 3 examples each of Hindi, Tamil & Telugu. All the images are JPEGs (as provided on the website).
- As can be seen, our current program does decent at isolating regions containing text, but character-bound regions are identified in case of English, Tamil & Telugu. Hindi text regions are not 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.

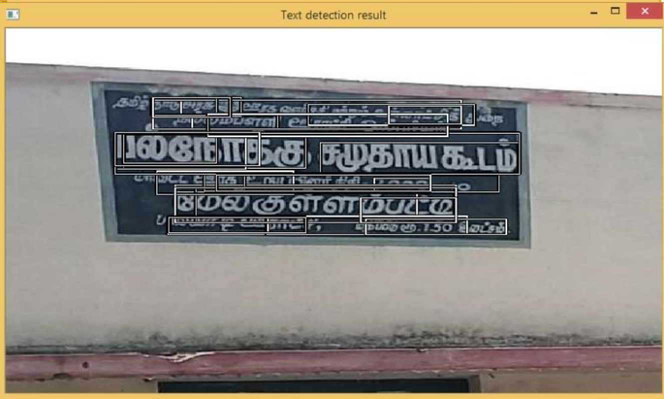


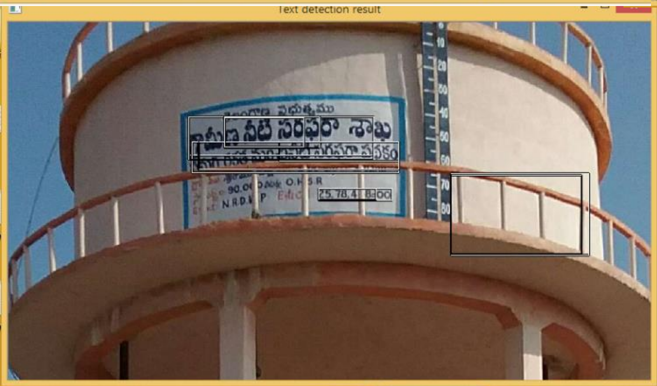


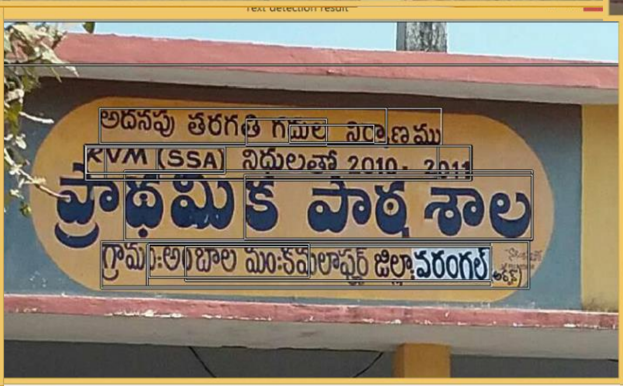


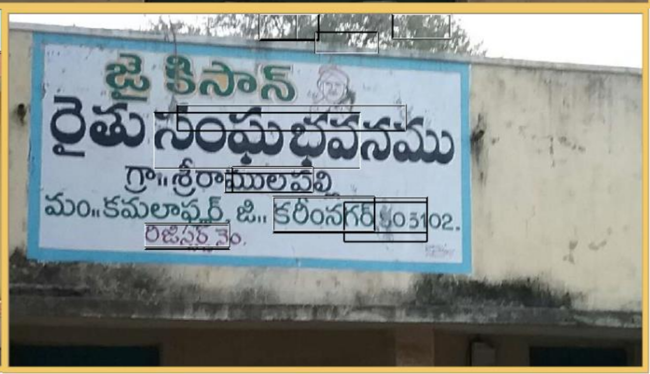












#MOD5

Ministry Category : Department of Defence Production, Ministry of Defence (MOD)
Problem Statement : Real-Time Image Processing & Forensic Verification of Documents **Problem Code :** #MOD5
Team Leader Name : Chaitanya Tejaswi **College Code :** #2451

Idea Description

The user will be able to detect & verify computer-based manipulations (if any) on a set of digitized images, using a GUI application on a PC.
In common image forgeries, such as copy paste, region duplication, image splicing, etc. , basic image operations are often involved. So, if we can find significant evidence in favour, it can be infered that the image is altered.
Our work is based on this paper. The application development will be done mainly in Python.
The 8 main stages are described below –

1. **Image Capture** - capturing the image on a flat-bed scanner.

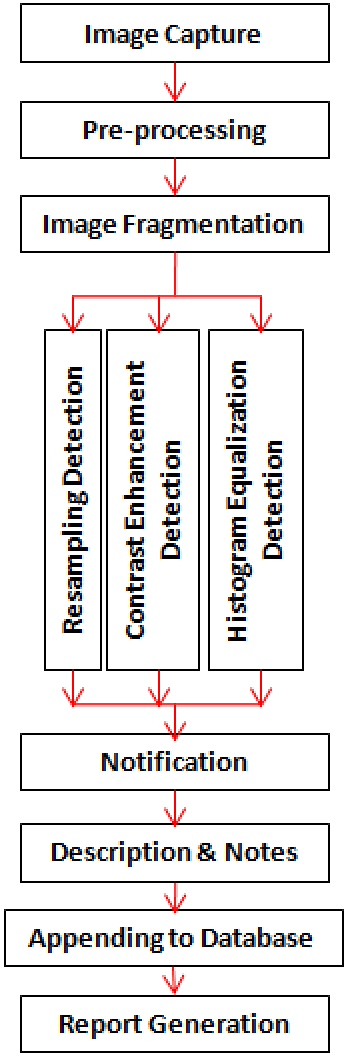
- Images obtained using a smartphone camera suffer from angular-defects, likely caused due to tilted position of the camera, when the image is taken. Compensation for this has to be provided, which results in poor performance & significant increase in code-size.
- Images obtained using a (flatbed) scanner don't need this. Also, most work in OCR technology has based on such images. So it's helpful if the images are scanned directly.
- One more reason for this is that we can choose to save the images in PNG/TIFF formats (instead of JPEG), which is useful for OCR detection. However, JPEG files are also processed.
- Also, if smartphone images are flattened using Android apps (e.g. CamScanner), the applied changes will be detected as "altered" (in later stages).
- In short, we need scanned document images.

2. Pre-processing - making the image more suitable feature detection.

- We are using the opencv-python module for cropping images, feature extraction and image enhancement of textual regions.

3. Image Fragmentation – based on the image resolution, it will be divided into 3x3 blocks. The main algorithm will be applied to each block concurrently.

Flow Diagram



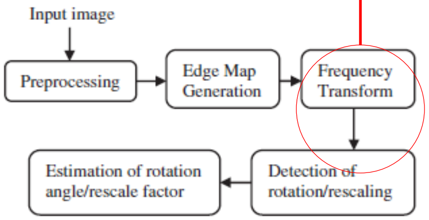
4. Main Algorithm –

This step consists of 3 main techniques:

A. *Resampling (Rotation + Rescaling) Detection*

B. *Contrast Enhancement Detection*

C. *Histogram Equalization*


Input image
Preprocessing
Edge Map Generation
Frequency Transform
Detection of rotation/rescaling
Estimation of rotation angle/rescale factor

The steps in re-sampling detection method.

In DA Method, the magnitude of DFT is calculated for each row of the edge map and then the average is taken over all the rows to get the horizontal spectrum. Assume that$E(m,n), m\in[1,M],\ n\in[1,N]$ are the entries of the edge map and F is the Discrete Fourier Transform. The DA method can be expressed as follows:

$$E_{DA} = \frac{1}{M}\sum_{m-1}^{M}|F[E(m,n)]| \qquad (1)$$

In AD Method, the average of all rows of the edge map is calculated to form a horizontal row and then the magnitude of DFT is calculated to get the horizontal frequency spectrum. The AD method can be defined as follows:

$$E_{AD} = \left|F\left[\frac{1}{M}\sum_{m-1}^{M}E(m,n)\right]\right| \qquad (2)$$

A. For ***Resampling Detection*** we need 2 frequency plots –

- DFT, then Averaging = DA Method
- Averaging, then DFT = AD Method

*DFT = Discrete Fourier Transform

The reason for having such plots is because if there are peaks only in the DA plot, a rotation has occurred. If there are peaks in both DA & AD plots, the image has been rescaled (zoomed). The rotation angle (ϴ) & rescale factor (R) are determined by equations –

$$f_{rot1} = \begin{cases} 1-\cos\Theta, & 0^{\circ} < \Theta \le 60^{\circ} \\ \cos\Theta, & 60^{\circ} < \Theta < 90^{\circ} \end{cases}$$

and

$$f_{rot2} = \begin{cases} \sin\Theta, & 0^{\circ} < \Theta \le 30^{\circ} \\ 1-\sin\Theta, & 30^{\circ} < \Theta < 90^{\circ} \end{cases}$$

$$f_{res} = \begin{cases} 1-1/R, & 1 < R \le 2 \\ 1/R, & R > 2 \end{cases}$$

or

$$f_{res} = 1/R-1,\ R < 1$$

Obviously, forged images have combined rescaling & rotation. Similar equations for such cases are also available.

B. A similar approach is taken for ***Contrast Enhancement Detection***.

- First, if it's a colour image, RGB channels are separated.
- Histogram for two of these channels is computed separately.
- Magnitude of each such histogram is calculated, and used to obtain a frequency plot.

If there are sudden peaks/zeroes in the plot, the image is said to be contrast-enhanced. Why? If images are contrast enhanced, their pixel values are increased. So is their energy content. Since Energy Frequency, the enhanced images have high frequency components.

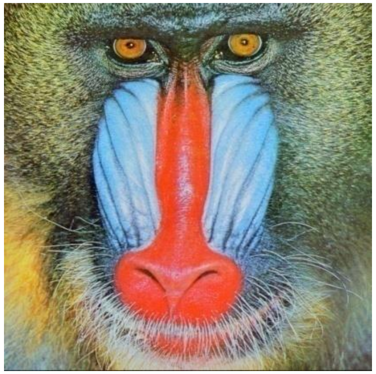
C. *Histogram Equalization*, also a form of contrast-enhancement, provides sharp peaks in frequency-plots of colour-enhanced images. If this is applied locally to the whole image by dividing into 3x3 blocks, we can estimate regions with the most alterations.

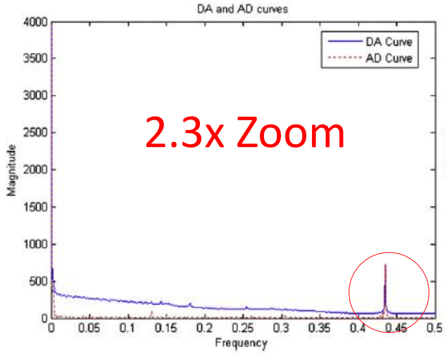
Steps **5-8** are **Notification**, **Description**, **Appending-to-Database** & **Report Generation**.

1. The user is *notified* about the detected alterations. In response, the user can add a *description* specifying what it is that has been altered.
2. The description for all the images is processed an stored into a database. A report for the same can optionally be generated.

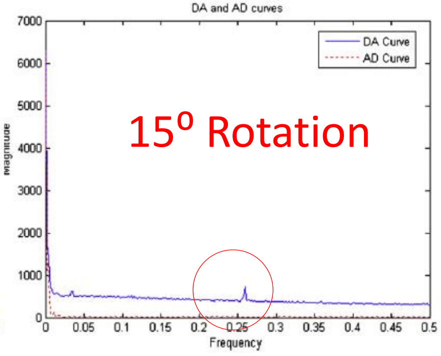
Example

DA & AD plots for Rescaled/Rotated Image





The DA and AD curves of rescaled Baboon image.



The DA and AD curves of rotated Baboon image.

Technology Stack

- PyQt4 module *(for GUI application)*
- numpy, scipy, opencv-python modules *(for image processing)*
- matplotlib module *(for graph plotting)*
- MySQL API *(for managing image database)*
- Notepad++ Editor

Users

Office Staff, who will upload the images for forensic verification.

Use-Case

- The User will specify the set of images, using "Upload Images". (See the GUI application template to the side)
- After a while, the "Previous/Next Image" button become active.
- Next, the "Issues" section starts listing all alteration-detected fragments for the current image. The "Check" button, which now becomes active, allows the user to verify these alterations
- Using the "Notes" section, the user can add details about the detected alterations.
- Once all the images have been processed, the user will click "Submit". All detected information will be appended to database.
- Optionally, a report can be generated for the session.

Dependencies

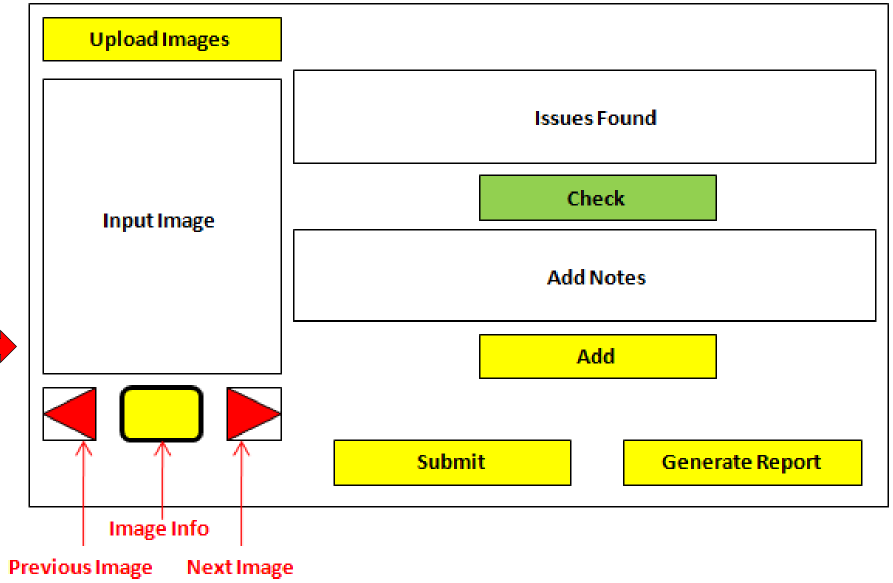
- Most of our application would be in Python, and the primary system dependency for it would be a Windows 7 system, with at least 1 GB of RAM.

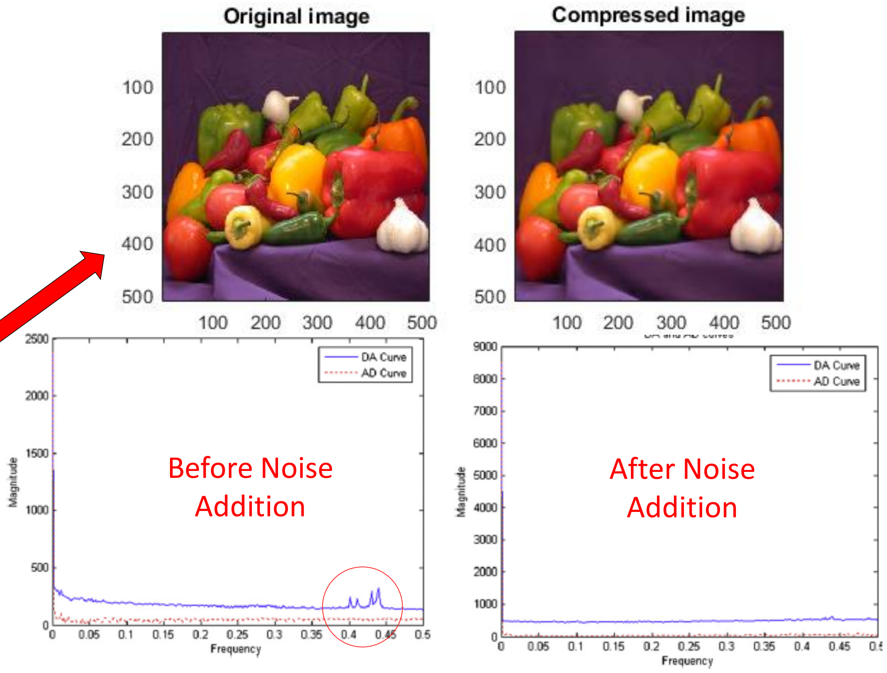
Showstoppers

JPEG images

- The proposed technique indicates the presence of re-sampled image regions in an image. However, it is susceptible to "JPEG attacks" – periodic JPEG blocking artifacts coinciding with periodic patterns introduced by resampling.
- This is suppressed by adding Gaussian noise to the JPEG image.

User Application Window

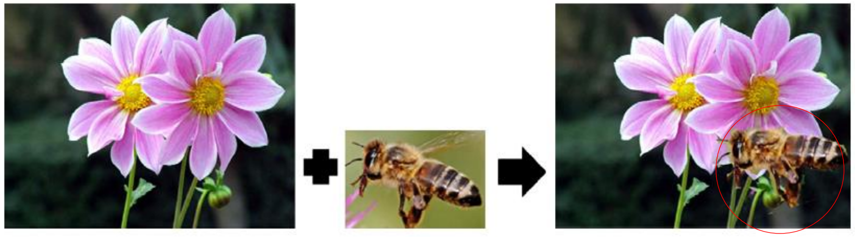




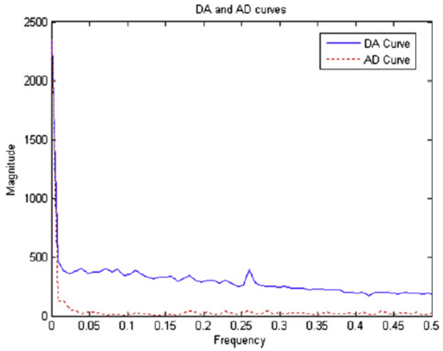
Before Noise Addition

After Noise Addition

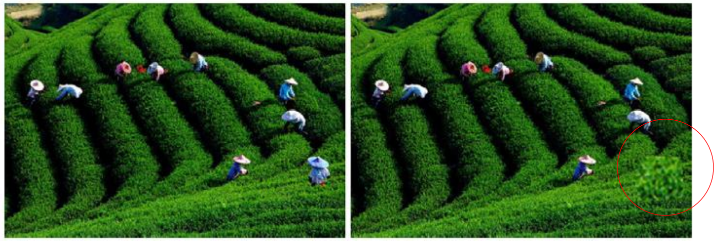
Examples



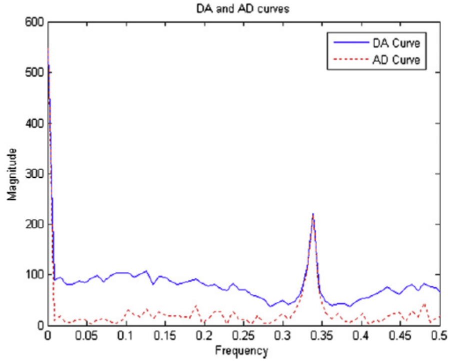
The original flower and bee images (first and second) and the forged image (third) obtained by rotating the bee image by 15° and pasting it to the flower image.



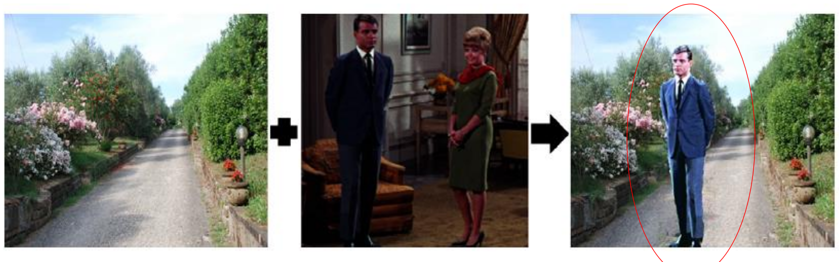
The DA and AD curves of one of the fake image blocks of the forged image. A peak appears at 0.26 in DA method because of rotation by 15°.



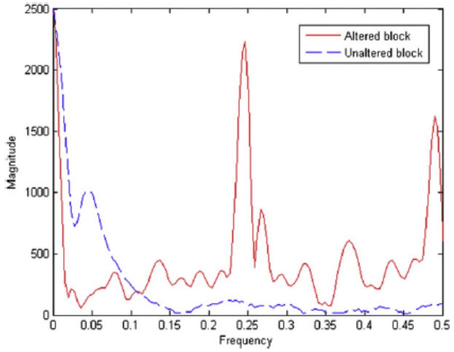
The original image (left) and the forged image (right) obtained by copying and rescaling an original image block and pasting it within that image.



The DA and the AD curves of one of fake image blocks of the forged image. A peak appears at 0.33 in both methods because of rescaling by a factor of 3.



The two original images (first and second) used for creating the forged image (third) which is an example for local histogram equalization.



The frequency spectrum plots of unaltered block and one of the altered blocks of the forged image. A striking peak appears in the plot of altered image block at 0.25.