A REPORT

ON

Electricity meter readings extraction using Video Processing

BY

Name(s) of the Student(s)	ID.No.(s)
Mayur Dhwaj Singh	**2016B3A30543P**
Priyanka Verma	**2016B3A30492P**
Malaika Rastogi	**2016B1A70926P**

AT

Bhaskaracharya Institute for Space Applications and Geo-informatics, Gandhinagar

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE,PILANI

(May 2018 to July 2018)

A REPORT

ON

Electricity meter readings extraction using Video Processing

BY

Name(s) of the Student(s)	ID.No.(s)	Discipline(s)
Mayur Dhwaj Singh.	**2016B3A30543P**	**Msc. Economics + BE. Electrical and Electronics Engineering**
Priyanka Verma	**2016B3A70492P**	**Msc. Economics + BE. Computer Science**
Malaika Rastogi	**2016B1A70926P**	**Msc. Biological Sciences +BE. Computer Science**

Prepared in partial fulfilment of the

Practice School-I Course No.

BITS F221/BITS F231/BITS F241

AT

Bhaskaracharya Institute for Space Applications and Geo-informatics, Gandhinagar

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE,PILANI

(May 2018 to July 2018)

ACKNOWLEDGMENT

We are grateful to **T.P.Singh**, **Director (BISAG)** for giving us this opportunity to work the guidance of renowned people of the field of MIS Based Portal also providing us with the required resources in the company.

We would like to express our endless thanks to our project guide **Manoj Pandya**, project coordinator Mr. Viraj Choski, Admin Staff **Mr.Saurabh Bhabhor** And **Mr. Siddharth Patel** at Bhaskaracharya Institute of Space Application and Geo-informatics for their sincere and dedicated guidance throughout the project development.

We would like to extend out special gratitude to **Mr. Rohit Patel** and **Mr. Mohein** of **Gujarat Power Research and Development** for their constant support, guidance, provision of resources and state of the art facilities in **IIT Gandhinagar**.

Also our hearty gratitude to our Practice School Instructor **Dr.Pratik Sheth** and Co-instructor **Mr. Anuj Dixit** , for encouraging, monitoring and supporting us in the project.

Malaika Rastogi

Student ID: 2016B1A70926P

Mayur Dhwaj Singh

Student ID: 2016B3A30543P

Priyanka Verma

Student ID: 2016B3A70492P

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

Practice School Division

Station: Bhaskaracharya Institute for Space Applications and Geo-informatics (BISAG), Gandhinagar

Center: BISAG, Gandhinagar

Duration: From: 22 July 2018 To: 13 May 2018

Date of Submission: 13 May 2018

Title of the Project: Electricity meter readings extraction using Video Processing

ID No.	Name(s) of student(s)	Discipline
2016B1A70926P	Malaika Rastogi	Msc. Biological Sciences & BE.Computer Science
2016B3A30492P	Priyanka Verma	Msc. Economics & BE.Computer Science
2016B3A30543P	Mayur Dhwaj Singh	Msc. Economics & BE. Electrical and Electronics

Name(s) of expert	Designation
Mr. Manoj Pandya	Project Scientist
Mr. Viraj Choksi	Project Coordinator

Name of the PS Faculty	
Dr.Pratik Sheth	PS Instructor
Mr. Anuj Dixit	PS Co-instructor

Key Words: Video Processing, OpenCV, Python, Electricity Meter

Project Area(s): Research and Development, application of IT in Power Sector

Abstract:

This project attempts to develop a system for processing videos to digitally obtain the readings of electricity meter. In this system, a pre-recorded video from mobile camera is used to read the units digitally- V, kWh, KW, A. This video is preprocessed, which ends up with cropped digital reading area using contour detection. It is followed by feature extraction of source image templates and the video. The features are then matched using brute force algorithm . The code runs in real time and displays the detected text on the video screen. The required frames are extracted from the video and stored as an image in .png format. The proposed system is built in Python using OpenCV libraries. It was tested on electricity meters and results show great accuracy and efficiency upto a distance of 1ft. The proposed system is being planned to be incorporate in a mobile application that could be used by the electricity companies to facilitate the electricity billing process.The scope of the mobile app can be further extended by incorporating Geolocation software and QR code so that the user takes video from his own registered electricity meter.

Priyanka

[illegible]

Signature(s) of Student(s)

Date: 13 May 2018

Signature of PS Faculty

Date: 13 May 2018

TABLE OF CONTENTS

Sn.No.	Content	Pg.No.
1.	Introduction 1.1 Project details 1.2 Purpose 1.3 Scope 1.4 Objectives 1.5 Literature and History	1
2.	Basic Concepts involved and technologies used 2.1 Basic Concepts 2.2 Technologies used	2-4
3.	Project Management 3.1 Libraries Requirements 3.2 System Analysis 3.3 Project Plan 3.4 Project Scheduling	5-6
4.	System Design	7-8
5.	Implementation Planning 5.1 Implementation Environment 5.2 Coding Standard	9-18

6.	System Testing and Results	19-22
7.	Limitations and Future Scope 7.1 Limitations 7.2 Future Scope	23
8.	Conclusion and References 8.1 Conclusion 8.2 Bibliography	24
9.	Appendix	25

1. INTRODUCTION

1.1 PROJECT DETAILS-:

The project “Extraction of electricity meter readings using Video processing” is an attempt to capture a live video and obtain efficiently and accurately readings of the electricity digital meter.

1.2 PURPOSE-:

The main purpose of this project is to save time, reduce manpower and prevent errors caused by human in the process of taking readings from electricity meters.

1.3 SCOPE -:

An application will be developed through which user can capture the live video which will be processed and bill will be generated at the same time. Online payments can also be added to it for fast payments.

1.4 OBJECTIVES-:

- To digitally extract the readings from electricity meter.
- To increase the accuracy of detecting the units and digits.
- To take the image of the screen when the reading is detected and save it in a file.

1.5 LITERATURE AND HISTORY-:

Earlier the process to take the meter readings and bill generation involved a company official to make home visit, note down the reading then a bill is generated at the office and couriered to the home. In order to solve this an application was planned to be developed by Gujarat Power Research and Development(GPRD) and a student of IIT Gandhinagar. Later, BISAG collaborated to proceed on the project.

2. BASIC CONCEPTS INVOLVED AND TECHNOLOGIES USED

2.1 BASIC CONCEPTS

A. **Open source-:** Open source refers to any program whose source code is made available for use or modification as users or other developers see fit.

B. **Digital Image Processing**-: Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it.

C. **Video Processing-:** Video processing is a method to perform some operations on a video, in order to modify the video , its appearance or to extract some useful information from it.

D. **Mask operations-:** The idea is that we recalculate each pixels value in an image according to a mask matrix (also known as kernel). This mask holds values that will adjust how much influence neighboring pixels (and the current pixel) have on the new pixel value i.e. we make a weighted average, with our specified values.

E. **Blur operations-:** Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noises. It actually removes high frequency content (eg: noise, edges) from the image. So edges are blurred a little bit in this operation.

F. **Thresholding-** : Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images.

G. **Color Filtering-:** This modifies a given image in order to keep a specific hue (given too) and to desaturate the rest of the image. This procedure originates an image with black and white colormap, excluding the parts colored with that hue.

H. **Template-:** Areas of an image that are similar to a patch forms the template.

I. **Template Matching-:** Template Matching is a method for searching and finding the location of a template image in a larger image. It simply slides the template image over the input image (as in 2D convolution) and compares the template and patch of input image under the template image.

J. **Feature-:** A feature is a piece of information which is relevant for solving the computational task related to a certain application. Features may be specific structures in the image such as points, edges or objects. Features may also be the result of a general neighborhood operation or feature detection applied to the image.

K. **Feature Matching-** Feature matching means finding corresponding features from two similar datasets based on a search distance. One of the datasets is named source and the other target, especially when the feature matching is used to derive rubber sheet links or to transfer attributes from source to target data.

2.2 Technologies used

A. **Anaconda Cloud-**: Anaconda is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment.

B. **Pip** -: pip is a package management system used to install and manage software packages written in Python. Many packages can be found in the default source for packages and their dependencies — Python Package Index (PyPI). ... pip is a recursive acronym that can stand for either "Pip Installs Packages" or "Pip Installs Python".

C. **Python**-: Python is an interpreted high-level programming language for general-purpose programming.Python is a scripting language like PHP, Perl, Ruby and so much more. It can be used for web programming (django, Zope, Google App Engine, and much more). But it also can be used for desktop applications (Blender 3D, or even for gamespy game). Python can also be translated into binary code like java.Python is used in GIS programming. It is used as a scripting language for ArcGIS, and for Quantum GIS.

D. **OpenCV Library**-: OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications.

3. PROJECT MANAGEMENT

3.1 LIBRARIES REQUIREMENTS-:

a) **numPy** -: A general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

b) **imutils** -: A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.

c) **os -:** The main purpose of the OS module is to interact with your operating system. The primary use I find for it is to create folders, remove folders, move folders, and sometimes change the working directory. You can also access the names of files within a file path.

3.2 SYSTEM ANALYSIS-

• **Earlier System -** Earlier the process to take the meter readings and bill generation involved a company official to make home visit, note down the reading then a bill is generated at the office and couriered to the home. Also on spot billing and photo billing system were present in which user takes the image of the meter and whatsapp- it to the company.

• **Problem Identification –** This is time and manpower consuming, large expenses and involves lot of human errors.

• **Feasibility Study-** By processing a live captured video , readings can digitally be extracted and saved in the database of server. Bill will be generated by calculating the total units consumed as the previous data is also stored on the server. Hence, it will be given to the user at the same time. And as we are working on video instead of images there are least chances of manipulation.

3.3 PROJECT PLAN-

A. Getting familiarized and Installing pip, python 3 and OpenCV library.
B. Learning video processing using OpenCV tutorials.
C. Installation of Anaconda Cloud in virtual machine.

D. Retrieving of sample videos from GPRD (Govt. Power Research and Development)
E. Pre-processing of video
F. Reading of units- Kwh, V, A, Kw
G. Storing and printing of the units associated with it simultaneously.
H. Documentation of the project.

3.4 Project Scheduling

Task Name	Start date	End date	Duration
1.Environment setup	**5/6/18**	**7/6/18**	**4days**
2.Learning Python and OpenCV	**7/6/18**	**13/6/18**	**7 days**
2.System Analysis	**14/6/18**	**16/6/18**	**3days**
3.System Implementation	**16/6/18**	**30/6/18**	**15 days**

4.Testing and Improvements	**1/7/18**	**8/7/18**	**7 days**
5. Final Documentation	**8/7/18**	**10/7/18**	**3 days**

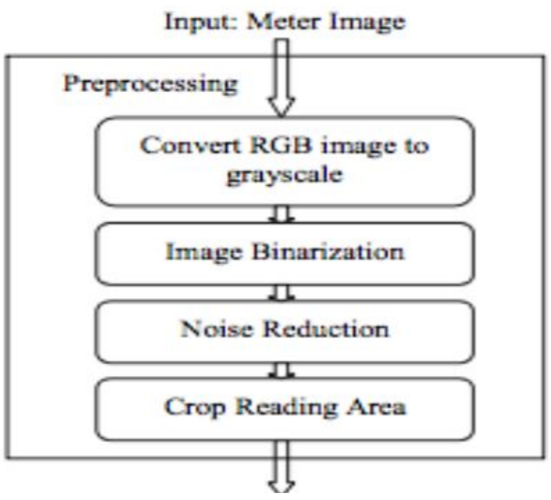
4. SYSTEM DESIGN

After the video is captured then following major three processes will take place on it:

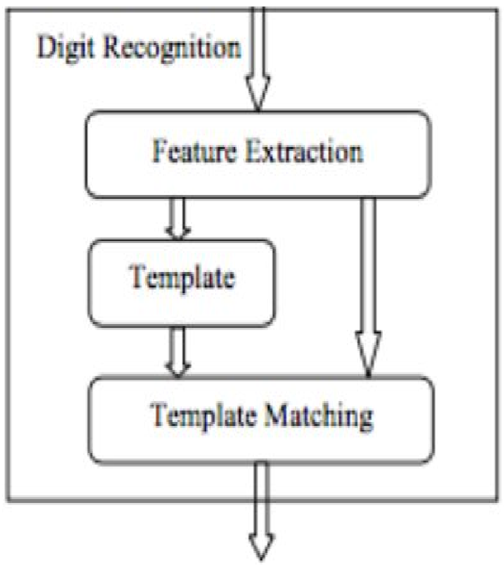
1. **Pre-processing**
    i. cropping the numeric reading area
    ii. color filtering

2. **Units reading**
    i. Template matching
    ii. Feature Matching
    iii. Homography

PROPOSED SYSTEM FLOWCHART



Individual Unit Segmentation

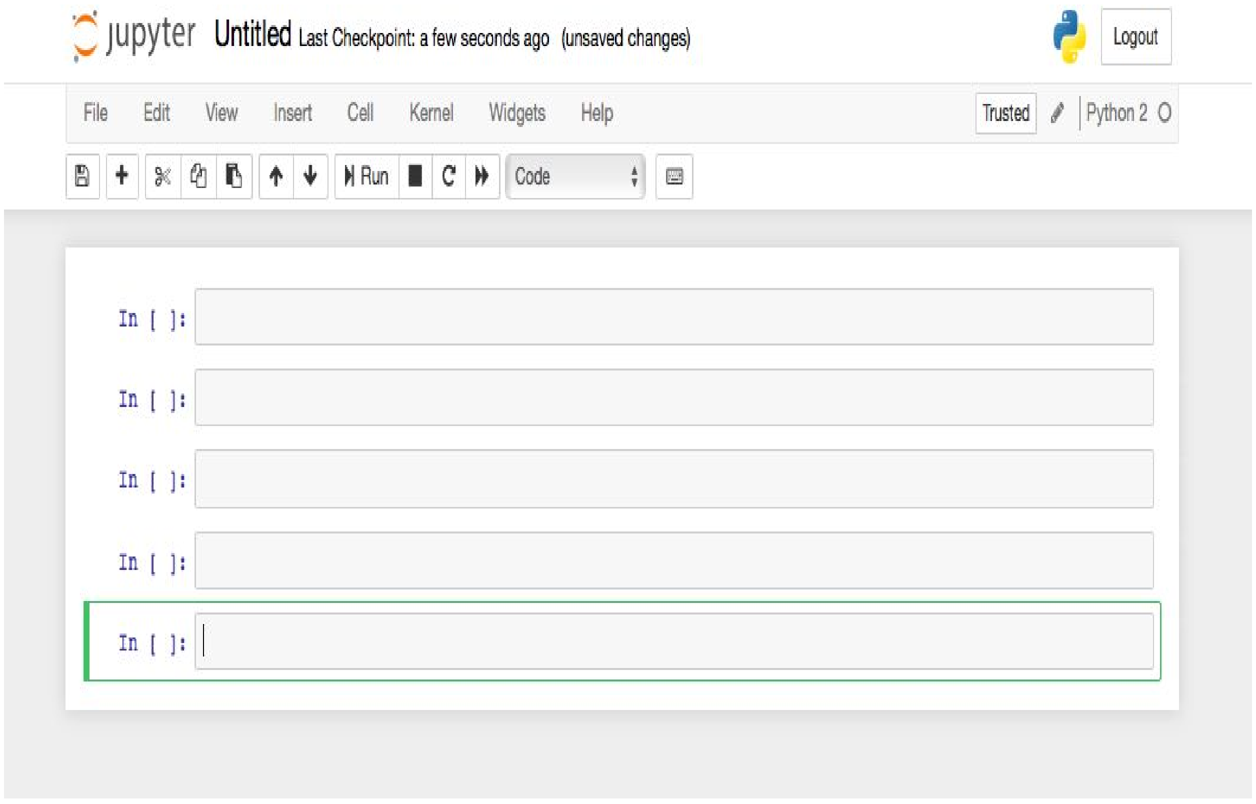


5. IMPLEMENTATION PLANNING

5.1 Implementation Environment-:

Project was initiated on SciLab-5.5.2. We learnt its installing and working using tutorials but many of the toolboxes weren't compatible with the version of our systems. So this environment was dropped.

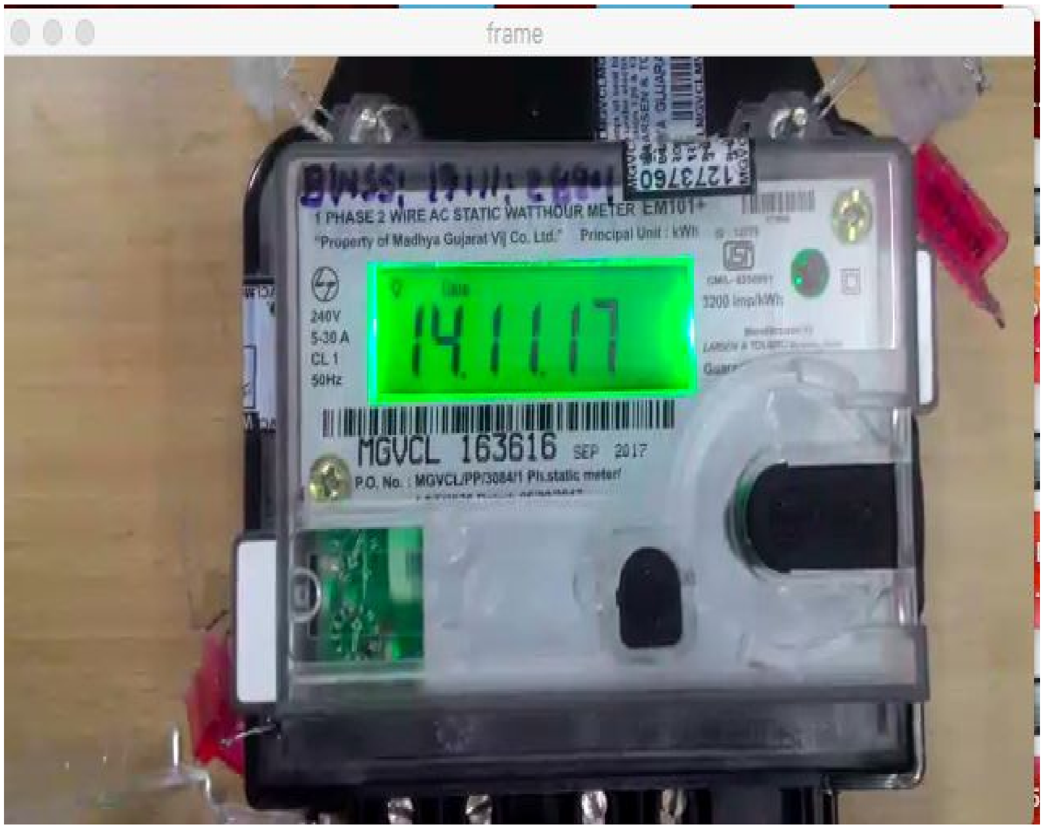
Finally, Jupyter Notebook 5.4.1 launched via Anaconda Cloud is used as IDE (Integrated Development Environment) for working on python 3. The screen looks as shown below.



```
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
```

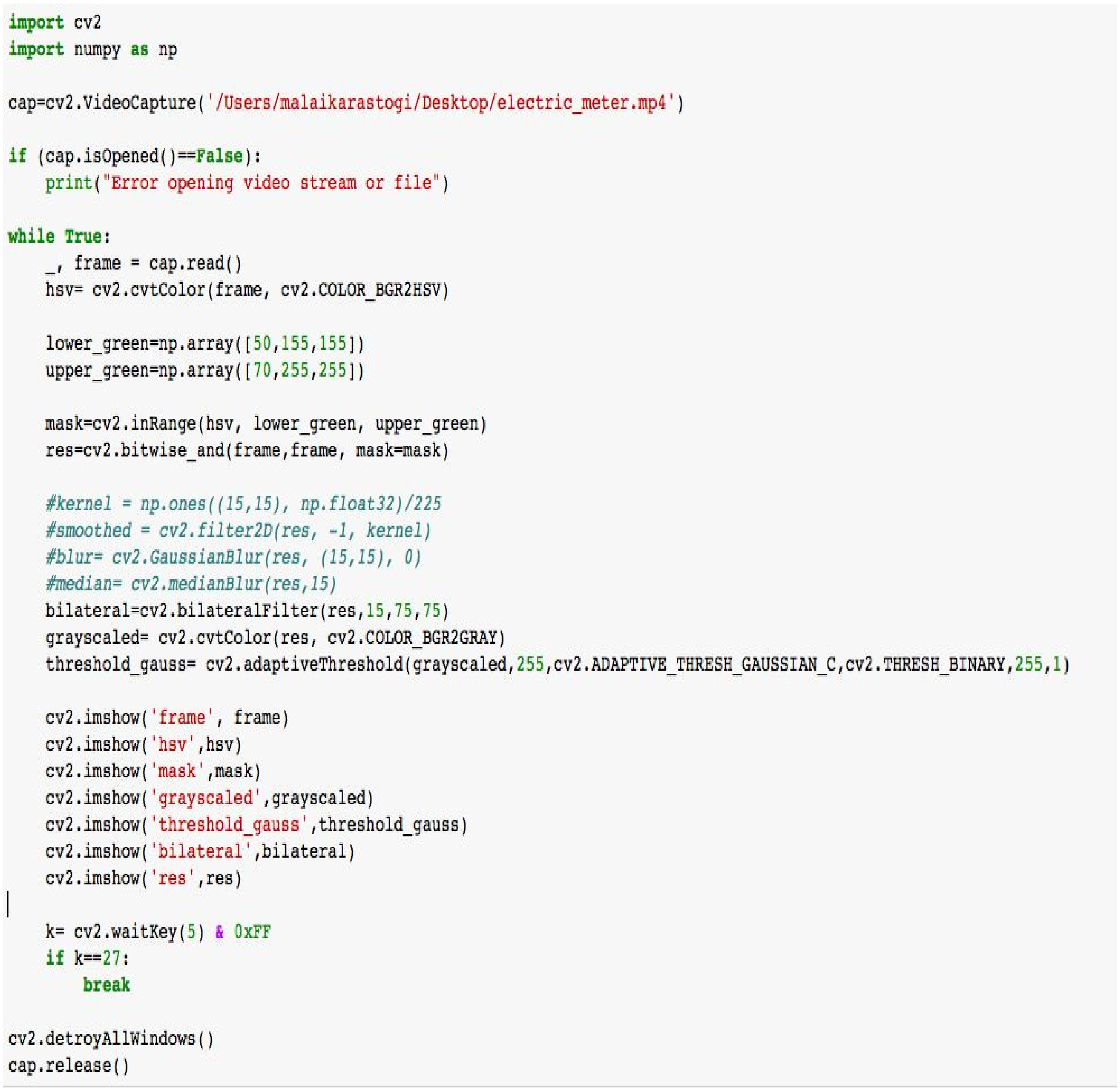
Original meter:

A screenshot of the electric meter from its video is shown below.



5.2 Coding Standard-: Code and its explanation.

1) <u>**PRE-PROCESSING STAGE**</u>
    a) **Attempt 1 (using color filtering only)**

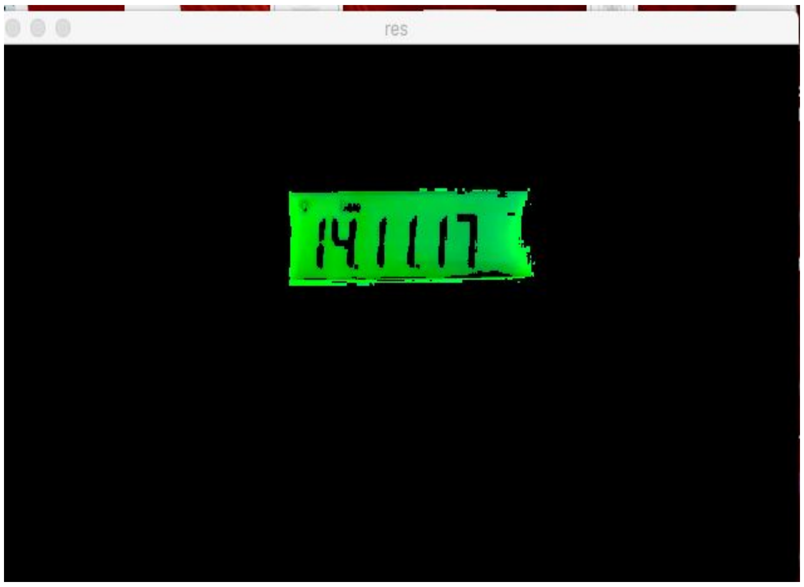


```
import cv2
import numpy as np

cap=cv2.VideoCapture('/Users/malaikarastogi/Desktop/electric_meter.mp4')

if (cap.isOpened()==False):
    print("Error opening video stream or file")

while True:
    _, frame = cap.read()
    hsv= cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower_green=np.array([50,155,155])
    upper_green=np.array([70,255,255])

    mask=cv2.inRange(hsv, lower_green, upper_green)
    res=cv2.bitwise_and(frame,frame, mask=mask)

    #kernel = np.ones((15,15), np.float32)/225
    #smoothed = cv2.filter2D(res, -1, kernel)
    #blur= cv2.GaussianBlur(res, (15,15), 0)
    #median= cv2.medianBlur(res,15)
    bilateral=cv2.bilateralFilter(res,15,75,75)
    grayscaled= cv2.cvtColor(res, cv2.COLOR_BGR2GRAY)
    threshold_gauss= cv2.adaptiveThreshold(grayscaled,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,255,1)

    cv2.imshow('frame', frame)
    cv2.imshow('hsv',hsv)
    cv2.imshow('mask',mask)
    cv2.imshow('grayscaled',grayscaled)
    cv2.imshow('threshold_gauss',threshold_gauss)
    cv2.imshow('bilateral',bilateral)
    cv2.imshow('res',res)

    k= cv2.waitKey(5) & 0xFF
    if k==27:
        break

cv2.detroyAllWindows()
cap.release()
```

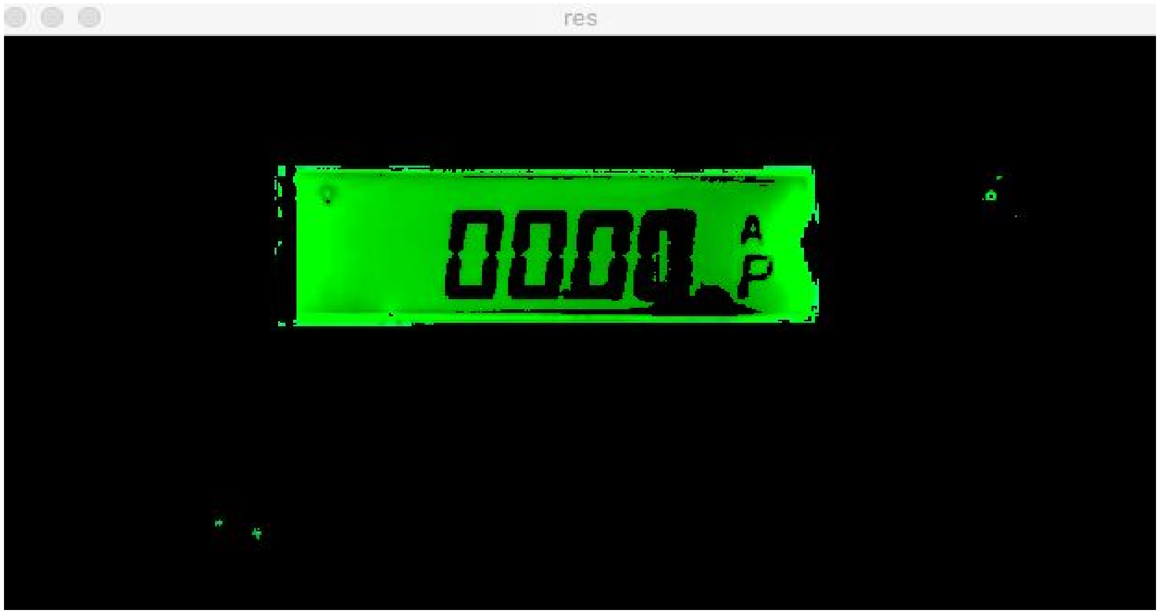
Explanation:

Code works in following way-

BGR -> HSV ->Color Filtering -> Mask -> Blurring -> Threshold -> Cropped out reading area -> Result



Problems faced with this attempt

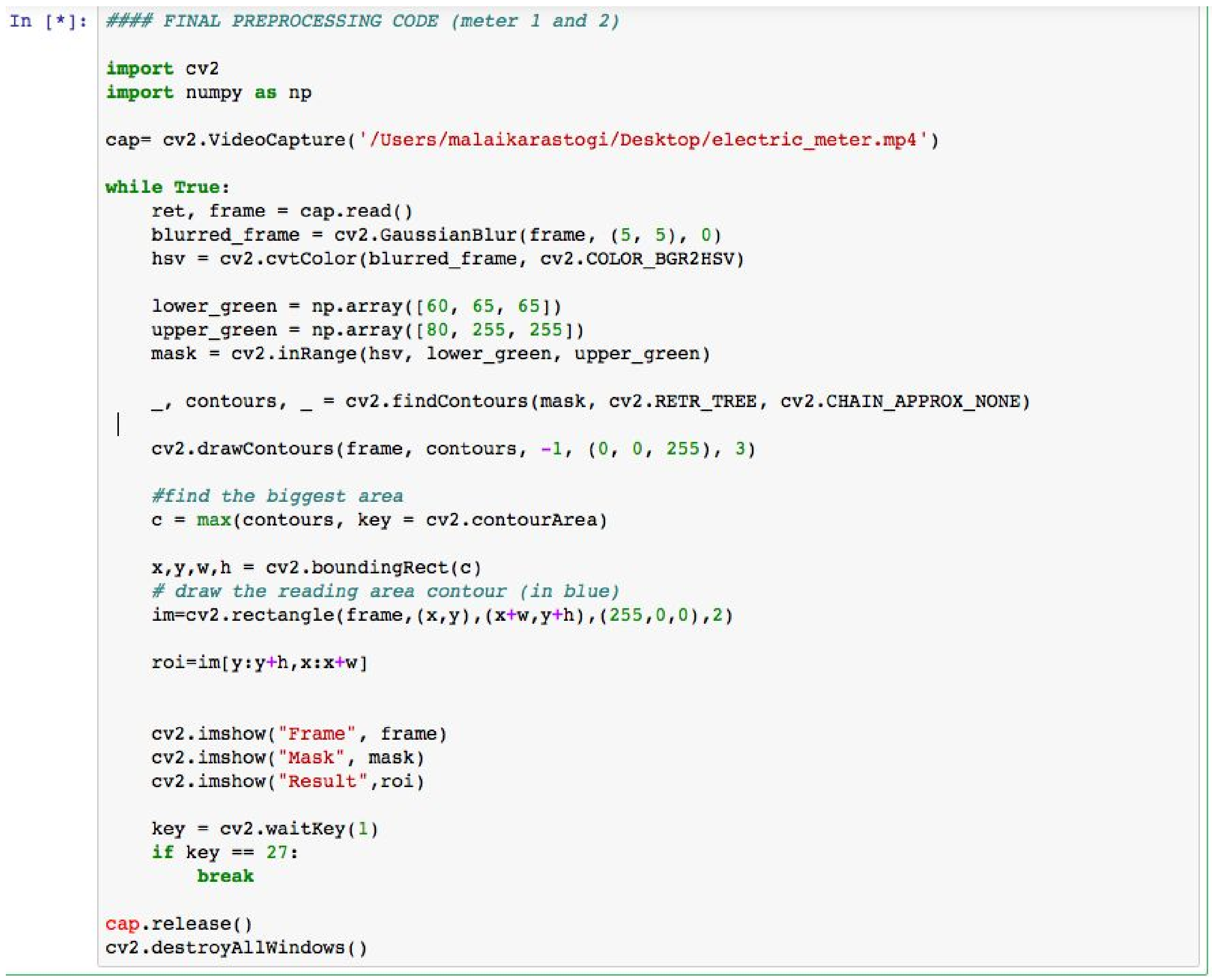


Various other methods can be required to use for noise reduction and thresholding. This is because using this we were not able read the digits and units properly and clearly.

b) **Attempt 2 (using Contour detection)**

Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

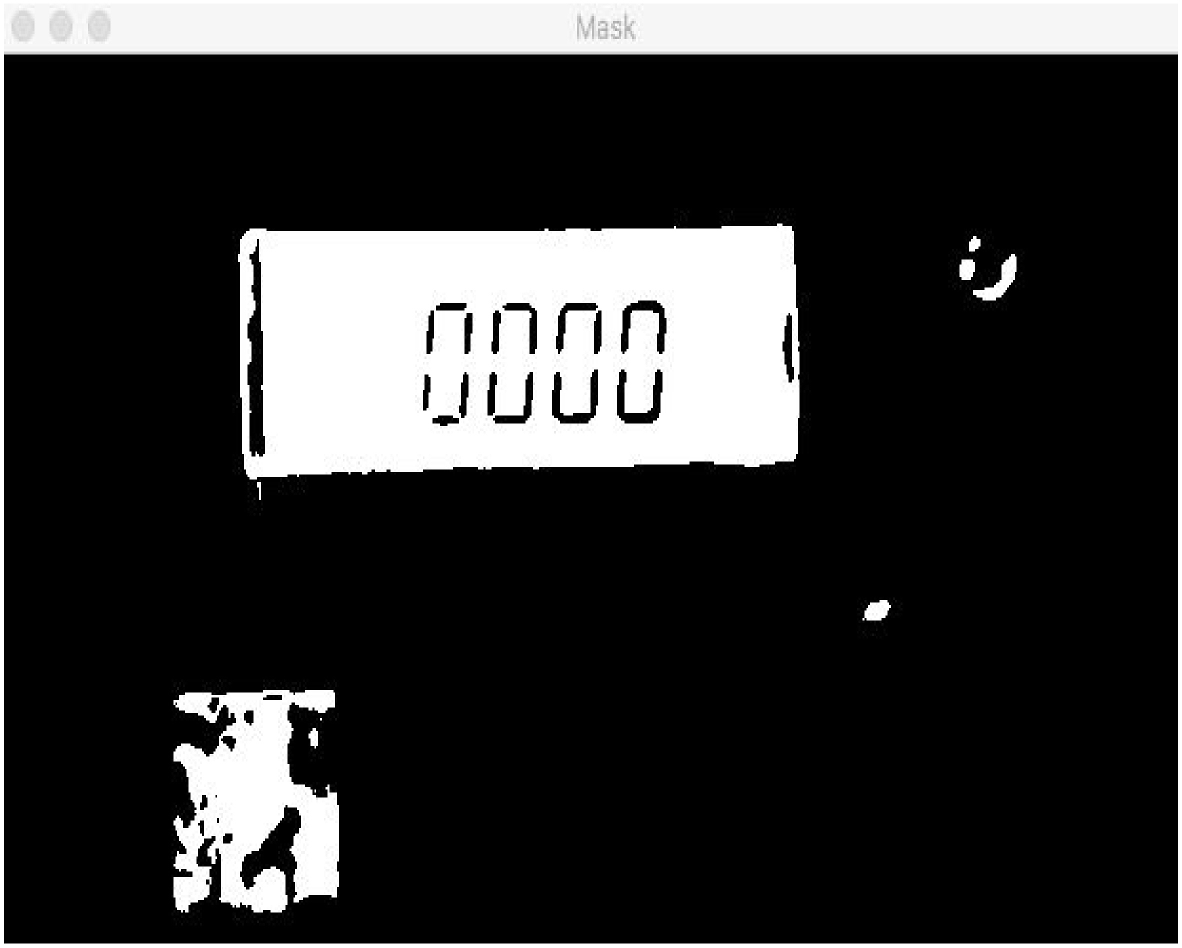
- For better accuracy, we use binary images. So before finding contours, we apply threshold Since OpenCV 3.2, **findContours()** no longer modifies the source image but returns a modified image as the first of three return parameters.
- In OpenCV, finding contours is like finding white object from black background. So object to be found is in white and background is in black.

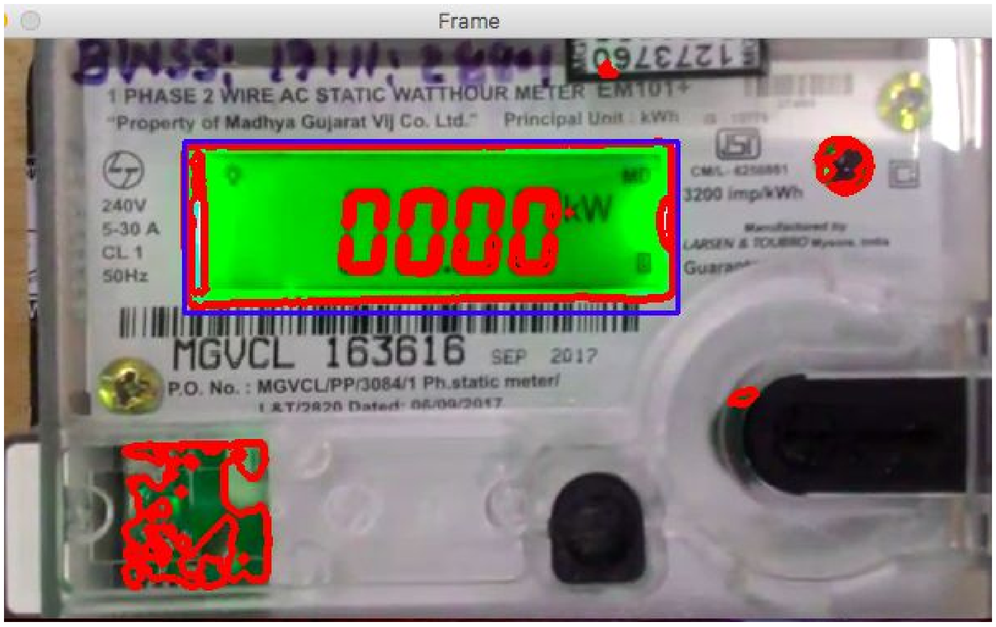


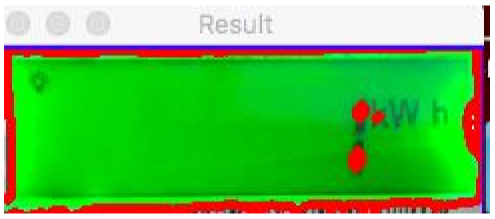
```python
#### FINAL PREPROCESSING CODE (meter 1 and 2)

import cv2
import numpy as np

cap= cv2.VideoCapture('/Users/malaikarastogi/Desktop/electric_meter.mp4')

while True:
    ret, frame = cap.read()
    blurred_frame = cv2.GaussianBlur(frame, (5, 5), 0)
    hsv = cv2.cvtColor(blurred_frame, cv2.COLOR_BGR2HSV)

    lower_green = np.array([60, 65, 65])
    upper_green = np.array([80, 255, 255])
    mask = cv2.inRange(hsv, lower_green, upper_green)

    _, contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

    cv2.drawContours(frame, contours, -1, (0, 0, 255), 3)

    #find the biggest area
    c = max(contours, key = cv2.contourArea)

    x,y,w,h = cv2.boundingRect(c)
    # draw the reading area contour (in blue)
    im=cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

    roi=im[y:y+h,x:x+w]


    cv2.imshow("Frame", frame)
    cv2.imshow("Mask", mask)
    cv2.imshow("Result",roi)

    key = cv2.waitKey(1)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Here, we have color filtered the video for green by giving it a range with lower and upper array to detect the colour. Then we mask it for image binarization i.e. object in white and background in black.

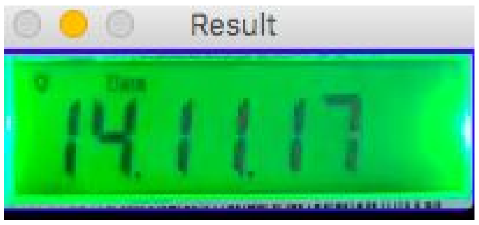
Then we find contours in that region using findContours() function. After that the contour with largest area i.e. the digital reading area is taken as the region of interest which is desired.


Mask
0000


Frame
1 PHASE 2 WIRE AC STATIC WATTHOUR METER EM101+
"Property of Madhya Gujarat Vij Co. Ltd." Principal Unit : kWh
240V
5-30 A
CL 1
50Hz
0000 kW
3200 imp/kWh
LARSEN & TOUBRO Mysore, India
MGVCL 163616 SEP 2017
P.O. No. : MGVCL/PP/3084/1 Ph.static meter/



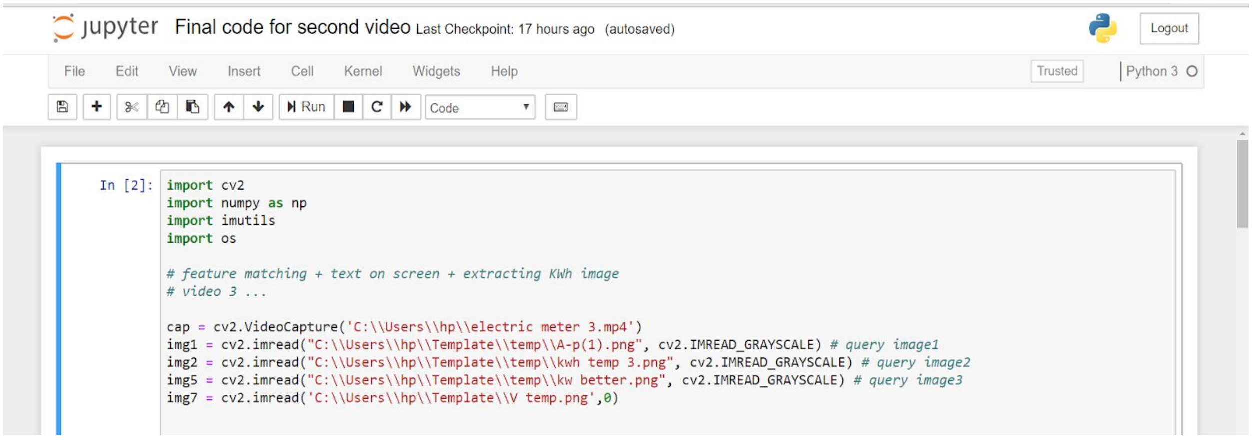
Largest area is bounded by blue rectangle



Results obtained with 100% efficiency.

Feature Matching

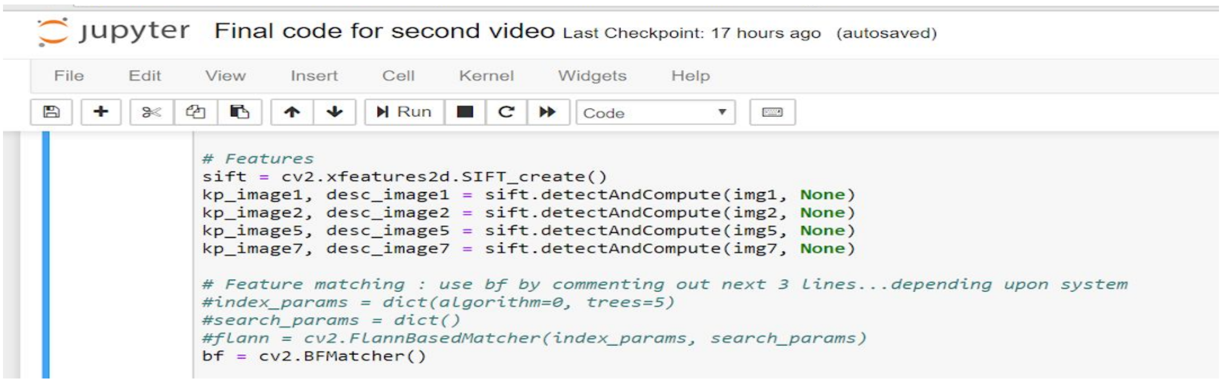
1.)


jupyter Final code for second video Last Checkpoint: 17 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [2]: import cv2
        import numpy as np
        import imutils
        import os

        # feature matching + text on screen + extracting KWh image
        # video 3 ...

        cap = cv2.VideoCapture('C:\\Users\\hp\\electric meter 3.mp4')
        img1 = cv2.imread("C:\\Users\\hp\\Template\\temp\\A-p(1).png", cv2.IMREAD_GRAYSCALE) # query image1
        img2 = cv2.imread("C:\\Users\\hp\\Template\\temp\\kwh temp 3.png", cv2.IMREAD_GRAYSCALE) # query image2
        img5 = cv2.imread("C:\\Users\\hp\\Template\\temp\\kw better.png", cv2.IMREAD_GRAYSCALE) # query image3
        img7 = cv2.imread('C:\\Users\\hp\\Template\\V temp.png',0)
```

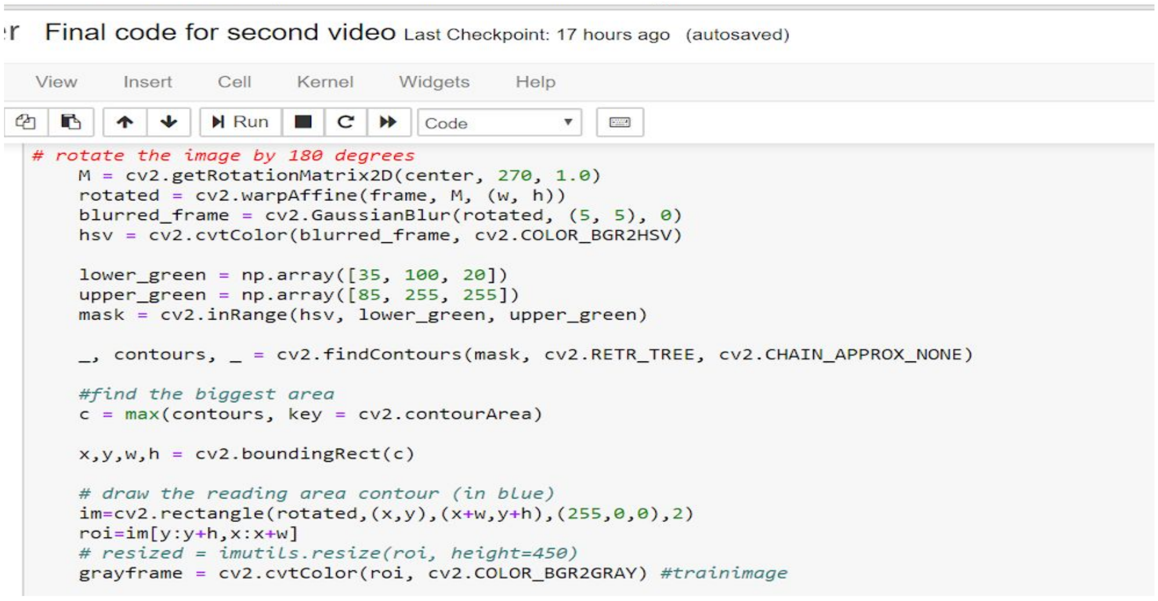
- In the first step we are importing various python libraries like OpenCV, Numpy, Imutils and Os.
- After that we are capturing our dataset(Video) using cv2.VideoCapture and loading multiple templates (A , V , KWH , KW) using cv2.imread.

2.)


jupyter Final code for second video Last Checkpoint: 17 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
# Features
sift = cv2.xfeatures2d.SIFT_create()
kp_image1, desc_image1 = sift.detectAndCompute(img1, None)
kp_image2, desc_image2 = sift.detectAndCompute(img2, None)
kp_image5, desc_image5 = sift.detectAndCompute(img5, None)
kp_image7, desc_image7 = sift.detectAndCompute(img7, None)

# Feature matching : use bf by commenting out next 3 lines...depending upon system
#index_params = dict(algorithm=0, trees=5)
#search_params = dict()
#flann = cv2.FlannBasedMatcher(index_params, search_params)
bf = cv2.BFMatcher()
```

- After this we have used SIFT(Scale-Invariant Feature Transform) algorithm to detect and describe local features in images.
- BFMatcher is going to try all the possibilities due to brute force matching and finds the best matches.

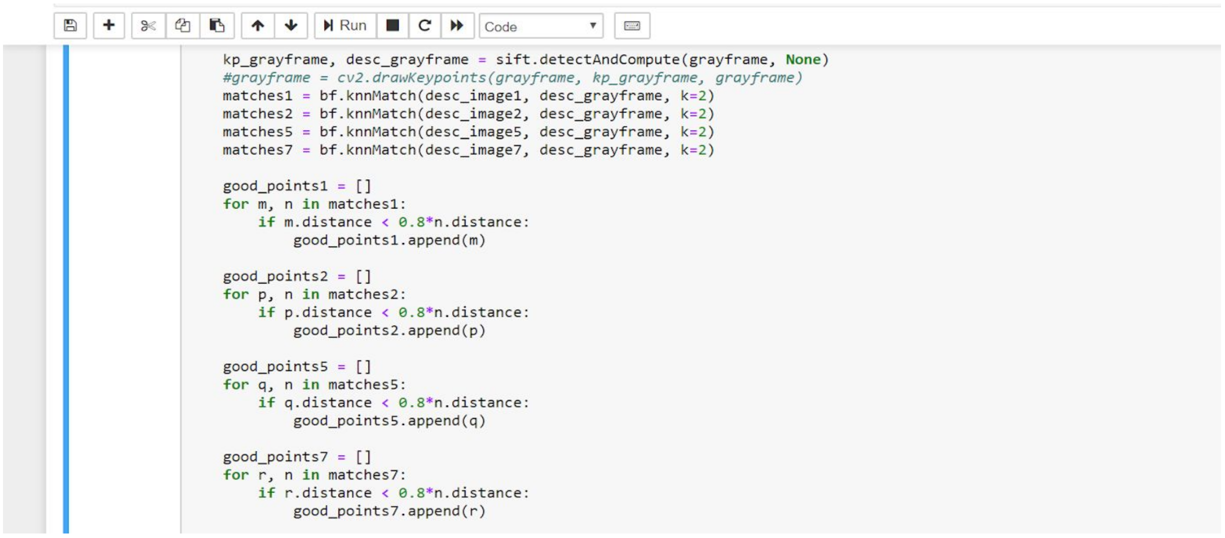

Final code for second video Last Checkpoint: 17 hours ago (autosaved)
View Insert Cell Kernel Widgets Help
Run
Code

```
# rotate the image by 180 degrees
    M = cv2.getRotationMatrix2D(center, 270, 1.0)
    rotated = cv2.warpAffine(frame, M, (w, h))
    blurred_frame = cv2.GaussianBlur(rotated, (5, 5), 0)
    hsv = cv2.cvtColor(blurred_frame, cv2.COLOR_BGR2HSV)

    lower_green = np.array([35, 100, 20])
    upper_green = np.array([85, 255, 255])
    mask = cv2.inRange(hsv, lower_green, upper_green)

    _, contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

    #find the biggest area
    c = max(contours, key = cv2.contourArea)

    x,y,w,h = cv2.boundingRect(c)

    # draw the reading area contour (in blue)
    im=cv2.rectangle(rotated,(x,y),(x+w,y+h),(255,0,0),2)
    roi=im[y:y+h,x:x+w]
    # resized = imutils.resize(roi, height=450)
    grayframe = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY) #trainimage
```

3.)

- To change the orientation from landscape to portrait we need to rotate the frame using Affine Transformation .
- Rotated frame is converted to HSV from BGR using cv2.cvtColor.
- Now we are using the preprocessed code to extract the region of interest as explained above.

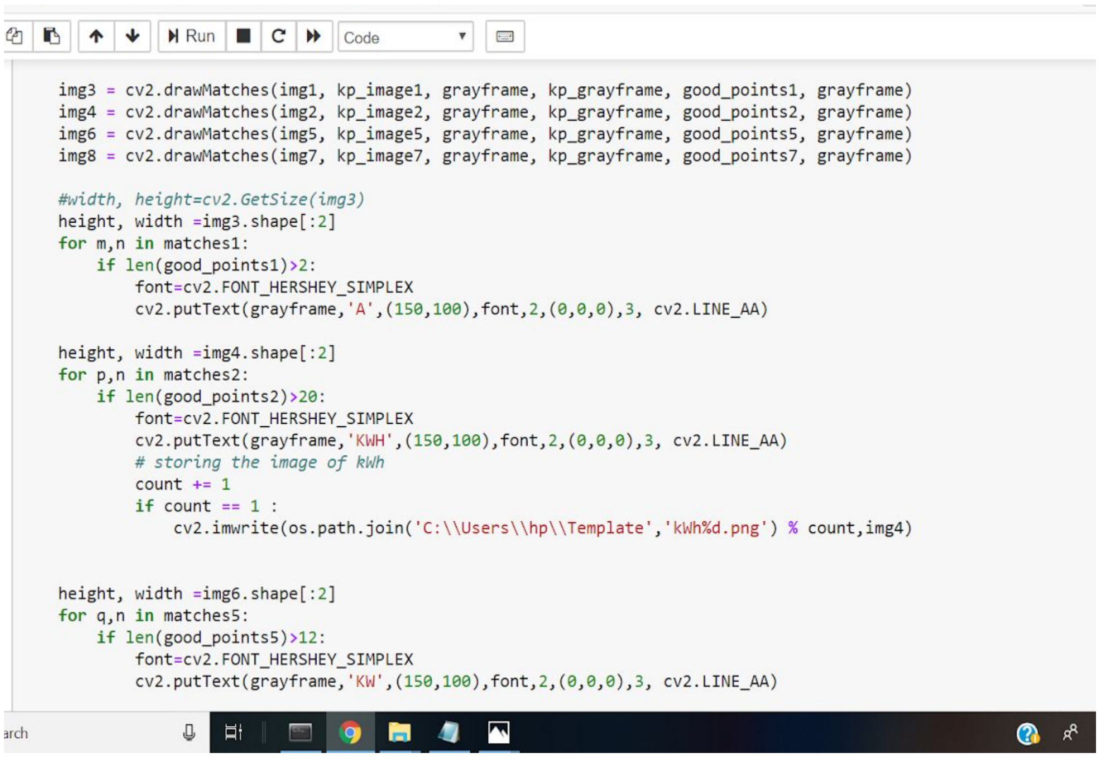
4.)



```
kp_grayframe, desc_grayframe = sift.detectAndCompute(grayframe, None)
#grayframe = cv2.drawKeypoints(grayframe, kp_grayframe, grayframe)
matches1 = bf.knnMatch(desc_image1, desc_grayframe, k=2)
matches2 = bf.knnMatch(desc_image2, desc_grayframe, k=2)
matches5 = bf.knnMatch(desc_image5, desc_grayframe, k=2)
matches7 = bf.knnMatch(desc_image7, desc_grayframe, k=2)

good_points1 = []
for m, n in matches1:
    if m.distance < 0.8*n.distance:
        good_points1.append(m)

good_points2 = []
for p, n in matches2:
    if p.distance < 0.8*n.distance:
        good_points2.append(p)

good_points5 = []
for q, n in matches5:
    if q.distance < 0.8*n.distance:
        good_points5.append(q)

good_points7 = []
for r, n in matches7:
    if r.distance < 0.8*n.distance:
        good_points7.append(r)
```

- Using SIFT algorithm we again detect the features and descriptors of our video.
- bf.knnMatch takes the descriptor of one feature in first set and is matched with all other features in second set and the closest one is returned .
- Now we are using Ratio Test proposed by David Lowe. This test rejects the poor matches by computing the ratio between the best and second-best match and if the ratio is below some threshold, the match is discarded as being low quality.

5.)



```
img3 = cv2.drawMatches(img1, kp_image1, grayframe, kp_grayframe, good_points1, grayframe)
img4 = cv2.drawMatches(img2, kp_image2, grayframe, kp_grayframe, good_points2, grayframe)
img6 = cv2.drawMatches(img5, kp_image5, grayframe, kp_grayframe, good_points5, grayframe)
img8 = cv2.drawMatches(img7, kp_image7, grayframe, kp_grayframe, good_points7, grayframe)

#width, height=cv2.GetSize(img3)
height, width =img3.shape[:2]
for m,n in matches1:
    if len(good_points1)>2:
        font=cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(grayframe,'A',(150,100),font,2,(0,0,0),3, cv2.LINE_AA)

height, width =img4.shape[:2]
for p,n in matches2:
    if len(good_points2)>20:
        font=cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(grayframe,'KWH',(150,100),font,2,(0,0,0),3, cv2.LINE_AA)
        # storing the image of kWh
        count += 1
        if count == 1 :
            cv2.imwrite(os.path.join('C:\\Users\\hp\\Template','kWh%d.png') % count,img4)


height, width =img6.shape[:2]
for q,n in matches5:
    if len(good_points5)>12:
        font=cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(grayframe,'KW',(150,100),font,2,(0,0,0),3, cv2.LINE_AA)
```

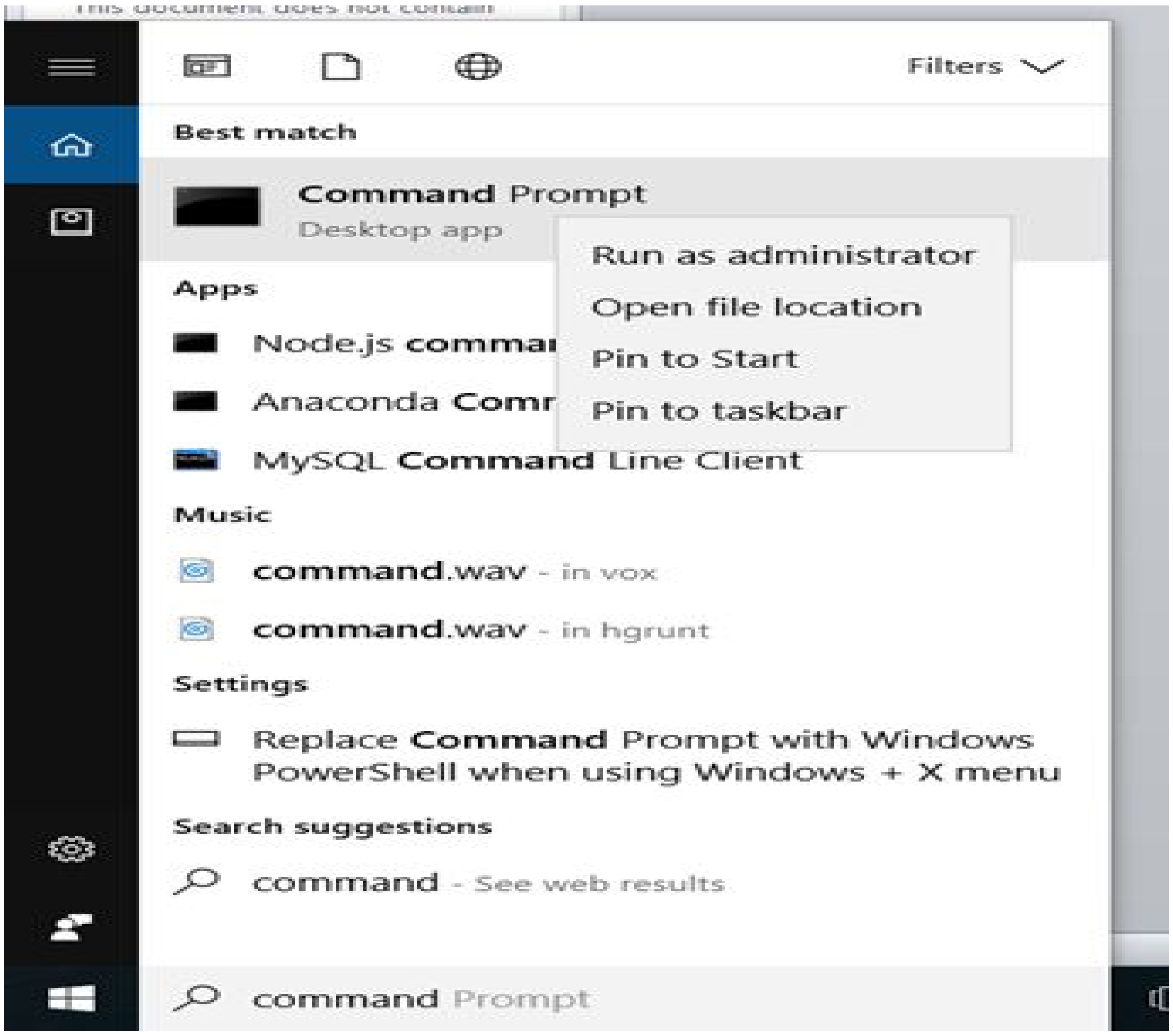
- cv2.drawMatches() helps us to draw matches. It stacks two images horizontally and helps us to draw lines from first image to second image showing best lines.
- We then use '**if**' condition for giving a threshold value on the number of good points to remove any erroneous matches.
- Using cv2.putText, we display the unit that we wanted to detect. Font and size of the text can be varied.

6. SYSTEM TESTING AND RESULTS

We were provided with a digital electric meter video as our dataset and this is the test sample on which we have tried the code. The code is saved in .py form and executed on Jupyter Notebook .

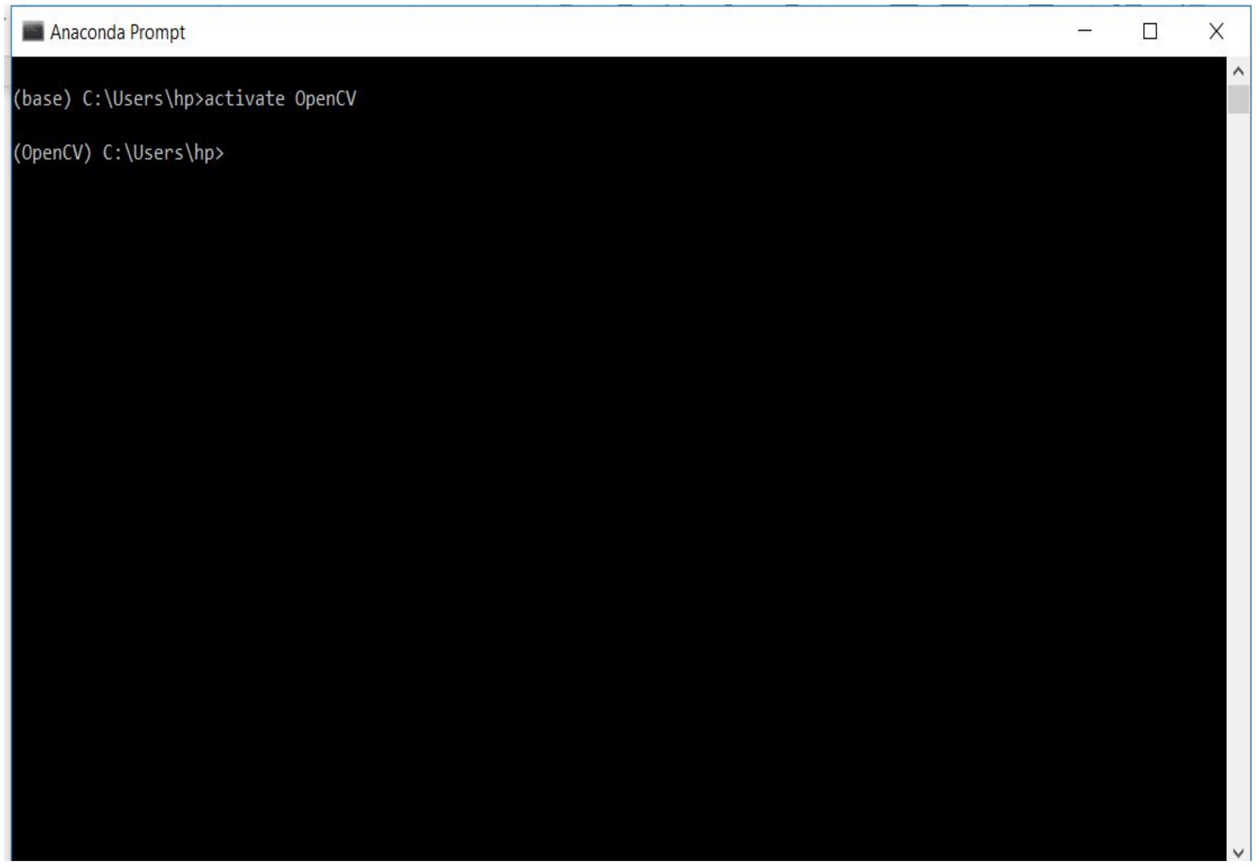
-Steps to implement the code.

Step1 -Right click on Command Prompt, Click Run as Administrator and click yes on the popup.



Step2- Then activate the virtual environment in which you have saved your code (Opencv in our case) .

You will see the command prompt screen as shown below.

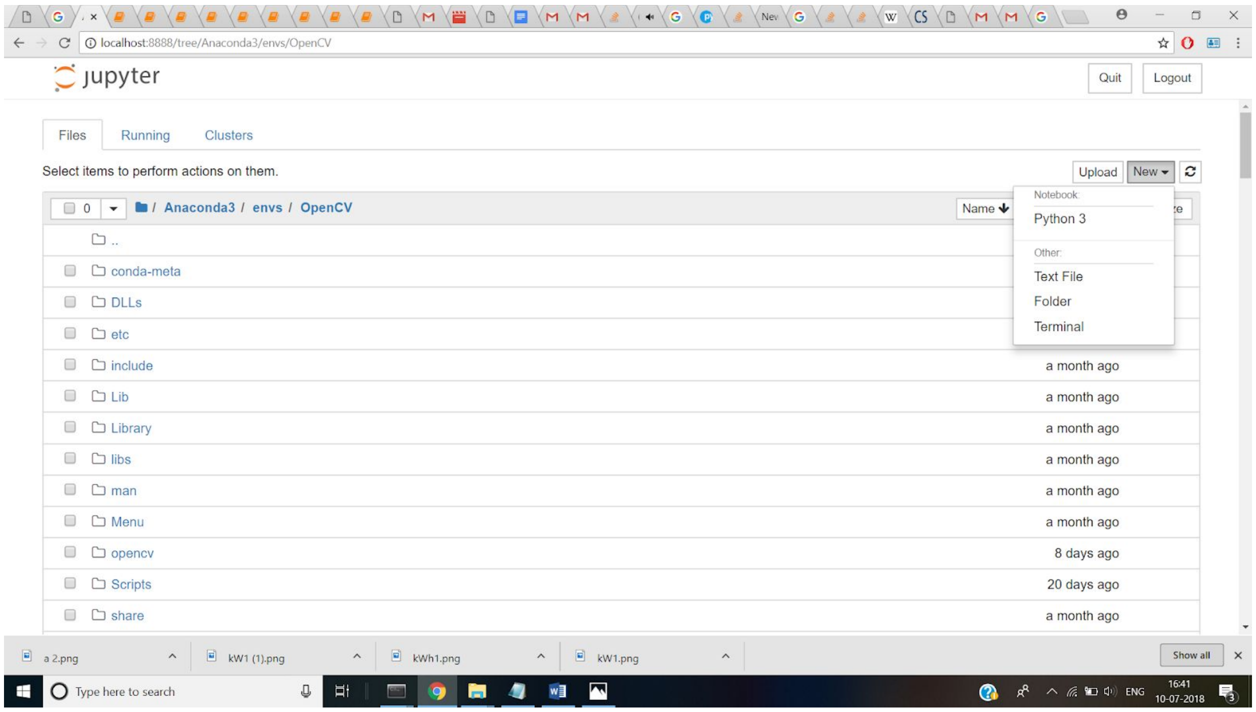


```
(base) C:\Users\hp>activate OpenCV

(OpenCV) C:\Users\hp>
```

Step 3 - Type jupyter notebook in your command prompt and press ENTER . A new jupyter window will be opened.

Step 4 - To work in your virtual environment click Anaconda3 -> envs -> OpenCV

Step 5 - Create a new notebook with Python 3 as shown below.



localhost:8888/tree/Anaconda3/envs/OpenCV

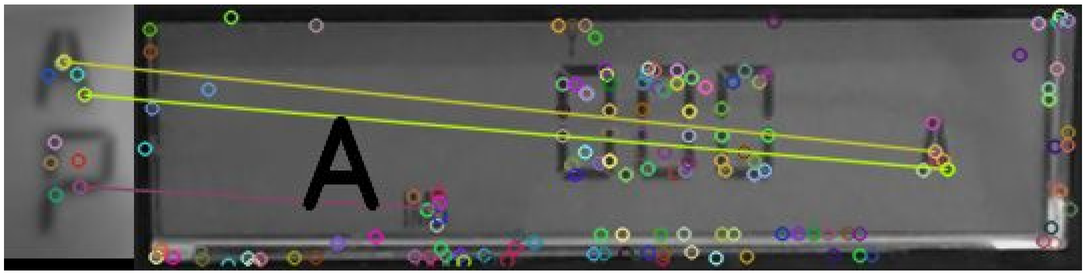
Select items to perform actions on them.

/ Anaconda3 / envs / OpenCV

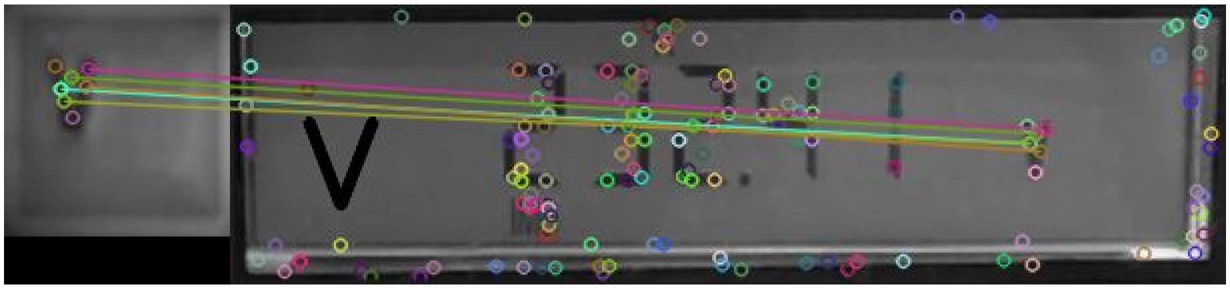
Name	Last Modified
..	
conda-meta	
DLLs	
etc	
include	a month ago
Lib	a month ago
Library	a month ago
libs	a month ago
man	a month ago
Menu	a month ago
opencv	8 days ago
Scripts	20 days ago
share	a month ago

Step 6 - Now we'll run our code on Jupyter notebook and it can be seen that the feature matching is giving satisfactory results.

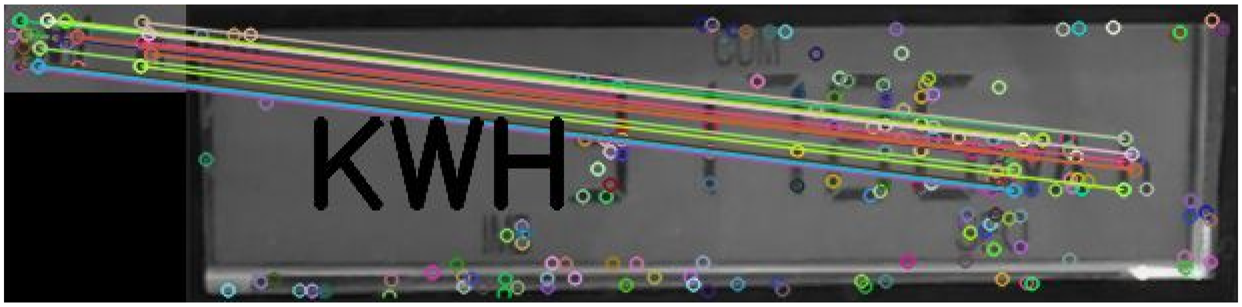
a.) Ampere:-



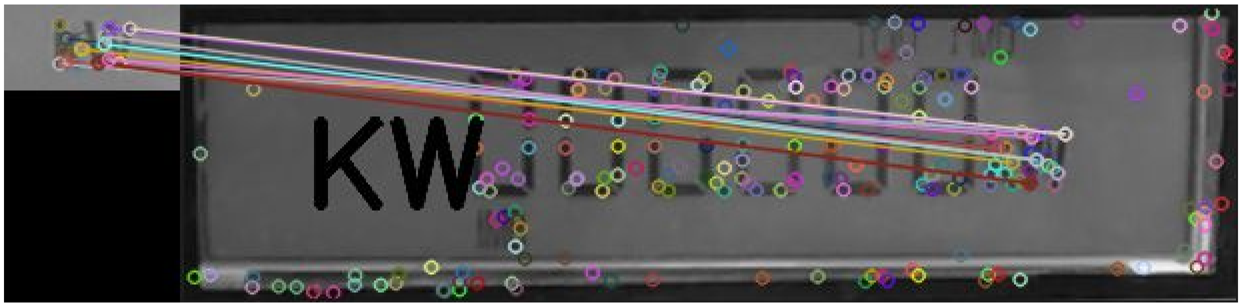
b.) Volt :-



c.) KiloWatt Hour :-



d.) KiloWatt :-



7. **Limitations and Future Scope**

7.1 Limitations:

- Distance from which the video is captured should not be larger than 1 feet0 otherwise it creates discrepancy in reading the digital area.
- Lighting needs to be taken care of. Uniformity is preferred
- Resolution of the video must also be taken care of. Low resolution leads to low quality of images giving less features hence difficulty in matching.
- There are three types of meters having different fonts which makes it difficult to apply one type of templates to all of them

7.2 Future Scope:

- The project can be extended for three phase meters also.
- QR code verification before capturing the video so that customer and meter information will be stored in the application instead of using bar code which provides less readability.
- Addition of Geolocation software to the application so that at the first time itself location of that particular area will be stored in the app and the user will be allowed to capture video from that meter only.
- Application will become open to users to use.

8. Conclusion and Bibliography

8.1 Conclusion

This project is a successful attempt to digitalize the reading of electricity meters by extracting four readings namely volt, ampere, kilowatt and kilowatt hour. The numerical reading area is 100% extracted of any meter and properly pre-processed.Then the units are read with the use of feature matching. Moreover, as soon as the unit is detected for the first time, the video frame is captured and saved in the system as an image in .jpg format for future reference and verification.

This work can make the electricity billing system time and cost effective for both the company and the user. With the help of this application manpower will be saved, human errors will be reduced, and will make payments procedures faster than before.

Different resolutions while using various smartphones is still a problem to be dealt with. An average distance and lighting must be taken in consideration while capturing the video for best results.

8.2 Bibliography

- “Automatic Electricity Meter Reading Based on Image Processing” by Lamiaa A. Elrefaei, Asrar Bajaber, Sumayyah Natheir, Nada AbuSanab, Marwa Bazi, 2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT) .
- Learning Python using New Boston tutorials on youtube .
- OpenCV Tutorials from https://pythonprogramming.net/.
- pyimagesearch and py source websites for feature matching algorithm references
- Stack overflow website for bugs fixing https://medium.com/init27-labs/installation-of-opencv-using-anaconda-mac-faded05a4ef6 .
- https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/
- https://www.geeksforgeeks.org/template-matching-using-opencv-in-python/ for template matching references.
- Sample videos provided by GPRD.

9. APPENDIX

```
import cv2
import numpy as np
import imutils
import os

# feature matching + text on screen + extracting KWh image
# video 3 ...

cap = cv2.VideoCapture('C:\\Users\\hp\\electric meter 3.mp4')
img1 = cv2.imread("C:\\Users\\hp\\Template\\temp\\A-p(1).png",
cv2.IMREAD_GRAYSCALE) # query image1
img2 = cv2.imread("C:\\Users\\hp\\Template\\temp\\kwh temp 3.png",
cv2.IMREAD_GRAYSCALE) # query image2
img5 = cv2.imread("C:\\Users\\hp\\Template\\temp\\kw better.png",
cv2.IMREAD_GRAYSCALE) # query image3
img7 = cv2.imread('C:\\Users\\hp\\Template\\V temp.png',0)


count = 0

#w,h = template.shape[::-1]
#for scale in np.linspace(.2, 1.0, 20)[::-1]:
#resized1 = imutils.resize(template, width = int(template.shape[1] * .35))
#r = template.shape[1] / float(resized1.shape[1])

# img1 = imutils.resize(img1, height=50)
# img2 = imutils.resize(img2, height=50)
# img5 = imutils.resize(img5, height=50)

# Features
sift = cv2.xfeatures2d.SIFT_create()
kp_image1, desc_image1 = sift.detectAndCompute(img1, None)
kp_image2, desc_image2 = sift.detectAndCompute(img2, None)
kp_image5, desc_image5 = sift.detectAndCompute(img5, None)
kp_image7, desc_image7 = sift.detectAndCompute(img7, None)

# Feature matching : use bf by commenting out next 3 lines...depending upon
system
#index_params = dict(algorithm=0, trees=5)
#search_params = dict()
#flann = cv2.FlannBasedMatcher(index_params, search_params)
bf = cv2.BFMatcher()

while True:
    ret, frame = cap.read()

    (h, w) = frame.shape[:2]
    center = (w / 2, h / 2)

# rotate the image by 180 degrees
    M = cv2.getRotationMatrix2D(center, 270, 1.0)
    rotated = cv2.warpAffine(frame, M, (w, h))
    blurred_frame = cv2.GaussianBlur(rotated, (5, 5), 0)
    hsv = cv2.cvtColor(blurred_frame, cv2.COLOR_BGR2HSV)
    lower_green = np.array([35, 100, 20])
    upper_green = np.array([85, 255, 255])
    mask = cv2.inRange(hsv, lower_green, upper_green)

    _, contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

    #find the biggest area
    c = max(contours, key = cv2.contourArea)

    x,y,w,h = cv2.boundingRect(c)

    # draw the reading area contour (in blue)
    im=cv2.rectangle(rotated,(x,y),(x+w,y+h),(255,0,0),2)
    roi=im[y:y+h,x:x+w]
    # resized = imutils.resize(roi, height=450)
    grayframe = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY) #trainimage

    # template matching
   # gray = cv2.cvtColor(rotated, cv2.COLOR_BGR2GRAY)
    #res = cv2.matchTemplate(gray,resized1,cv2.TM_CCOEFF_NORMED)
    #threshold = 0.90

   # loc = np.where( res >= threshold)
    # if loc.any() is True :

    #for pt in zip(*loc[::-1]):
     #   cv2.rectangle(frame, pt, (pt[0] + int(w*0.35), pt[1] + int(h*0.35)),
(0,255,255), 1)
```

```
  #  font=cv2.FONT_HERSHEY_SIMPLEX
   # cv2.putText(frame,'V',(796,240),font,2,(0,0,0),5, cv2.LINE_AA)


kp_grayframe, desc_grayframe = sift.detectAndCompute(grayframe, None)
#grayframe = cv2.drawKeypoints(grayframe, kp_grayframe, grayframe)
matches1 = bf.knnMatch(desc_image1, desc_grayframe, k=2)
matches2 = bf.knnMatch(desc_image2, desc_grayframe, k=2)
matches5 = bf.knnMatch(desc_image5, desc_grayframe, k=2)
matches7 = bf.knnMatch(desc_image7, desc_grayframe, k=2)

good_points1 = []
for m, n in matches1:
    if m.distance < 0.8*n.distance:
        good_points1.append(m)

good_points2 = []
for p, n in matches2:
    if p.distance < 0.8*n.distance:
        good_points2.append(p)

good_points5 = []
for q, n in matches5:
     if q.distance < 0.8*n.distance:
        good_points5.append(q)

good_points7 = []
for r, n in matches7:
        if r.distance < 0.8*n.distance:
            good_points7.append(r)


    img3 = cv2.drawMatches(img1, kp_image1, grayframe, kp_grayframe,
good_points1, grayframe)
    img4 = cv2.drawMatches(img2, kp_image2, grayframe, kp_grayframe,
good_points2, grayframe)
    img6 = cv2.drawMatches(img5, kp_image5, grayframe, kp_grayframe,
good_points5, grayframe)
    img8 = cv2.drawMatches(img7, kp_image7, grayframe, kp_grayframe,
good_points7, grayframe)


    #width, height=cv2.GetSize(img3)
    height, width =img3.shape[:2]
    for m,n in matches1:
        if len(good_points1)>2:
            font=cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(grayframe,'A',(150,100),font,2,(255,255,255),3,
cv2.LINE_AA)


    height, width =img4.shape[:2]
    for p,n in matches2:
        if len(good_points2)>20:
            font=cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(grayframe,'KWH',(150,100),font,2,(255,255,255),3,
cv2.LINE_AA)
            # storing the image of kWh
             count += 1
            if count == 1 :
                cv2.imwrite(os.path.join('C:\\Users\\hp\\Template','kWh%d.png') %
count,img4)



    height, width =img6.shape[:2]
    for q,n in matches5:
        if len(good_points5)>12:
            font=cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(grayframe,'KW',(150,100),font,2,(255,255,255),3,
cv2.LINE_AA)

    height, width =img8.shape[:2]
    for r,n in matches7:
        if len(good_points7)>4:
            font=cv2.FONT_HERSHEY_SIMPLEX
            cv2.putText(grayframe,'V',(150,100),font,2,(255,255,255),3,
cv2.LINE_AA)

    #cv2.imshow("Frame", rotated)
    #cv2.imshow("A", img3)
    #cv2.imshow("KWH", img4)
    #cv2.imshow("KW", img6)
    #cv2.imshow("V", img8)
    cv2.imshow("result",grayframe)
     key = cv2.waitKey(1)
    if key == 27:
         break
cap.release()
cv2.destroyAllWindows()
```