Smart Surveillance Parking Camera

Presented By:
Jeet Thakore (181020011008)
Satyajeet Kumar Verma (181020012015)

Under the Guidance of:
Dr. Raghavendra H. Bhalerao
Assistant Professor, IITRAM

Department Of Electrical and Computer Science (EECS)
Institute of Infrastructure, Technology, Research And Management

Contents

Literature Survey

TITLE OF THE RESEARCH PAPER	AUTHOR	YEAR OF PUBLICATION	MOTIVATION TAKEN FROM THE STUDY
Face Detection and Tracking using OpenCV	S.V, Viraktamath, Mukund Katti, Aditya Khatawkar and Pavan Kulkarni.	2015	How a surveillance camera can record the event, identify, and recognise the individual's face
Study on Object Detection using Open CV - Python	Gupta, B., Chaube, A., Negi, A., & Goel, U.	2017	Distinguish a specific object among a vast number of objects in real time
Object Detection System using Arduino and Android Application for Visually Impaired People	Vadwala, Ms.Ayushi & Karmakar, Ms.Yesha & Suthar, Ms.Krina & Thakkar, Nirali.	2018	Working principle and applications of an Arduino board together with Android

Motivation

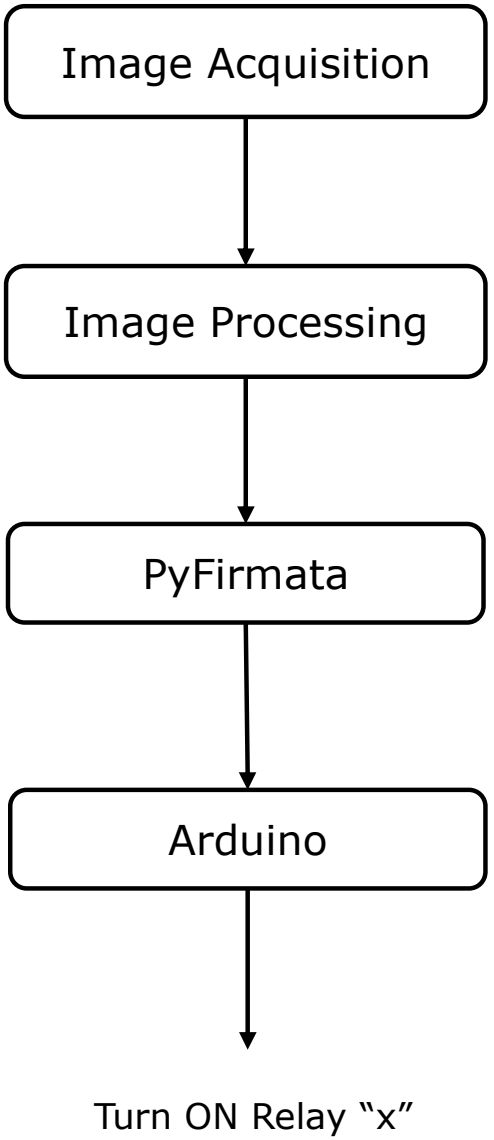
- Usually, All the lights are ON
- Tremendous waste of energy
- Need felt for controlled illumination based on object presence

Fig.1,2 Standard Parking Lots

Image Courtesy: Wikipedia

Proposed Solution

- Only Those lights must be turned on which have a vehicle under them.
- Object detection can be used.
- Knowledge of Computer vision and Image processing can be used.



Methodology

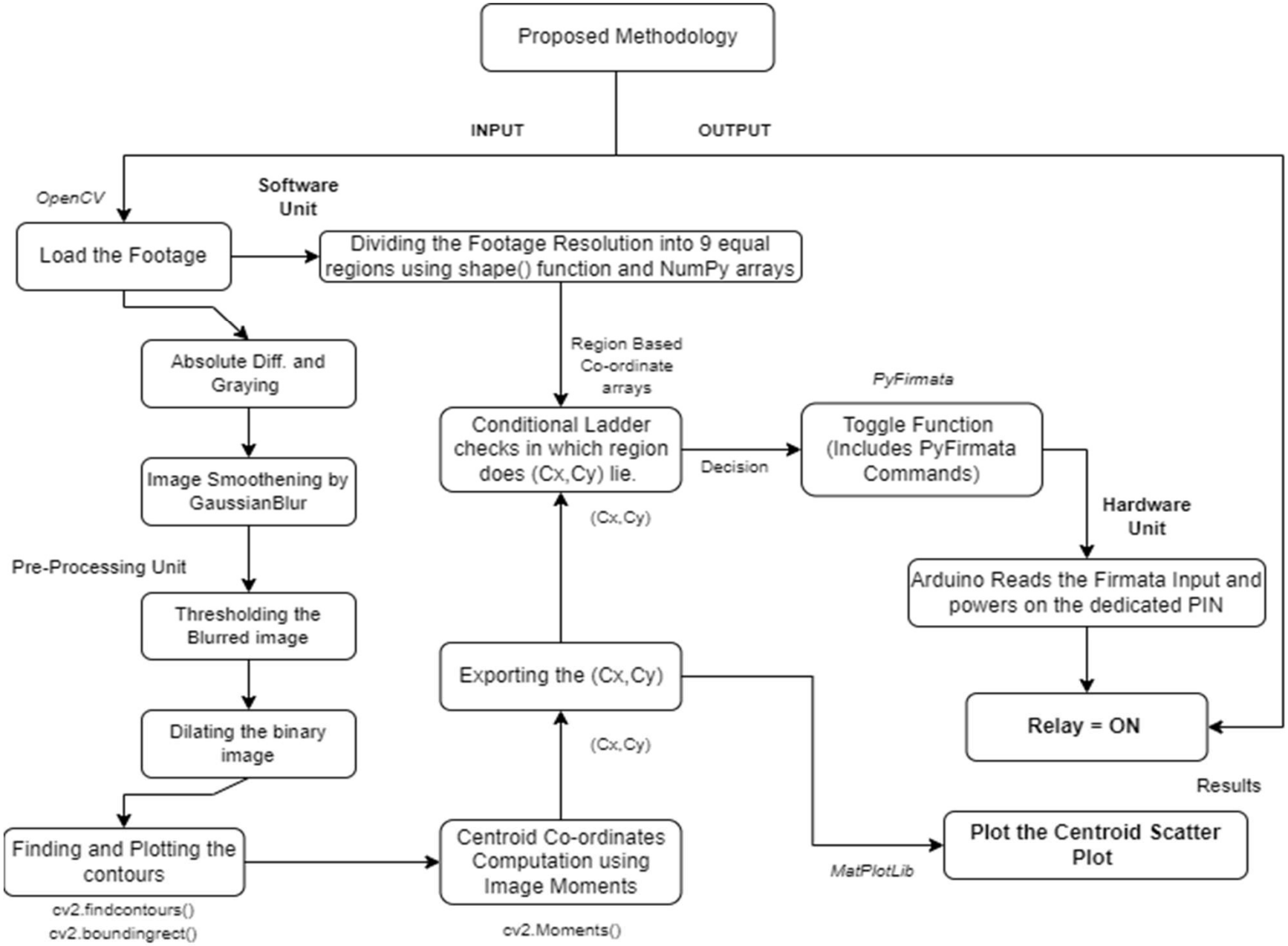
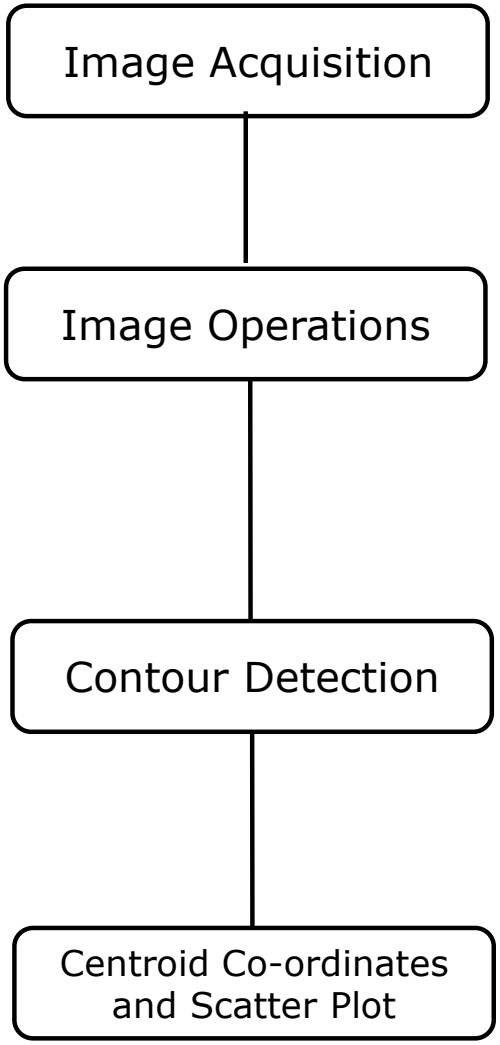
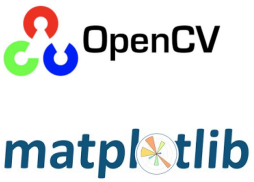


Image Processing

- Image Operations
- Contour Detection
- Centroid



Python



Arduino

Image Operations

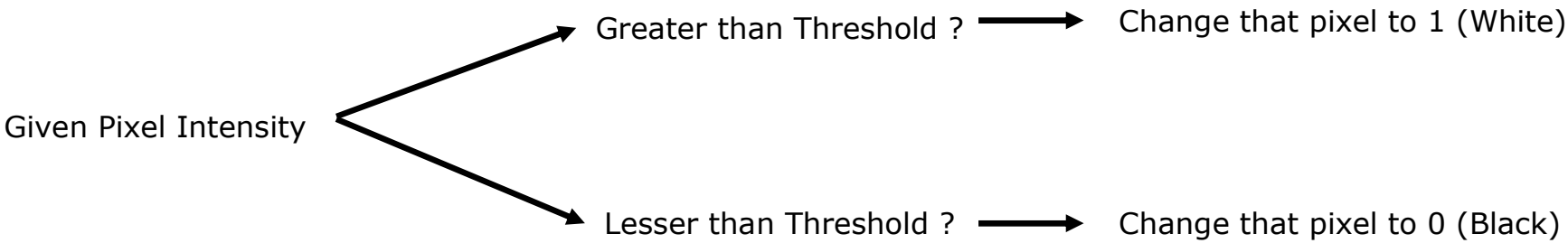
- Gaussian Blur

- Low pass filter used.
- Reduces Noise.
- Improves Detection.

Fig.3,4 Original image (Left) and Blurred Image (Right)

Image Operations

- Thresholding



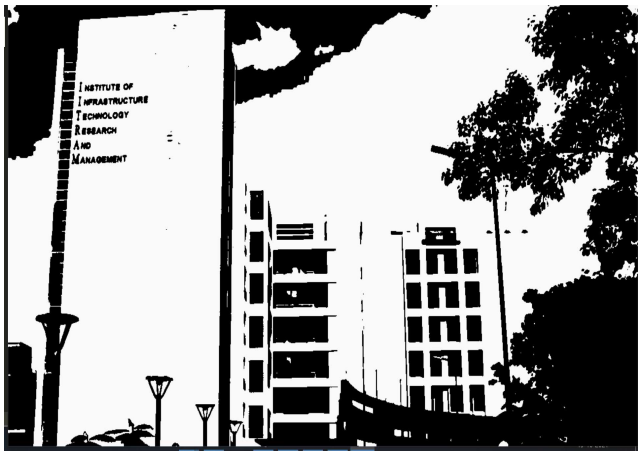


Fig.5,6 Original image (Left) and Thresholded Image (Right)

Image Operations

- Dilation

- Expands the boundary pixels.
- Adds pixels to the object boundary.
- Enhances Features of the image.

Fig.7,8 Original image (Left) and Dilated Image (Right)

Recap: Contour Detection

- When we join all the points on the boundary of an object, we get a contour.
- Processed image is fed to contour detection.

Fig.9 Original image (Left) and Image with detected contour(s) (Right)

Image Courtesy: Learnopencv.com

Contour Detection Algorithms

CHAIN_APPROX_SIMPLE	CHAIN_APPROX_NONE
Removes all redundant points; Only the endpoints are retained	Retains each and every point
Compresses the contour, thereby saving memory	Uses more memory.
Faster	Slower

Centroid Computation

- Centroid

- Arithmetic mean of all the positions of a contour. (Weighted average of all the pixels)

$$\mathbf{c} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

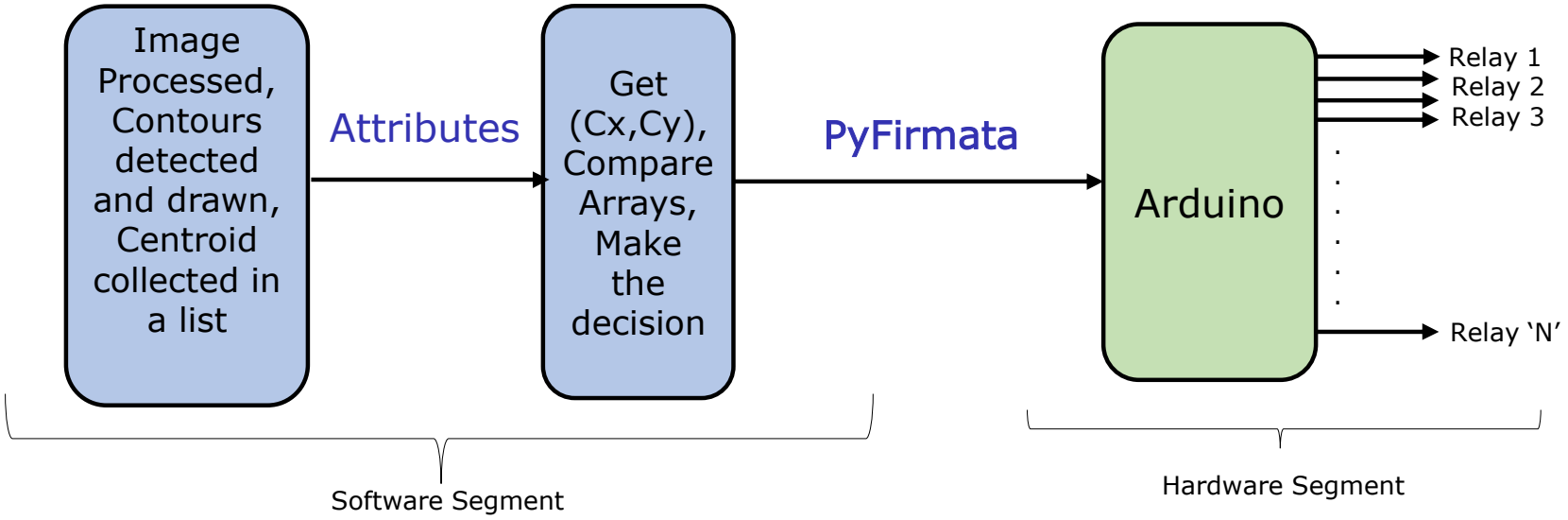
- **Image Moments:**
- Particular weighted average of image pixel intensities.
- Used to find specific properties of an image, like radius, area, centroid etc.
- Co-ordinates of Centroid (C_x,C_y) can be found in terms of the moment:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

$$C_x = \frac{M_{10}}{M_{00}} \qquad C_y = \frac{M_{01}}{M_{00}}$$

PyFirmata

- Abridges the gap between software and hardware.
- Works on Firmata protocol (MIDI Based)
- 2 modes : Either via Arduino itself or via host PC (StandardFirmata)



Grid Division (2D/Non-Trapezoidal)

- Entire resolution divided into 3x3 grid.
- Height and Width parsed via image. Shape() method and then fed to arrays.
- These arrays later used in decision making.

Fig.10 Process of dividing the entire footage into 9 probable regions.

Grid Division (Trapezoidal)

- Real-life scenario as most cameras are installed at an angle.
- Conventional 2D approach fails due to Geometric dissimilarity.
- Use Masking for optimum results. (To be discussed next)

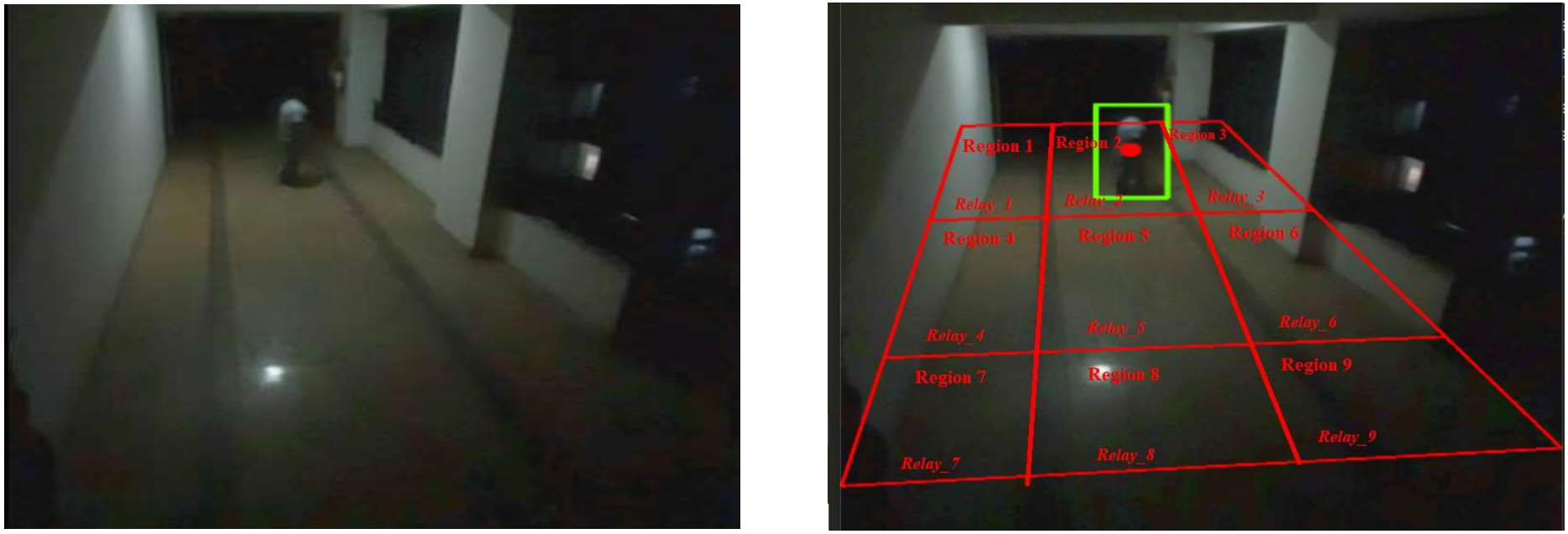


Fig.11 Process of dividing the entire footage into 9 trapezoidal regions.

Masking Methods

- Need : Geometric dissimilarity causes faulty detection.
- Solution : Replace the conventional grid divider by masking
- Masking methods:
  - Bitwise_AND
  - PointPolygonTest
  - Matplotlib.Path

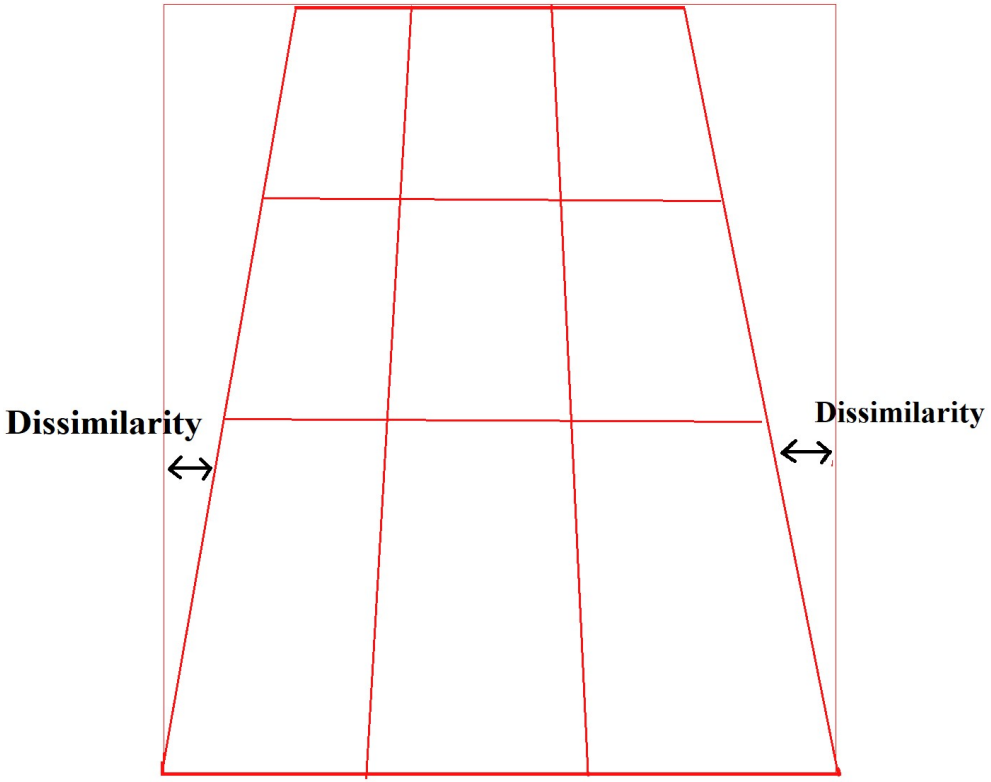


Fig.6 Rectangular vs Trapezoidal Grid.

Masking Methods (Contd.)

Bitwise_AND

- Function Used: *bitwiseAnd = cv2.bitwise_and(Regions1, Region2)*
- A bitwise AND is true if and only if both pixels are greater than zero.

PointPolygonTest

- Function Used: *cv2.pointPolygonTest(Polygon_pts_array, (x,y), False)*
- If Measure_Dist: "True": Returns Shortest Distance.
  If Measure_Dist: "False": Inside(1) Outside(-1) On(0)

Matplotlib.Path

- A Numpy array containing polygon coordinates is fed to *.path()* method.
- This above defined path is fed to *.contains_point()* method which returns True/False.

Simple/Global Thresholding

- Pixel intensity set to White if greater than a certain value, else Black. (Binarized)
- For every value, the same threshold is considered. (Global threshold).
- In OpenCV we have :

1. cv2.THRESH_BINARY
2. cv2.THRESH_BINARY_INV
3. cv2.THRESH_TRUNC
4. cv2.THRESH_TOZERO
5. cv2.THRESH_TOZERO_INV

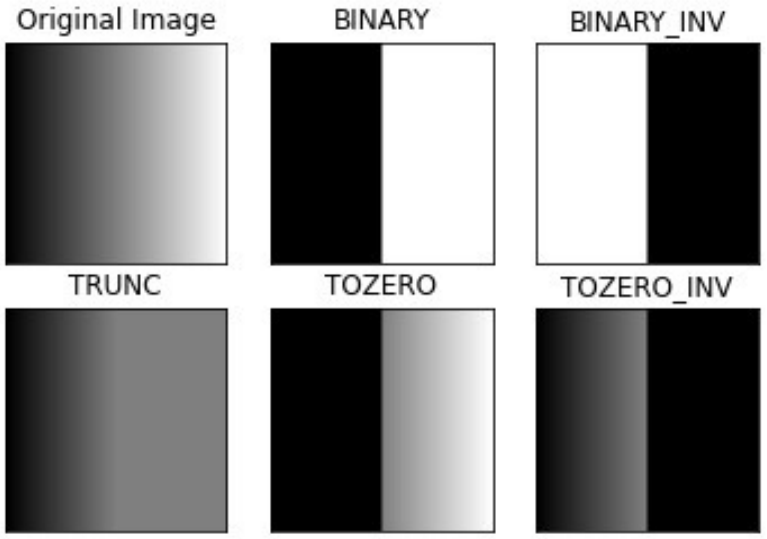


Fig.12 Types of Global Thresholding algorithms in OpenCV.

Image Courtesy: Learnopencv.com

Adaptive Thresholding

- Multiple thresholds for different image parts rather than having a single global threshold for the entire image.
- Regional Threshold decided as per neighborhood.
- Preferred for uneven lighting.

Fig.13 Global vs Adaptive Thresholding.

Image Courtesy: Learnopencv.com

Otsu's Method

- Threshold is decided as per the histogram of the image.
- Works best for Bimodal Images.

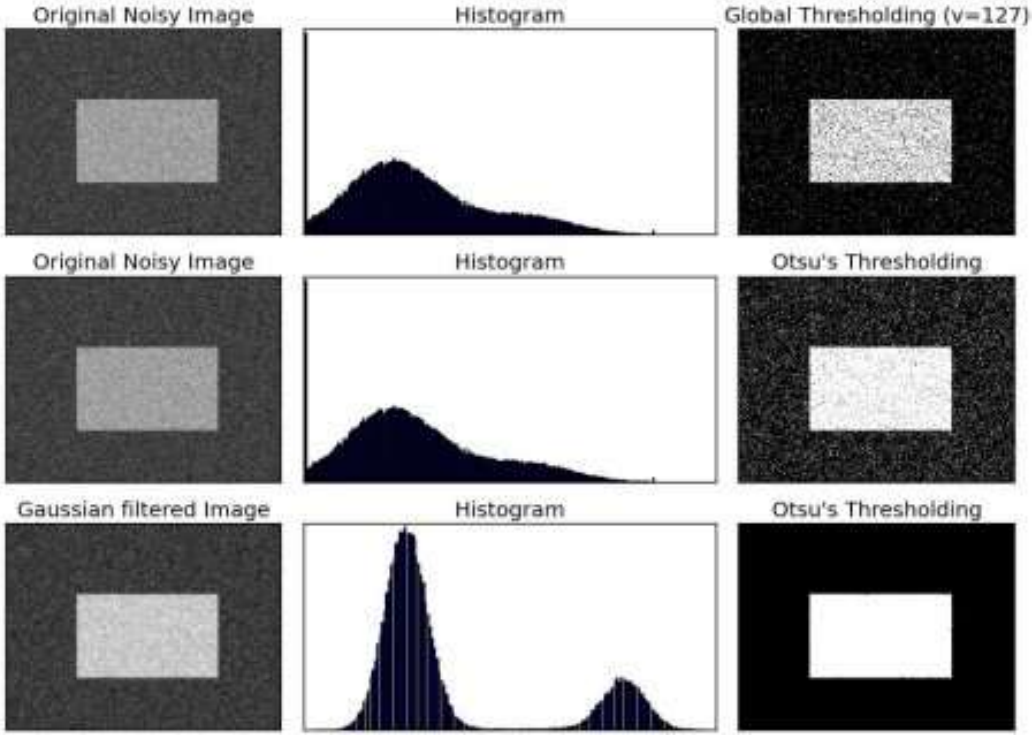


Fig.14 Implementing Otsu's Method on filtered image.

Image Courtesy: Learnopencv.com

Trackbars

- Hands-On Live Control over the parameters.
- Enhances the overall Accessibility.
- At present 2 trackbars;
  - Brightness Control
  - Manual Relay Control

  More can be added as per need.

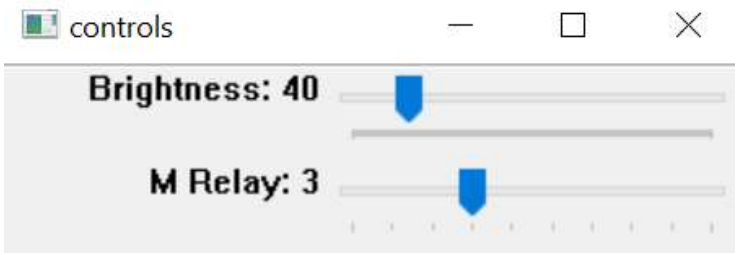


Fig.15 Trackbars.

Warp Perspective

- Perspective Transform; Transforms the viewpoint.
- Preserves Collinearity and Incidence. Doesn't Preserve Length and Angle.
- Mathematically,

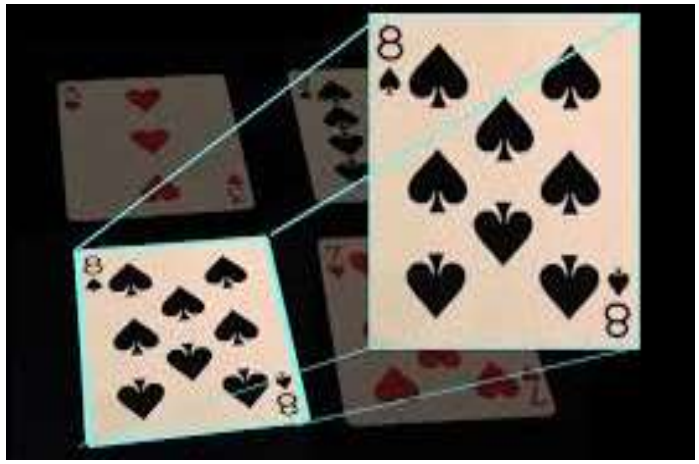


Fig.16 Warp Perspective : An Illustration

$$\begin{bmatrix} t_i x' \\ t_i y' \\ t_i \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$ Defines Transformtions

$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ Defines Translation Vector

$\begin{bmatrix} c_1 & c_2 \end{bmatrix}$ Projection Vector

t_i = Scaling Factor

Image Courtesy: WWW.AIlearner.com

Recap: Relays

- Electronic Switch that opens and closes W/O mechanical/manual operation.
- Conventional relay use electromagnetism for switching.
- Use: To achieve faster Switching
  To control multiple circuits via a single signal.

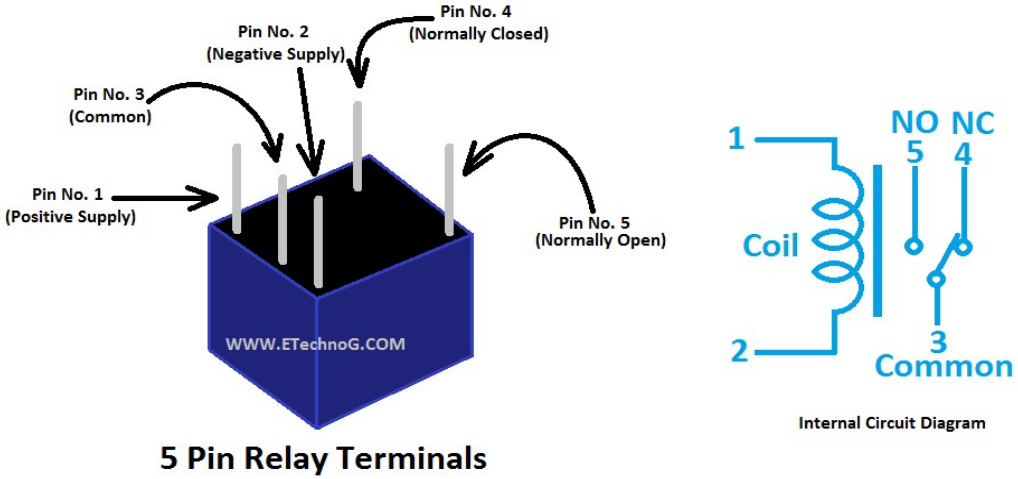


Fig.17 Relay Pins (Left) and Internal Circuit Diagram (Right)

Image Courtesy: ETechnoG.com

Recap: Arduino UNO

- Programmable micro-controller with a software IDE.
- IDE is C++ Based.

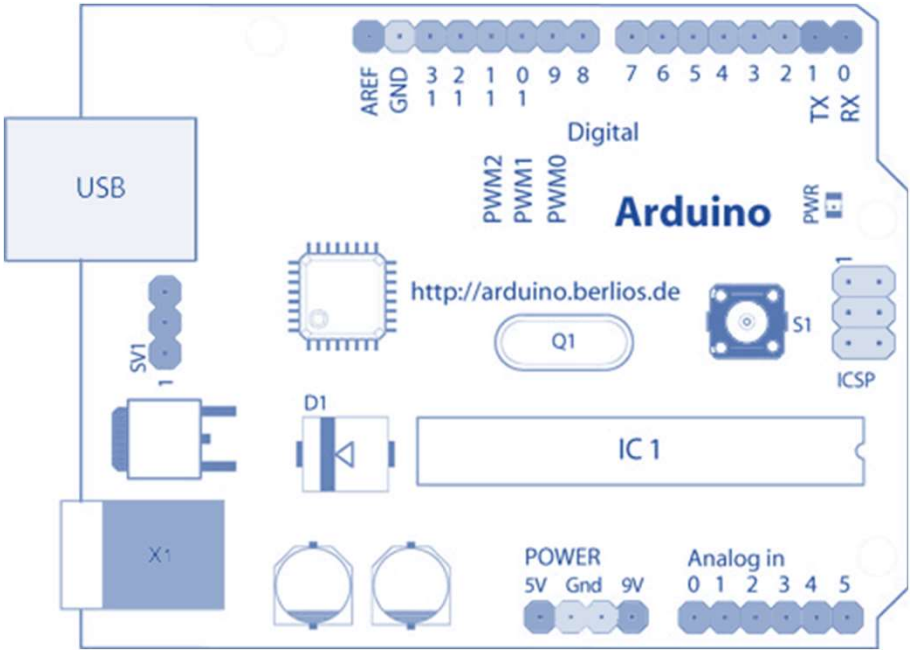


Fig.18 Arduino UNO Schematic Diagram

- Specs:
  - Atmega328p Micro-controller
  - 2kb SRAM
  - 32kb Flash + 1kb EEPROM
  - 16 MHz clock speed
  - Vin = 5v
  - 14 pins

Image Courtesy: arduino.cc/en/reference/board

Noise

- Salt and pepper noise

- Induced by addition of both random bright (with 255 pixel value) and random dark (with 0 pixel value) all over the image.
- Low-light invites a lot of noise.
- Also known as data drop noise.

Fig.19 Original image (Left) and Noisy Image (Right)

- Motion Blur

- Induced as the image being recorded changes during the recording of a single exposure, due to rapid movement or long exposure.

Fig.20 Original image (Left) and Motion Blurred Image (Right)

Image Courtesy: WWW.AILearner

Implications and Mitigation

Implications

- Can cause mis-triggering of the lights (Undesirable).
- Might cause Accidents.
- Unpleasant user experience.

Mitigation

- Improved Image Acquisition.
- Better Algorithms.

Conclusions

- Precision drops marginally in low-lighting scenarios as compared to well-lit conditions.
- The process of contour detection can be improved by improving the pre-processing unit of the project.
- Noise or unwanted entities might cause mis-triggering. Hence its imperative to reduce them to the best extent possible.
- Different footages have different parametric demands (i.e pre-processing parameters like kernel size, dilation iterations and so on).

References

[1] Gupta, Bhumika & Chaube, Ashish & Negi, Ashish & Goel, Umang. (2017). Study on Object Detection using Open CV - Python. International Journal of Computer Applications. 162. 17-21. 10.5120/ijca2017913391.
[2] S.V, Viraktamath, Mukund Katti, Aditya Khatawkar and Pavan Kulkarni. "Face Detection and Tracking using OpenCV." (2016).Guennouni, Souhail & Ali, Ahaitouf & Mansouri, Anass. (2015). Multiple Object Detection using OpenCV on an Embedded Platform. 2015. 374-377. 10.1109/CIST.2014.7016649.
[3] Vadwala, Ms.Ayushi & Karmakar, Ms.Yesha & Suthar, Ms.Krina & Thakkar, Nirali. (2018). Object Detection System using Arduino and Android Application for Visually Impaired People. International Journal of Computer Applications. 181. 975-8887.
[4] Guennouni, Souhail & Ali, Ahaitouf & Mansouri, Anass. (2015). Multiple Object Detection using OpenCV on an Embedded Platform. 2015. 374-377. 10.1109/CIST.2014.7016649.
[5] Turkay, Mustafa. (2017). OBJECT DETECTION AND TRACKING FOR REAL TIME FIELD SURVELLIANCE APPLICATIONS. 10.13140/RG.2.2.34716.49282
[6] Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. International Journal of Computer Vision, 59, 167–181.
[7] Shabbir Bhusari, Sumit Patil, Mandar Kalbhor. —Traffic Control System using raspberry-pi,‖ International Journal Of Advanced Engineering Technologies. [2] Rajeshwar Kumar Dewangan, Yamini Chouhan. A Review on Object Detection using Open CV Method. International Research Journal of Engineering and Technology (IRJET)

Thank You !