Subject Code:	21AI639
Subject Name:	Computer Vision
Roll No.:	Cb.en.p2cse21012
Name:	Raahul Varman
Branch:	M.Tech CSE

Course Instructor : Dr. SENTHILKUMAR .T , Associate Professor, Dept. of CSE

***in partial fulfillment of the requirements for the COURSE* – 21AI639- Computer Vision**

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF ENGINEERING , AMRITA VISHWA VIDYAPEETHAM

COIMBATORE – 641112

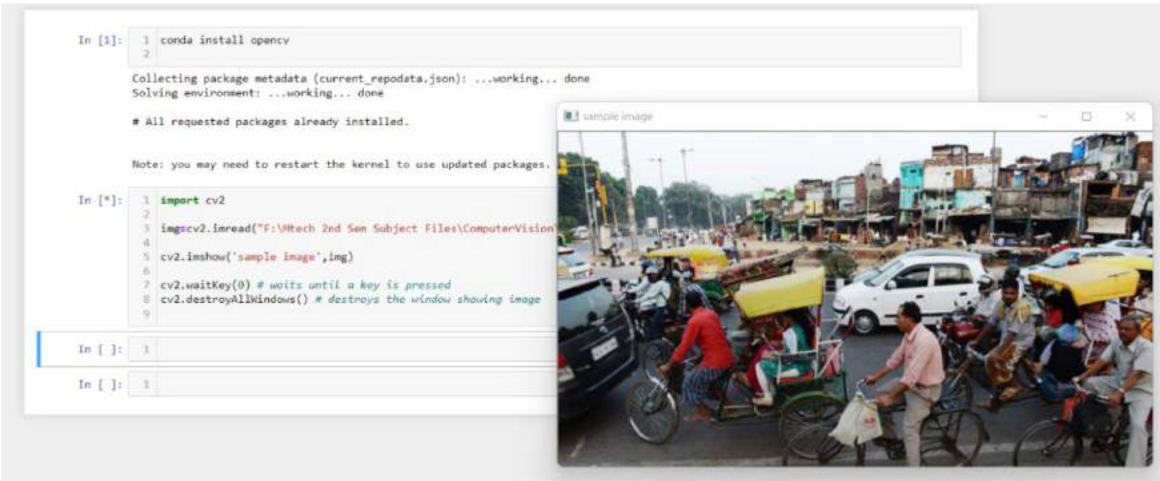
MAY 2022

INDEX:

➢ LAB 1: <u>Basic Image Processing:</u>

Question 1: **Read and display – image**

Output:

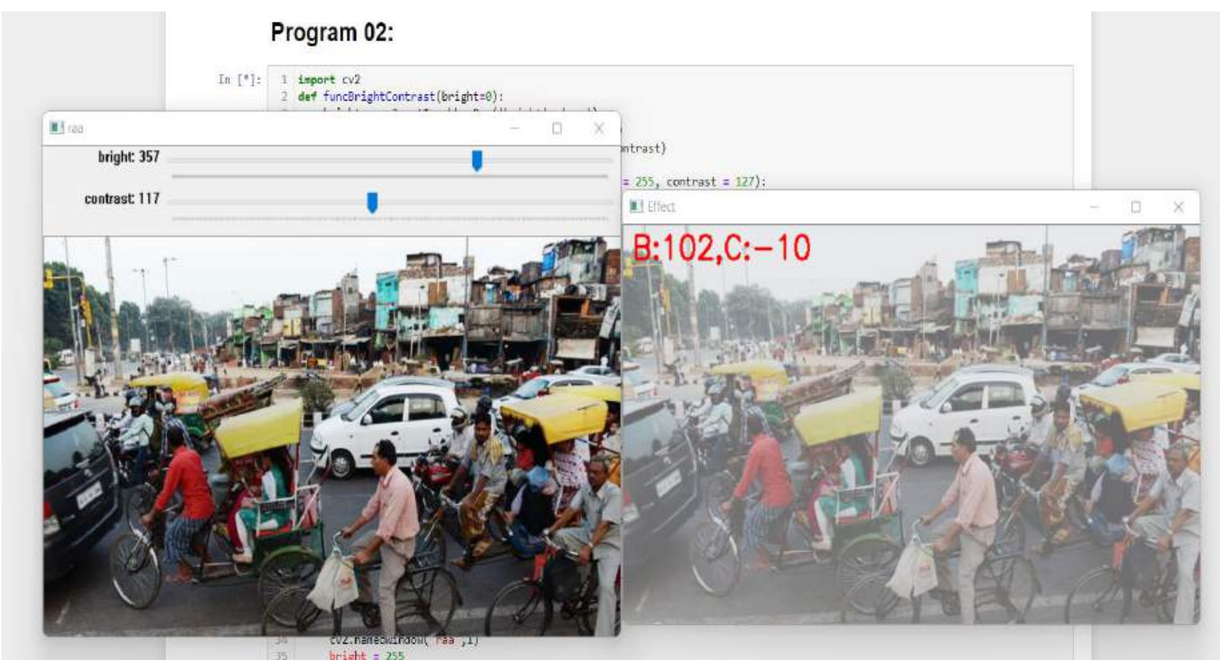


Inference:

Using OpenCV libraries, we read and display our input image.

Question 2: **Brightness and Contrast**

Output:



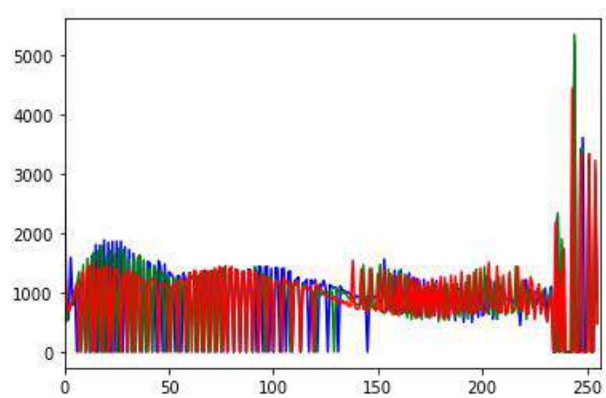
Inference:

Contrast refers to the brightness difference between various objects or parts of an image, whereas brightness refers to the overall lightness or darkness of an image.

Adding a positive constant to all of the image pixel values makes the image brighter.

Question 3: Histogram Equalization

Output:



Inference:

In image processing, a histogram is a crucial tool. It's a graphical representation of how data is distributed. An image histogram is a graphical depiction of a digital image's pixel intensity distribution. The x-axis represents the variable's possible range of values.

Question 4: Averaging filter

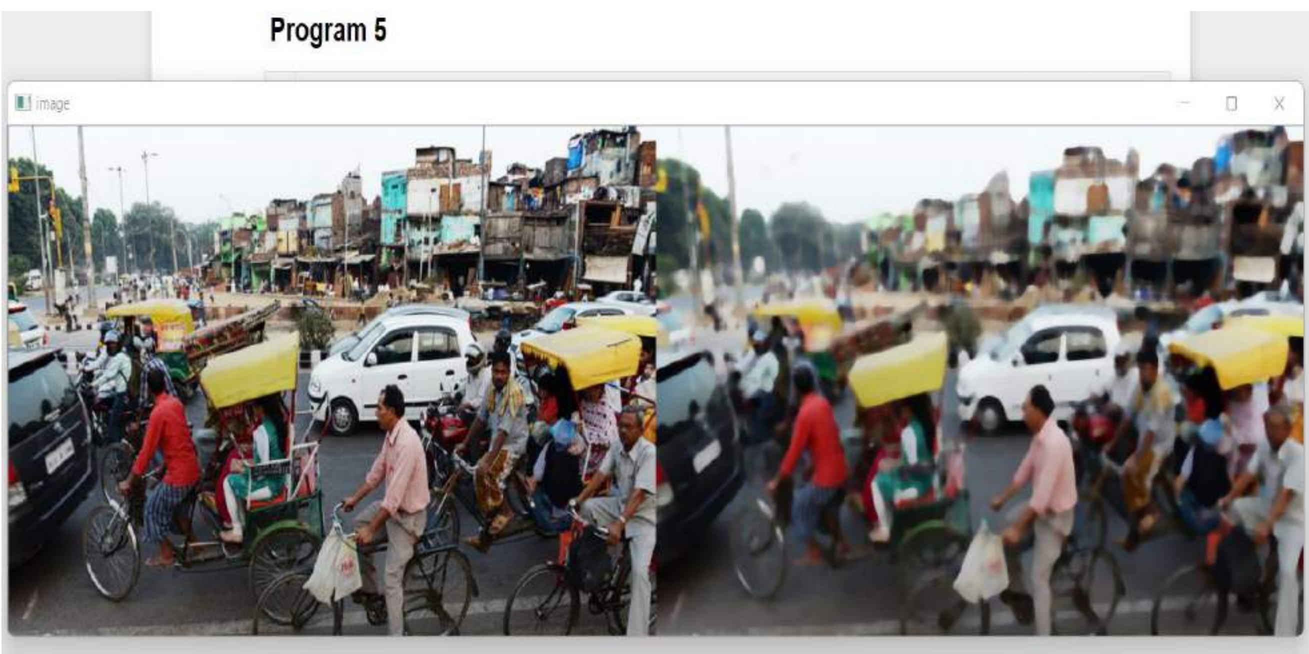
Output:

Inference:

Average filtering is a technique for smoothing photographs by lowering the intensity fluctuation between adjacent pixels. The average filter replaces each value with the average value of neighbouring pixels, including itself, as it moves through the image pixel by pixel.

Question 5: Median Filter

Output:

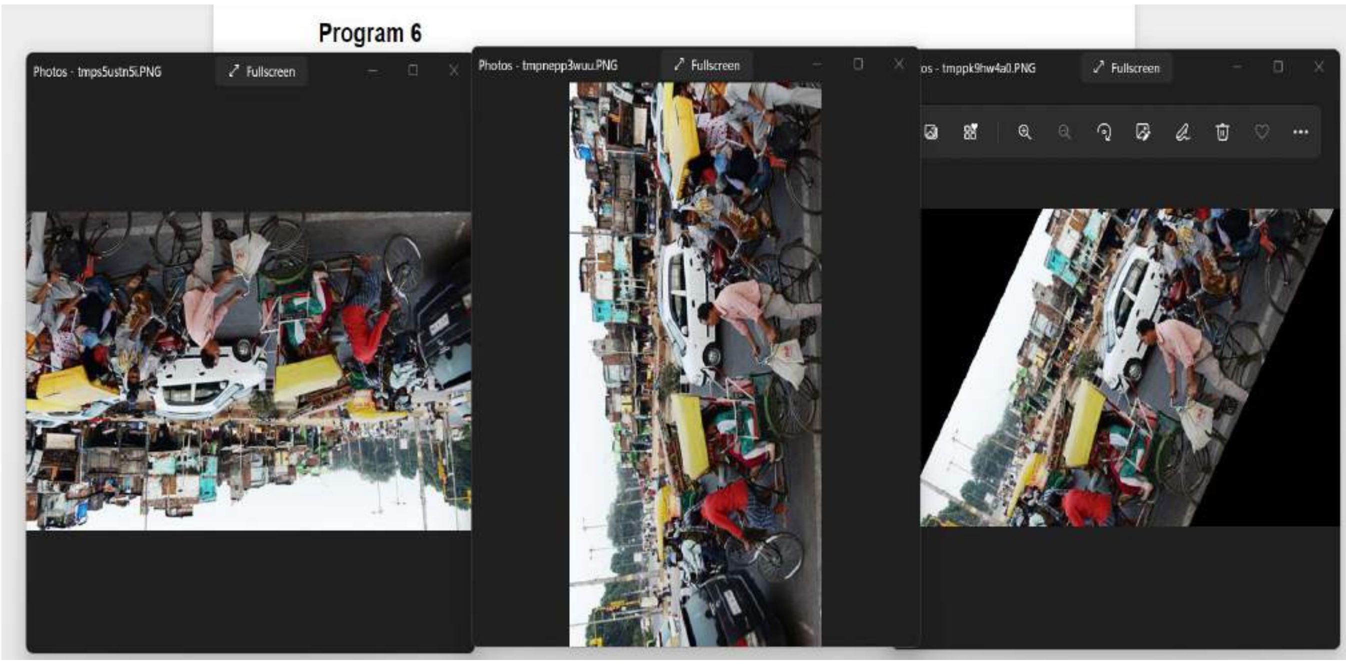


Inference:

The median filter is a non-linear digital filter that is commonly used to reduce noise from images.

Question 6: Image rotation

Output:

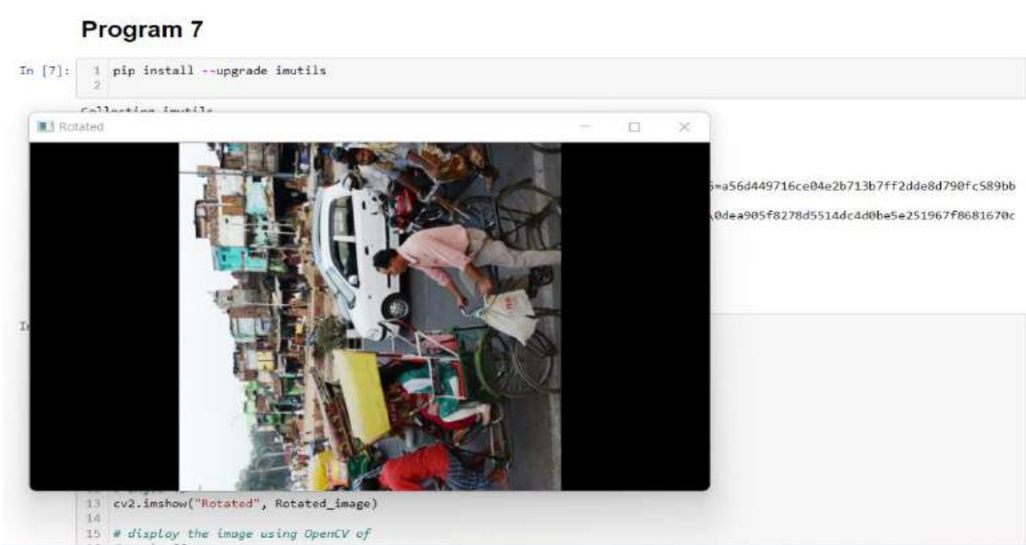


Inference:

We rotate the image.

Question 7: image rotation 2

Output:



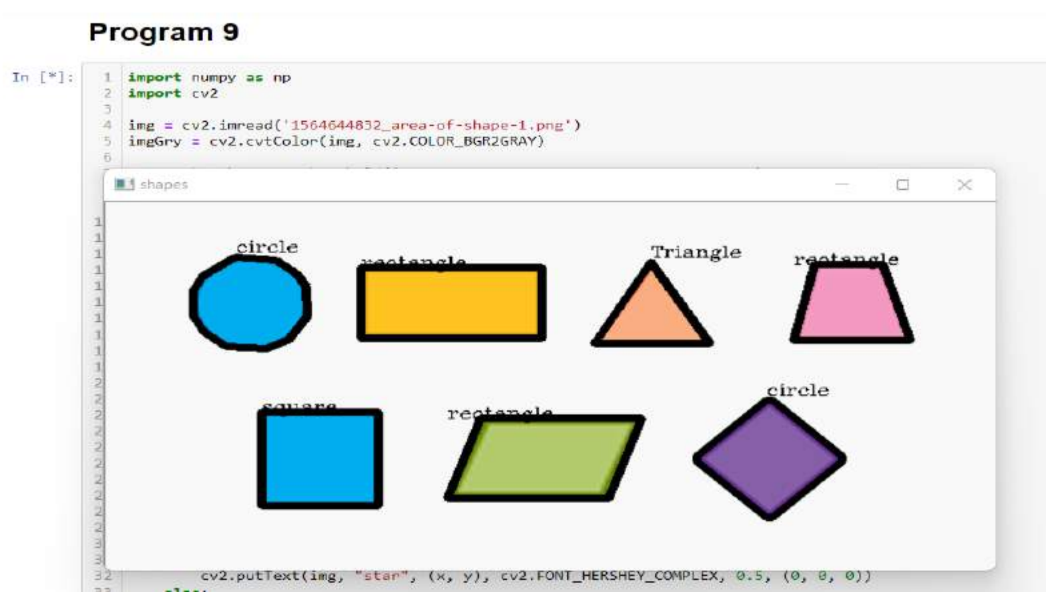
Question 8: **Video Rotation**

Output:

Inference: We rotate the video.

Question 9: **Shape Detection**

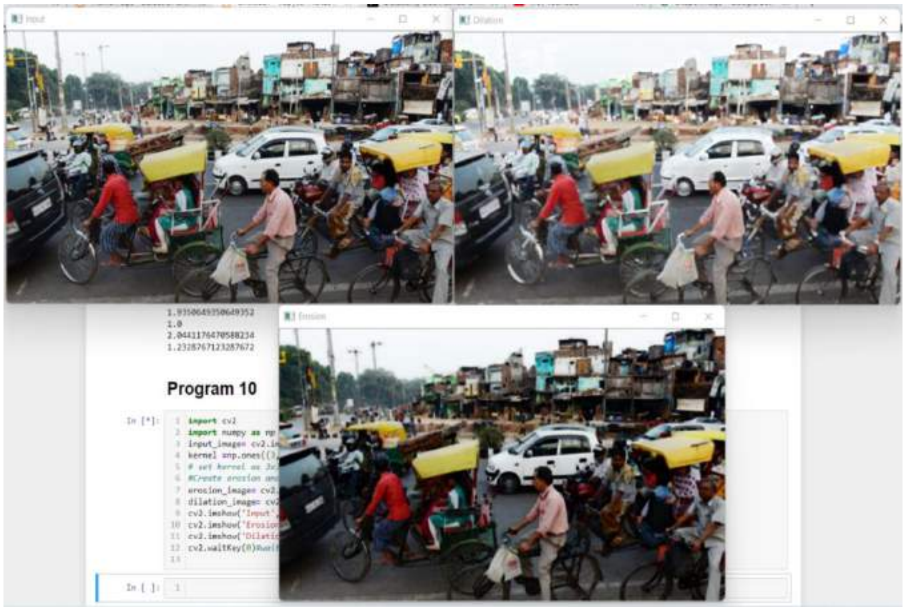
Output:



Inference:
To detect shapes in an image, use OpenCV's findContours() and approxPolyDP() functions. The OpenCV functions findContours() and approxPolyDP() can be used to discover forms in an image. We can recognize shapes based on how many corners they have.

Question 10: Erosion and Dilation

Output:



Inference:
In an image, dilation adds pixels to object boundaries, whereas erosion removes pixels from object boundaries. The size and shape of the structuring element used to process the image determines the number of pixels added or removed from the objects in the image.

Question 11: Inverting an image

Output:

Inference:
Light areas are mapped to dark, and dark areas are mapped to light in this image processing technique.

- **Inference:**

Image processing is a method for performing certain operations on an image, in order to get an improved image or extract useful information.

This is a type of signal processing where the input is an image and the output can be an image or features/characteristics related to that image.

➢ **LAB 2: <u>Feature Extraction and Matching:</u>**

Aim:

The aim of this evaluation is to learn the different feature extraction algorithms and feature matching algorithms and suggest which suitable approach can be applied for the data-set.

<u>Data-set Description:</u>

<u>Image Sample 1:</u>

<u>Image Sample 2:</u>

<u>Identify the following:</u> [Features required to be detected &Features that need to be matched]

1. Corners.
2. Scaled Image.
3. Rotational Image.
4. Affine transformation.

• Feature Extraction Algorithm Name:

1. Harris Corner Detection:

Working principle: Harris Corner Detector is a corner detection operator that is commonly used in computer vision algorithms to extract corners and infer features of an image.

- **Adv of Harris corner detection algorithm:**
  Commonly, Harris corner detector algorithm can be divided into five steps.

  - Color to grayscale.
  - Spatial derivative calculation.
  - Structure tensor setup.
  - Harris response calculation.
  - Non-maximum suppression.

- **Limitation:**
  The big drawback of Harris corner detector is the need to set a different threshold for each image in order to detect the most important interesting points

Output:

Out[3]: <matplotlib.image.AxesImage at 0x1de792f8610>

2. Scale-Invariant Feature Transform (SIFT):

Output:

3. Oriented FAST and Rotated BRIEF (ORB):

Working principle: ORB performs as well as SIFT on the task of feature detection (and is better than SURF) while being almost two orders of magnitude faster. ORB builds on the well-known FAST key-point detector and the BRIEF descriptor. Both these techniques are attractive because of their good performance and low cost.

- **ORB's Advantages:**

  - The addition of a fast and accurate orientation component to FAST.
  - The efficient computation of oriented BRIEF features.
  - Analysis of variance and correlation of oriented BRIEF features.
  - A learning method for decor-relating BRIEF features under rotational in variance, leading to better performance in nearest-neighbor applications.

Output:

```
Number of Keypoints Detected In The Training Image:  500
Number of Keypoints Detected In The Query Image:  234
```

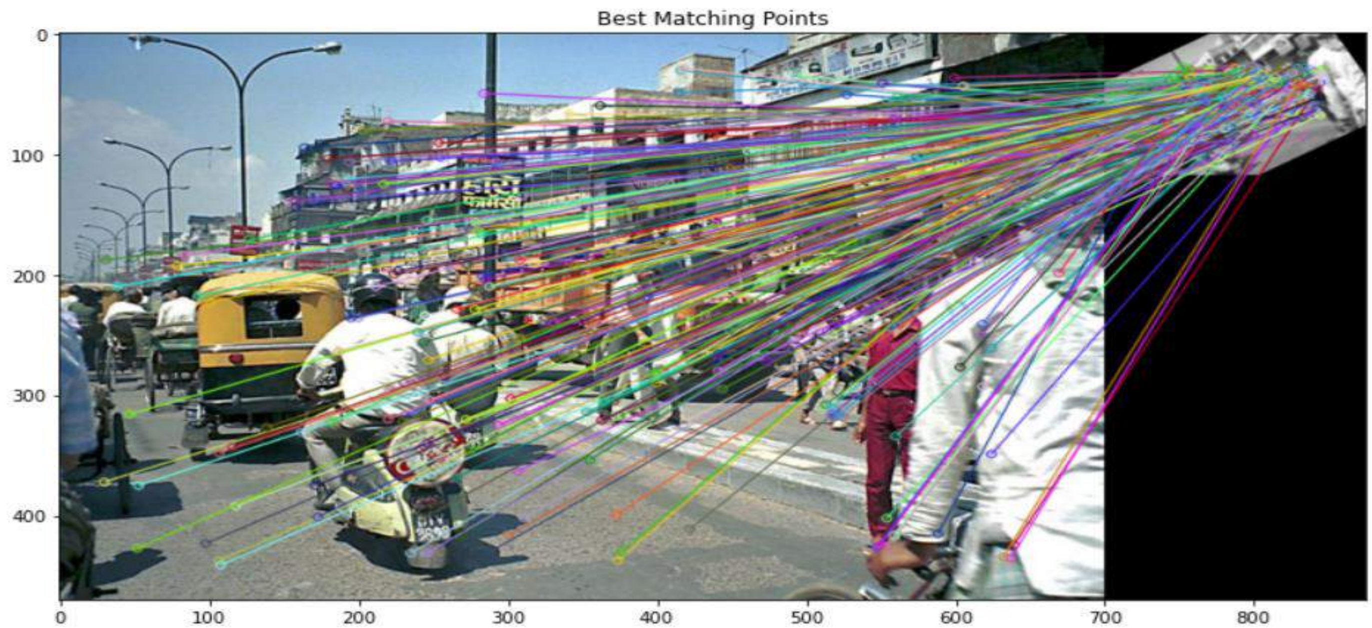
- ## Feature Descriptor Algorithm Name:

1. BRIEF(Binary Robust Independent Elementary Features):

Working Principle: Brief takes all key-points found by the fast algorithm and convert it into a binary feature vector so that together they can represent an object. Binary features vector also know as binary feature descriptor is a feature vector that only contains 1 and 0. In brief, each key-point is described by a feature vector which is 128–512 bits string.

Advantages of BRIEF:

Brief relies on a relatively small number of intensity difference tests to represent an image patch as a binary string. Not only is construction and matching for this descriptor much faster than for other state-of-the-art ones, but it also tends to yield higher recognition rates, as long as invariance to large in-plane rotations is not a requirement.



Number of Matching Keypoints Between The Training and Query Images: 238

2. **Oriented FAST and Rotated BRIEF (ORB):**

Number of Matching Keypoints Between The Training and Query Images: 113

- **Metrics used for Feature Matching:**

1. Brute Force Matcher:

we identify image features, distinctive points in our images. Second, we associate a descriptor for each feature from its neighborhood. Finally, we use descriptors to match features across two or more images. Afterwards, we can use the matched features for a variety of applications including state estimation, visual odometry, and object detection.

The simplest solution to the matching problem is referred to as "brute force feature matching", and is described as the following. First, define a distance function d that compares the descriptors of two features fi and fj, and defines the distance between them.

The more similar the two descriptors are to each other, the smaller the distance between them. Second, for every feature fi in image one, we apply the distance function d to compute the distance with every feature fj in image two. Finally, we will return the feature which we'll call fc from image two with the minimum distance to the feature fi in image one as our match.

This feature is known as the nearest neighbor, and it is the closest feature to the original one in the descriptor space. The most common "distance function" used to compare descriptors is the "sum of squared distances or SSD".

- **Sum of Squared Differences (SSD):**

$$d(f_i, f_j) = \sum_{k=1}^{D} (f_{i,k} - f_{j,k})^2$$

Limitation:

Brute force matching is suitable when the number of features we want to match is reasonable, but has quadratic computational complexity making it ill-suited as the number of features increases.

▪ Inference:

Edges and interest points in images convey a wealth of information about the image's content. They correspond to picture local regions and are crucial in many image analysis applications such as recognition, matching, reconstruction, and so on.

Feature extraction aids in the reduction of unnecessary data in a data set. Finally, reducing the data makes it easier to build the model with less machine effort, as well as speeding up the learning and generalization processes in the machine learning process.

Using a search distance method, feature matching finds equivalent features from two similar photos. The feature matching technique is used to either find or deduce and transfer properties from the source to the target image, with one image serving as the source and the other as the target.

• INTERACTIVE UI Feature Detection and Matching:

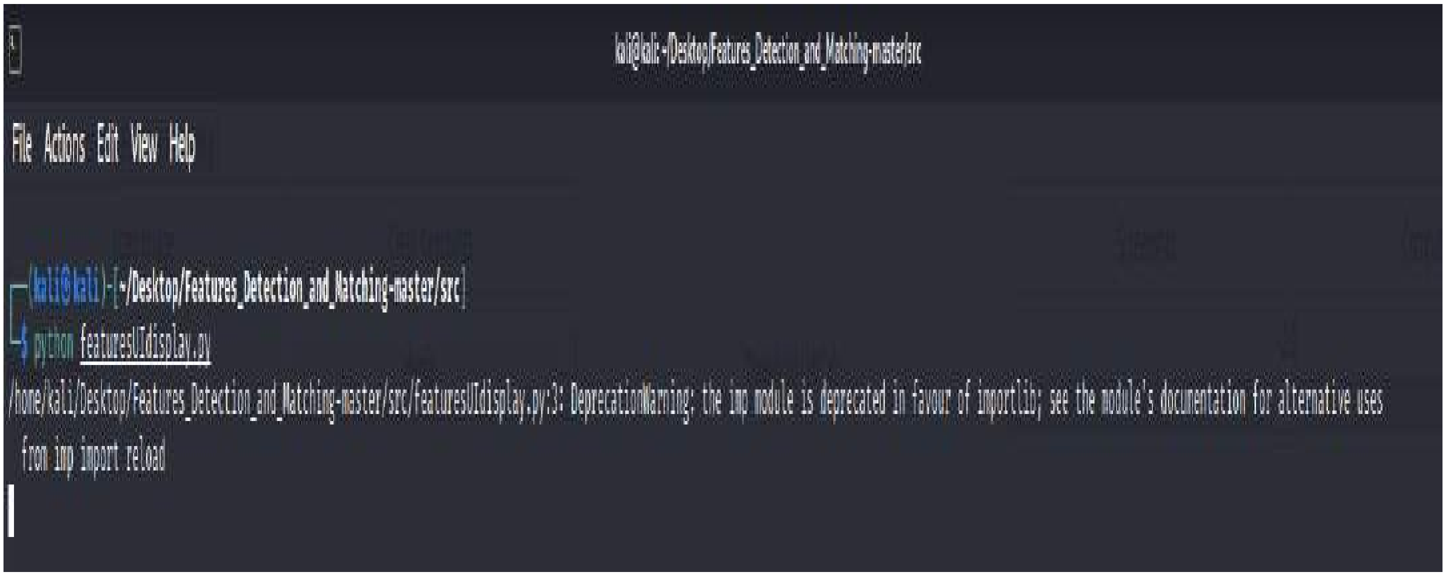
Note: I did this entire setup and execution in kali Linux.

1. **There are two python files: one python files is for creating/ adding feature algorithms and one more for user interface setup for using featuring algorithms.**

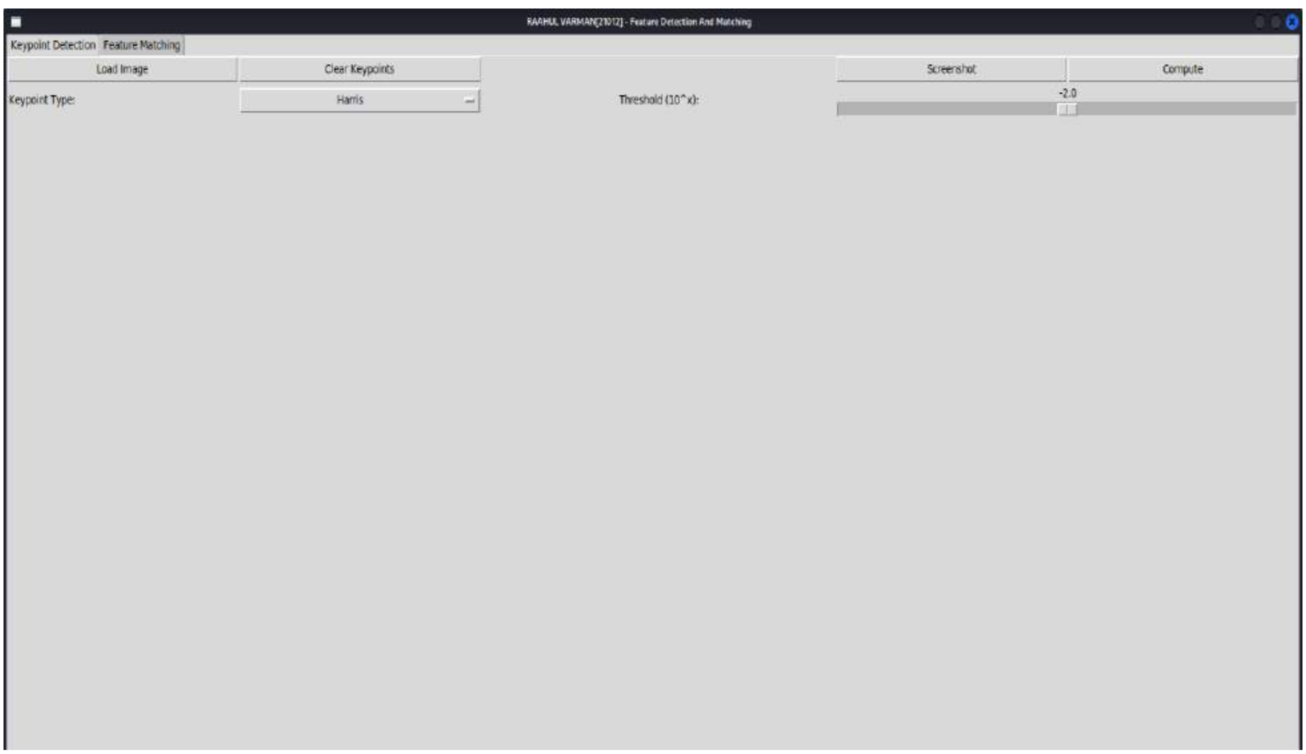
> Features.py

> FeaturesUIdisplay.py

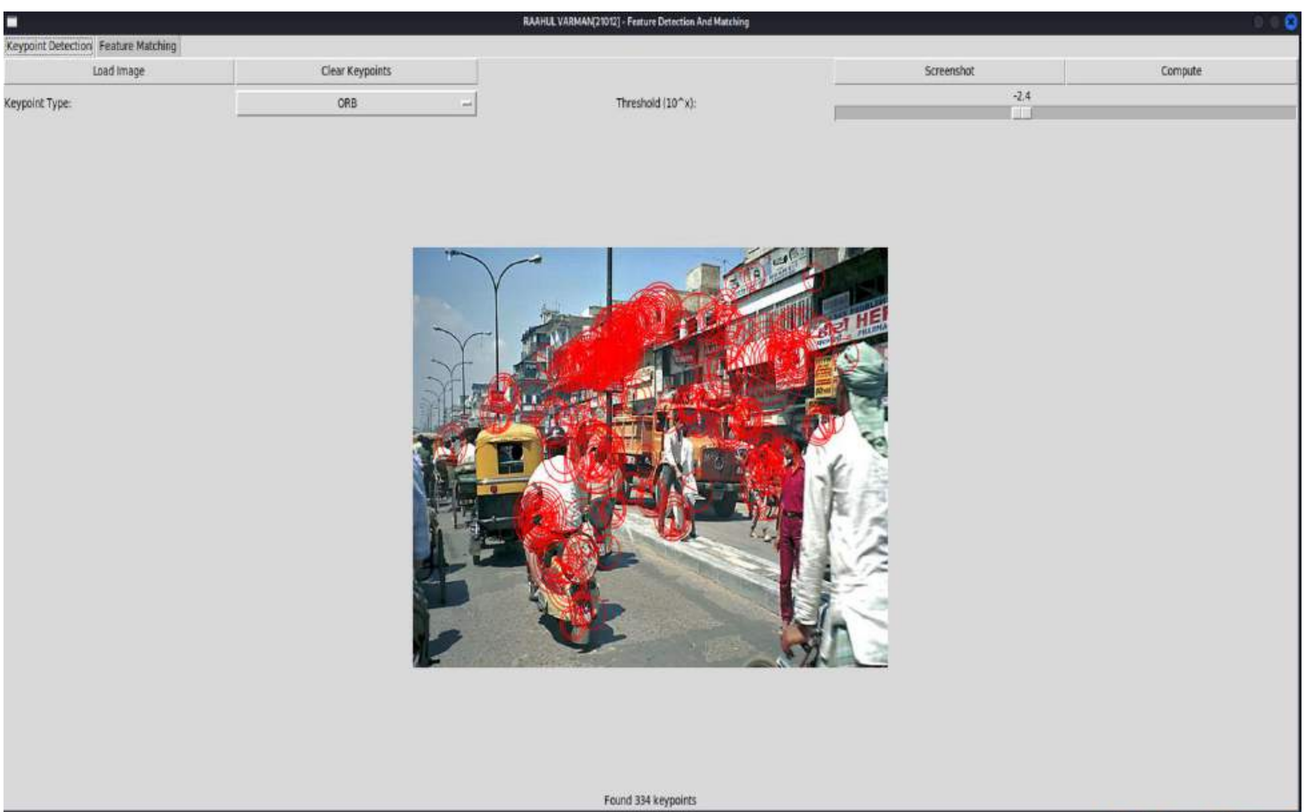
2. **Using command prompt we set the path to python file location to call the python file for execution.**
3. **We use -> python <filename.py>.**
4. **In our case we call -> python FeaturesUIdisplay.py and press Enter.**



5. Output will be loaded.



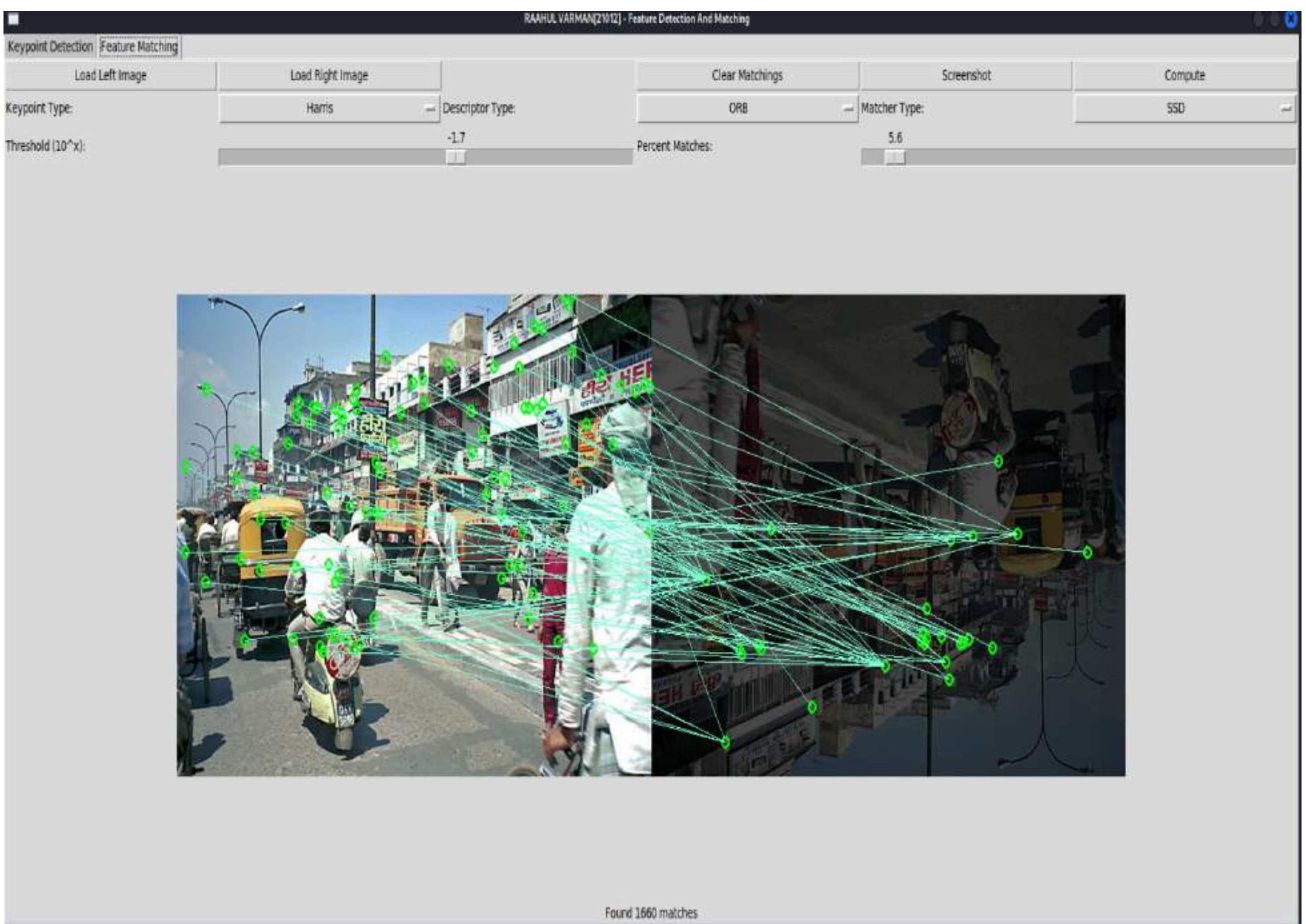
Key Feature Detection from my sample image:



Feature Matching:

Sample image: One image is original and second image is rotated and added some filter (noise).


Output:



➢ Lab 03 : <u>DATA POPULATION:</u>

- **<u>Title</u>:** “Model For Assisting Blind People By Detecting The Surrounding Objects/People/Activities With Voice Description.”

- **<u>Abstract</u>**:

There have been several systems designed to support visually-impaired people and to improve the quality of their lives. Unfortunately, most of these systems are limited in their capabilities. We decided to build a model which detects all possible surrounding things and guide them through voice description. Our main objective is to detect the obstacles in the person's pathway and warn them using voice description.

<u>Output</u>:

1		sharpness	contrast	mean_sat	std_sat	min_sat	max_sat	mean_bright	std_bright	min_bright	max_bright	colorfulness	hue_hist_1	hue_hist_2	hue_hist_3	shannon_entropy
2	0	1477.053	0.263872	102.68141	114.6114	255	0	108.9957952	102.864254	255	0	3.064395899	14	147	46.7053067	8.888408288
3	1	4543.997	0.259352	76.52696	56.62634	255	0	121.5755413	73.9336103	255	0	34.51540471	18	84	46.4239517	13.54138463
4	2	1740.611	0.331498	63.754197	72.50882	255	0	88.46784044	70.0220229	255	0	27.15711482	20	97	59.3380661	12.52376452
5	3	2769.083	0.225151	148.447594	75.33106	255	0	88.77501302	50.7272271	255	0	51.92119559	16	217	40.3020655	13.15744944
6	4	3121.815	0.170011	83.5869466	55.83494	255	0	109.3293522	58.807363	255	0	39.18421205	14	135	30.4319237	12.30113589
7	5	1124.724	0.219065	71.7033189	64.18686	255	0	148.6929084	51.0884041	255	0	51.2284675	14	84	39.2125942	12.18934376
8	6	189.3224	0.301611	36.784528	33.12133	228	0	119.732819	73.2426575	255	22	21.7024955	15	88	53.9883826	10.75647046
9	7	703.414	0.309182	103.23351	73.71014	255	0	96.65379041	64.0930649	255	0	71.31867585	20	10	55.343617	12.31874489
10	8	3127.229	0.280199	46.5957454	53.59137	255	0	120.1293099	82.1876149	255	0	29.60098717	12	545	50.1556087	12.30173162
11	9	3395.363	0.148805	95.2081711	45.46012	255	0	97.271381	49.6309649	255	0	37.78645975	17	48	26.6360416	13.3826206
12	10	5264.446	0.28279	72.0500928	60.18805	255	0	100.4784204	64.934584	255	0	37.94815978	20	179	50.6193212	13.91473623
13	11	2289.165	0.326508	77.2764225	61.2455	255	0	93.06748698	69.8960605	255	0	33.93625138	20	300	58.4449094	13.20427646
14	12	386.9361	0.437889	206.779769	69.73481	255	0	28.06677409	59.6336499	255	0	81.20001489	11	458	78.3820949	6.48821717
15	13	4327.952	0.216743	173.421423	62.11964	255	0	85.24505706	62.5925097	255	0	53.96807458	20	30	38.7969305	13.421823
16	14	2424.411	0.272996	71.3469596	39.32545	255	0	139.3976544	50.176872	255	0	43.94417881	7	137	48.8662689	12.46852461
17	15	2208.279	0.319879	41.5486719	33.90807	255	0	154.4046224	68.7067761	255	2	23.41485499	20	450	57.2583126	13.06359323
18	16	3520.001	0.196402	88.9670302	65.37251	255	0	162.0028835	65.203602	255	0	61.72770794	20	54	35.1560414	13.09650071
19	17	10275.09	0.173935	105.891663	54.41864	255	0	133.4038574	72.8623815	255	0	48.92667297	20	165	31.1344259	14.51299042
20	18	4774.396	0.166907	188.781817	73.71669	255	0	42.16796196	64.6732957	255	0	46.11209813	20	43	29.8762723	11.07037375
21	19	7404.018	0.306592	62.35179	62.28873	255	0	89.66992467	59.218121	255	0	33.68074231	20	108	54.8799577	13.67333008
22	20	2763.642	0.270204	83.5134433	60.85452	255	0	106.4424825	66.3760119	255	0	59.76832494	20	52	48.3664732	13.05296298
23	21	1231.514	0.321021	31.9909819	50.78078	255	0	173.5219036	66.4348485	255	0	25.93276655	12	350	57.4627434	10.16870035
24	22	3712.637	0.117291	112.472061	52.75428	255	0	97.17611979	54.9159267	255	0	57.14263084	18	59	20.9951626	12.40858599
25	23	5228.616	0.249572	46.9773387	43.05727	255	0	129.5107627	62.8979166	255	0	25.24885236	6	302	44.673394	12.95778131
26	24	2063.501	0.204231	19.805168	24.36841	212	0	170.2842187	63.4532253	255	22	16.50098474	10	421	36.5573314	12.13066847
27	25	5822.647	0.290135	58.6510968	48.84845	255	0	128.5952315	60.2586314	255	0	41.7120145	19	79	51.9341521	13.66920677
28	26	16637.04	0.256034	87.2983735	72.47033	255	0	62.39764458	75.973646	255	0	22.42663286	20	41	45.8301732	12.72732377
29	27	952.1554	0.23965	73.0964213	51.40065	255	0	103.9513867	54.2206188	255	0	37.07146248	18	125	42.8974141	11.93527667
30	28	4354.452	0.286668	44.9596029	53.44046	255	0	114.8688542	69.1182519	255	0	42.59907131	20	412	51.3135862	12.29873017
31	29	3273.591	0.229014	56.6622546	45.38545	255	0	104.1372137	61.2149407	255	0	21.40022073	3	541	40.993588	12.43377402

Feature Name	Purpose	Data Type of the feature
Sharpness	**Sharpness is a combination of two factors: resolution and acutance. Resolution is straightforward and not subjective. It's just the size, in pixels, of the image file. All other factors equal, the higher the resolution of the image—the more pixels it has—the sharper it can be. Acutance is a little more complicated. It’s a subjective measure of the contrast at an edge. There’s no unit for acutance—you either think an edge has contrast or think it doesn’t. Edges that have more contrast appear to have a more defined edge to the human visual system.**	Float64
Contrast	**The term contrast refers to the amount of color or grayscale differentiation that exists between various image features in both analog and digital images. Images having a higher contrast level generally display a greater degree of color or grayscale variation than those of lower contrast.** **Contrast enhancement processes adjust the relative brightness and darkness of objects in the scene to improve their visibility.**	Float64

Saturation	**The term saturation describes the depth or intensity of colour present within an image.**	Float64
Brightness	**The term saturation describes the depth or intensity of colour present within an image.**	Float64
Colorfulness	**Colorfulness is the "attribute of a visual perception according to which the perceived color of an area appears to be more or less chromatic.** **The colorfulness evoked by an object depends not only on its spectral reflectance but also on the strength of the illumination, and increases with the latter unless the brightness is very high**	Float64
Hue_Histogram	**An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance.**	Float64/int64
Shannon entropy	**In Image, Entropy is defined as corresponding states of intensity level which individual pixels can adapt. It is used in the quantitative analysis and evaluation image details, the entropy value is used as it provides better comparison of the image details.** **Shannon entropy as a measure of image information is extensively used in image processing applications.**	Float64

Lab 04: DEEP LEARNING BASED ON OPTICAL FLOW

OBJECTIVE:

Our aim is to estimate the motion of the image intensities over time in a video using optical-flow algorithms like Lucas-Kanade, Horn-schunck and Dense optical flow. i.e. to calculate velocity of points within the images, and to provide the estimation of where the points could be in the next image sequence in our shopping video dataset.

PERFORMANCE METRICS:

Metrics Name	Purpose	Formula	Expected Value
Point Rotational Error (PRE)	To prevent deviation (error) of a objects angular orientation from an expected, nominal, or commanded angle or displacement.	$PRE = \begin{cases} cos^{-1}\left(\frac{uu_{GT}+vv_{GT}}{\sqrt{u^2+v^2}\sqrt{u_{GT}^2+v_{GT}^2}}\right), \\ \quad \text{if } (u^2+v^2) \neq 0 \wedge (u_{GT}^2+v_{GT}^2) \neq 0 \\ \pi, \quad \text{if } (u^2+v^2) \oplus (u_{GT}^2+v_{GT}^2) = 0 \\ 0, \quad \text{if } (u^2+v^2) = 0 \wedge (u_{GT}^2+v_{GT}^2) = 0 \end{cases}$	Between 0.01 and 0.05
Angular Error (AE)	AE is very sensitive to small displacements, To determine the errors in angles.	$AE = cos^{-1}\left(\frac{uu_{GT}+vv_{GT}+1}{\sqrt{u^2+v^2+1}\sqrt{u_{GT}^2+v_{GT}^2+1}}\right).$	Between 1.00 – 2.00
End point Error (EPE)	It measures the distance between the endpoints of two optical flow vectors (u_0, v_0) and (u_1, v_1)	**sqrt((u_0 - u_1)^2 + (v_0 - v_1)^2)**	Should have small/min value
Normalized Euclidean Error (NEE)	IT gives the squared distance between two vectors where there lengths have been scaled to have unit norm. This is helpful when the direction of the vector is meaningful but the magnitude is not.	$NEE = \begin{cases} \frac{\sqrt{(u-u_{GT})^2+(v-v_{GT})^2}}{min\left((u^2+v^2),(u_{GT}{}^2+v_{GT}{}^2)\right)}, \\ \quad \text{if } min\left((u^2+v^2),(u_{GT}{}^2+v_{GT}{}^2)\right) > \varepsilon \\ \frac{\sqrt{(u-u_{GT})^2+(v-v_{GT})^2}}{\varepsilon}, \\ \quad \text{if } min\left((u^2+v^2),(u_{GT}{}^2+v_{GT}{}^2)\right) = 0 \end{cases}$	Value should be between 0 to 1
Enhanced Normalized Euclidean Error (ENEE)	Another way to get over EPE drawbacks, is to calculate the relative distance between ~E and G~T vectors and to use different normalization methods.	$ENEE = \begin{cases} \frac{\sqrt{(\Vert\vec{P_{GT}}\Vert)^2+\tau(\Vert\vec{N_{GT}}\Vert)^2}}{min\left((u^2+v^2),(u_{GT}{}^2+v_{GT}{}^2)\right)}, \\ \text{if} \quad min\left((u^2+v^2),(u_{GT}{}^2+v_{GT}{}^2)\right) > \varepsilon \\ \frac{\sqrt{(\Vert\vec{P_{GT}}\Vert)^2+\tau(\Vert\vec{N_{GT}}\Vert)^2}}{\varepsilon}, \\ \text{if} \quad min\left((u^2+v^2),(u_{GT}{}^2+v_{GT}{}^2)\right) = 0 \end{cases}$	Value should be between 0 to 1
Linear Projection Error (LPE)	This metric is a kind of mixture between AE and EPE, where the magnitude difference between (u;v) is added to the perpendicular distance between them. We can the angular distance between the two non-null vectors based on the perpendicular distance between both vectors.	$LPE = \begin{cases} \Vert\vec{GT}-\vec{E}\Vert + max\left(\Vert proj_{\vec{GT}}\vec{E}\Vert, \Vert proj_{\vec{E}}\vec{GT}\Vert\right), \\ \quad \text{if } \Vert\vec{GT}\cdot\vec{E}\Vert \neq 0 \\ \Vert\vec{GT}-\vec{E}\Vert + max\left(\Vert\vec{GT}\Vert, \Vert\vec{E}\Vert\right), \\ \quad \text{if } \Vert\vec{GT}\cdot\vec{E}\Vert \neq 0 \end{cases}$	Should have small/min value

Part-A :

1. **Analyze the video for slow, fast and medium frame rate based videos.**

   Dataset Drive Link : https://drive.google.com/drive/folders/1-1uNvgEhFRPvqYipvp6wQQhljHLi-KFP?usp=sharing

   Dataset: Shopping Mall Video.

   - **Original Video Length is 13 seconds.**
   - **Fast Video Length is 3 seconds.**
   - **Medium video length is 7 seconds.**
   - **Slow video length is 27 seconds.**

2. **Try out Horn and Schunck, Lucas Kanade, Dense Optical Flow algorithm and present the results in a Table form.**

➢ Lucas Kanade Algorithm:

➢ **Frame 1:** original video

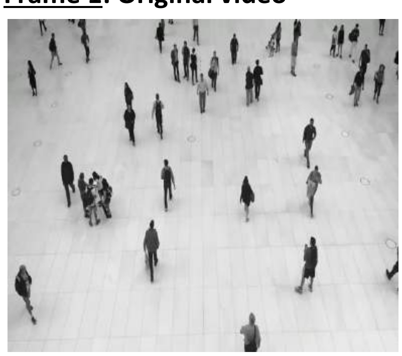

Output:

Original video		**Time taken for original video is 3mins 41secs**

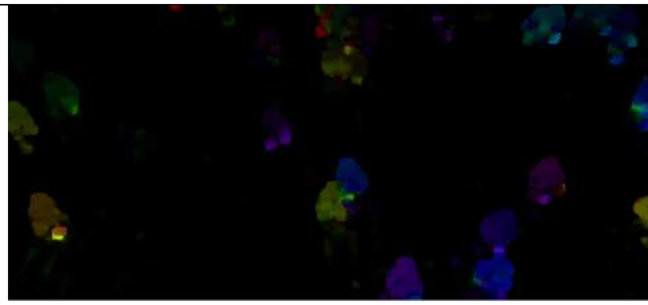
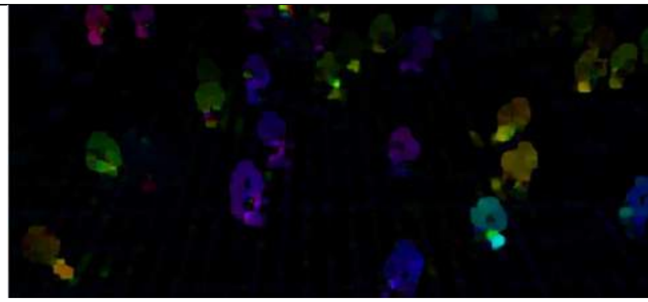
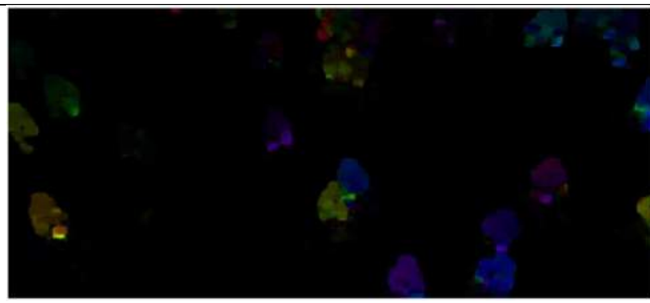
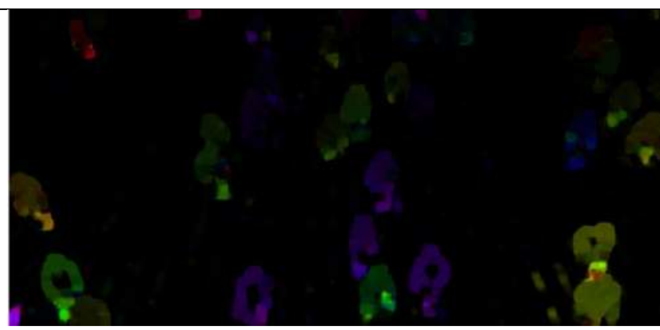
Fast video		**Time taken for fast video is 1mins 13secs**
Medium video		**Time taken for medium video is 2mins 9secs**
Slow video		**Time taken for slow video is 3mins 23secs**
Inference	**Lucas Kanade algorithm don’t works well in fast video footage, compare to orginal,medium and slow video footage.** **Algorithm was able to detect the corners and was able to estimate the where the flow could be in next image sequence.**	

➢ <u>Dense Optical Flow algorithm</u>:

<u>Frame 1</u>: Original video



Output:

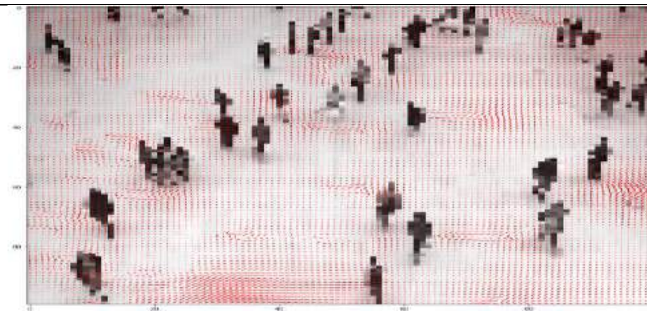
Original video		Time taken for original video is 3mins 53secs
Fast video		Time taken for fast video is 1mins 21secs
Medium video		Time taken for medium video is 2mins 12secs
Slow video		Time taken for slow video is 3mins 47secs
Inference	Dense Optical Flow algorithm works well in fast video footage and Slow video footage, compare to orginal and medium video footage. Dense optical flow able to compute optical flow vector for each and every pixel of each frame.	

➢ Horn and Schunck:

In horn and schunck we use Image(Each video frame) as the input to estimate the where the points could be in the next image sequence.

Frame 1: Original video

Original video (Frame 137)	
Fast video (Frame 35)	
Medium video (Frame 71)	

Slow video (Frame 50)	
Inference	**As the people in the video don’t have constant velocity movement the estimation in all the footage is little poor, which result in error at the boundaries.**

3. Write a Discussion section and present the findings in a paragraph.

In Horn and Schunck, as we know it can only handle object/person/things which has constant velocity. In our dataset we don’t have any constant movement kind of objects/person/things, which resulted in poor estimation.

In Lucas-Kanade, this algorithm works fine with slow motion footages/recordings. And it don’t work for fast motion footages/recordings. We can clearly see in our output, where we got better estimation of flow in next image sequence in slow motion footage then fast motion footage. In fast motion footage, the estimation was mismatched and uneven.

In Dense Optical Flow, as it compute optical flow vector for each and every pixel of each frame. We where able to estimate the flow in all the footage clearly. Even though the computational time is much higher then other two algorithm. Its give accurate result and denser result which is suitable for application like motion and video segmentation.

From comparison, Dense optical flow was able to give better estimation to describe image motion.

Part-B:

1. Introduce the noise in the video and complete the following:

a. Apply Filtering in the video:

Dataset Drive Link:
https://drive.google.com/drive/folders/1IL6MeTv1drSc_cggQ7WPeSsvg4EvJB2N?usp=sharing
Dataset: Shopping Mall Video.

- **Original Noise Video Length is 13 seconds.**
- **Fast Noise Video Length is 3 seconds.**
- **Medium Noise video length is 7 seconds.**
- **Slow Noise video length is 27 seconds.**

b. Try out Horn and Schunck, LucasKanade, Dense Optical Flow algorithm:

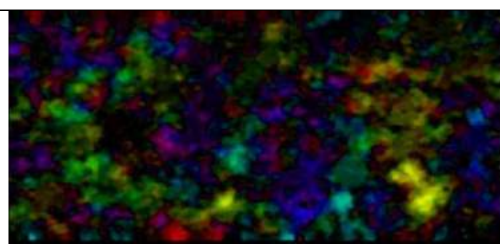
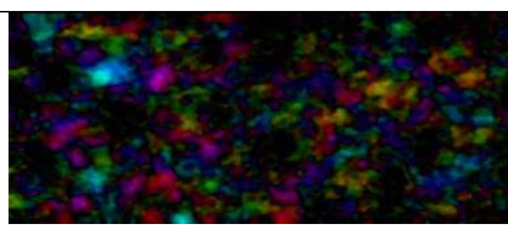
➢ <u>**Lucas Kanade:**</u>

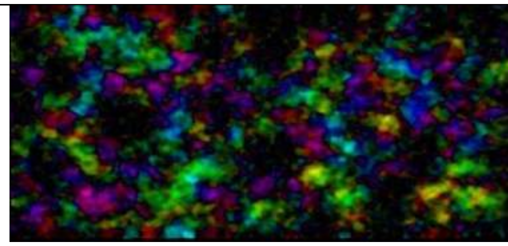
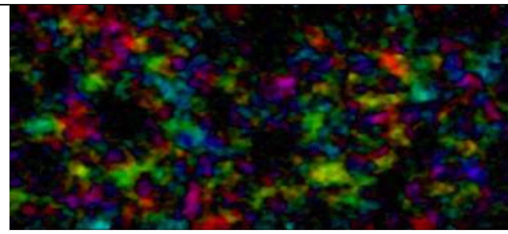
<u>**Frame 1:**</u>

Original video		**Time taken for original video is 2mins 49secs**
Fast video		**Time taken for original video is 3mins 14secs**

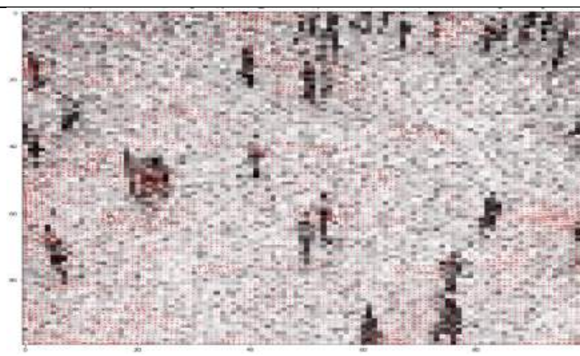
Medium video		**Time taken for original video is 2mins 40secs**
Slow video		**Time taken for original video is 2mins 43secs**
Inference	**Lucas Kanade algorithm didn't work well in any of the noise footage properly.** **Algorithm was not able to detect the corners and was not able to estimate the where the flow could in next image sequence**	

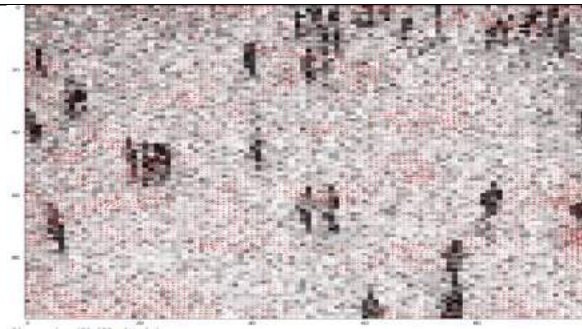
➢ **<u>Dense Optical Flow</u>:**

Original video		**Time taken for original video is 10mins 9secs**
Fast video		**Time taken for original video is 2mins 43secs**

Medium video		**Time taken for original video is 5mins 20secs**
Slow video		**Time taken for original video is 20mins 51secs**
Inference	**Dense Optical Flow algorithm didn't work well in any of the noise video footage.** **Dense optical flow able to compute optical flow vector for each and every pixel of each frame.**	

- **<u>Horn and Schunck</u>:**

Original video	
Fast video	

Medium video	
Slow video	
Inference	**Horn and Schunck Algorithm didn't work for noise video footage. The estimation is very poor.**

C. Write a Discussion section and present the findings in a paragraph:

In Horn and Schunck, as we know it can only handle object/person/things which has constant velocity. In our dataset we have noise also, which made the estimation very poor or not estimated properly the flow of motion in the image sequence.

In Lucas-Kanade, this algorithm didn't work well for the noise dataset. The estimation was very poor. It was unable to detect the flow properly. Which gave an poor estimation of where the flow could be in next image sequence. in

In Dense Optical Flow, as it compute optical flow vector for each and every pixel of each frame. We where not able to estimate the flow in all the footage clearly. And the computational time is much higher then other two algorithm. It didn't give accurate result for noise dataset.

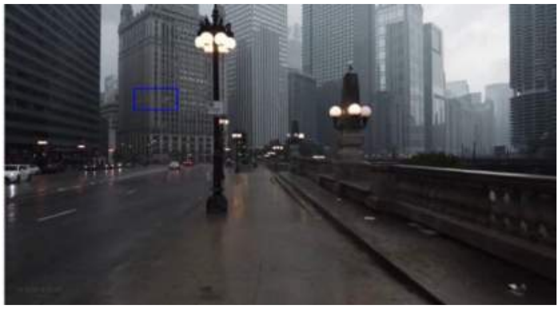
From comparison, None of the algorithm worked for the dataset with noise.

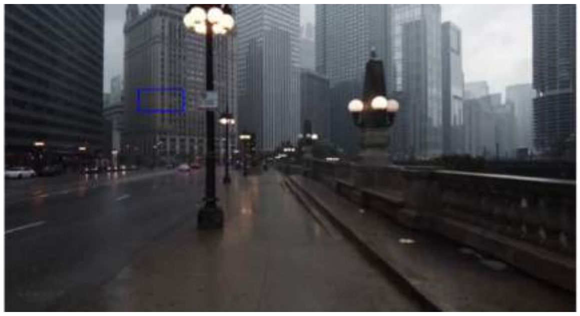
MEANSHIFT AND CAMSHIFT:

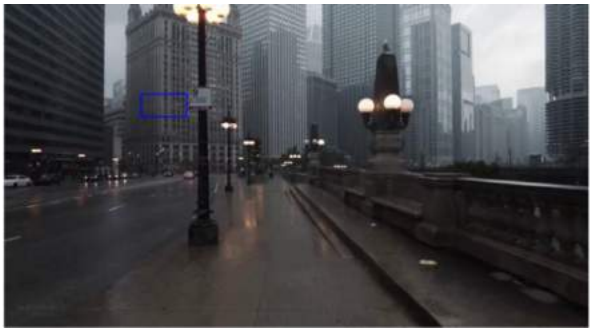
Dataset: https://drive.google.com/drive/folders/1GA449sOZ4AzdH80C3b1f3A83Vla0MCSq?usp=sharing

MEANSHIFT-

Output:



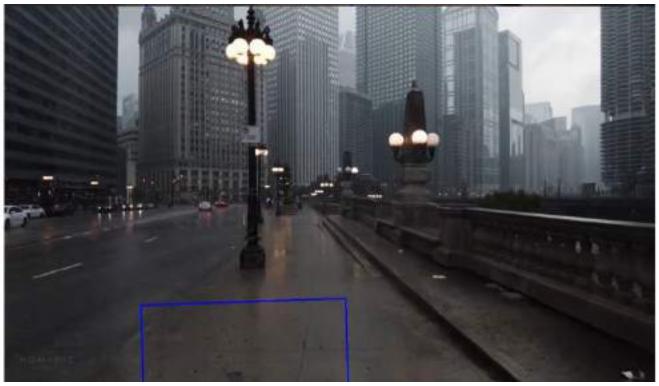


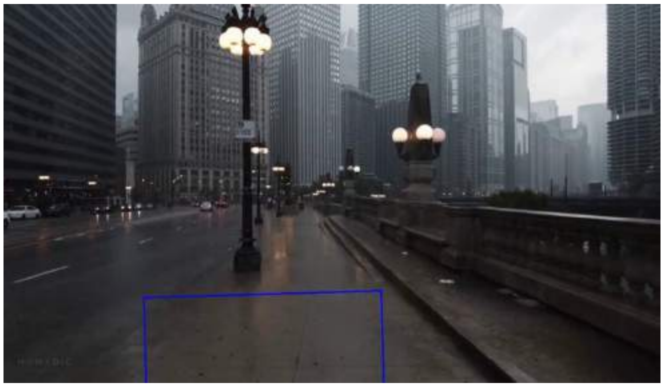


- The mean shift method is a statistical concept that has to do with clustering. The mean shift technique, like most other clustering algorithms, looks for regions in the data set with a large concentration of data points, or clusters.
- The Mean Shift clustering algorithm is an unsupervised clustering algorithm that organizes data without using labeled data to train it. The Mean Shift clustering algorithm is hierarchical in nature, which means it is based on a cluster hierarchy.
- The Mean Shift Algorithm is a clustering algorithm that uses the highest density points or mode value as the major parameter for machine learning development. It's a supervised machine learning algorithm that's not supervised. Kernel Density Estimation (KDE) is the basis for the algorithm. The mode searching algorithm is another name for it. The Kernel is involved in mathematical calculations including data point weighting. The flat kernel and the Gaussian Kernel are two of the most common kernel functions linked with the mean Shift Algorithm. Computer vision and picture segmentation are the most common applications for this approach.

CAMSHIFT:

Output:





- The CAMShift algorithm is based on the mean shift algorithm, which is responsible for locating the centre of the object's probability distribution. The primary distinction is that CAMShift adapts to the size of the search window, which is useful when object sizes change as they move closer or farther away from the camera.

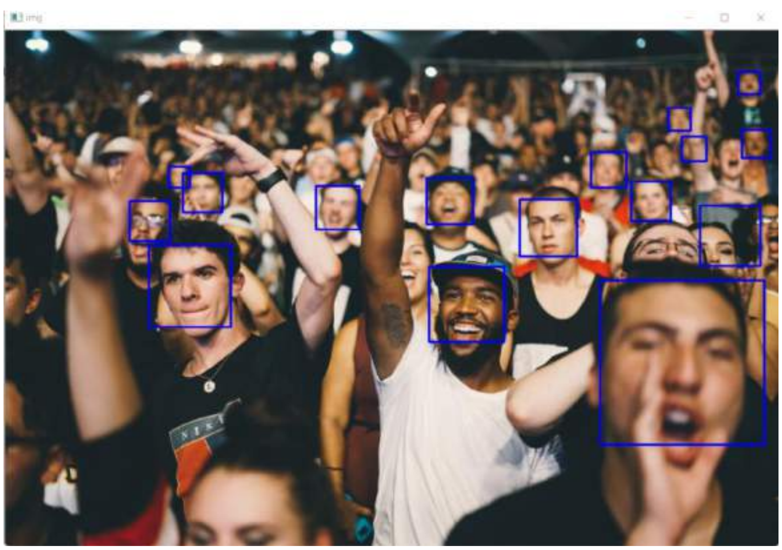
LAB 4b: FACE DETECTION:

Aim: The primary aim of face detection algorithms is to determine whether there is any face in an image or not.

Face identification with **Haar cascades** is a machine learning strategy that involves training a cascade function with a collection of input data. We'll use the face classifier from OpenCV, which already has several pre-trained classifiers for faces, eyes, smiles, and so forth.

Input:

Output:



Face identification with Haar cascades is a machine learning strategy that involves training a cascade function with a collection of input data. We'll use the face classifier from OpenCV, which already has several pre-trained classifiers for faces, eyes, smiles, and so forth.

- Accuracy is obtained for the Haar cascade is **96.24%.**

LAB 3B: (**Data population-customized**)

Data Description:

- Customized Dataset Drive Link:

https://drive.google.com/drive/folders/1oE2vpCSvuDcnnDGpuIBO-m3LIpYI9iD_?usp=sharing

- Training Sample Size : 735.
- Testing Sample Size : 183.

Sample Images:

- No. of Positive image: 558
- No. of Negative Image: 336

The dataset contain common scenario scene images where it can be used for object/human detection and tracking.

Challenges in the image dataset:

1. Some of the image has noise and unclear.
2. shadows are there.
3. Image perspective is different, which make difficult yo algorithm for training.
4. Luminous issues in many.
5. Occlusion problem.

Feature Extraction from the customized dataset:

	sharpness	contrast	mean_sat	std_sat	min_sat	max_sat	mean_bright	std_bright	min_bright	max_bright	colorfulness	hue_hist_1	hue_hist_2	hue_hist_3	shannon_entropy
0	4399.562465	0.05624065506	189.7331665	45.8707028	255	0	119.2013514	36.72508709	255	0	52.11309331	7	44	11.55708262	13.34471705
1	8244.84231	0.2502985557	67.79518107	80.96958063	255	0	89.34696254	47.28009675	255	0	28.89028428	20	206	44.80344147	13.73528963
2	10104.18687	0.2041440586	82.78518926	73.21670064	255	0	139.9614217	63.81622867	255	0	49.87082042	20	34	36.54178665	14.25644132
3	210.8174242	0.3232985518	75.60708709	49.40484661	255	0	81.00927928	46.38823288	252	1	31.20198661	15	77	57.87045868	10.84089064
4	36883.12614	0.2756379285	118.5060256	81.80930626	255	0	103.924806	83.08481448	255	0	44.9155765	20	25	49.33918921	14.87334389
5	1727.319781	0.1223710878	60.35980143	40.81663439	255	0	138.4005046	54.24219207	254	0	22.81212575	14	112	21.90442113	11.02849201
6	4482.415403	0.269608798	67.15509463	58.00191288	255	0	109.8017386	66.6692088	255	0	23.80018552	15	600	48.25961685	13.6996456
7	3652.491726	0.2726632894	83.05838193	61.13382937	255	0	86.15143229	57.85118749	255	0	30.98082698	20	226	48.80673089	13.08221734
8	1253.834837	0.1334883967	37.80305029	27.19802348	255	0	119.8712221	87.28505335	243	0	21.436422	8	18	23.59442301	10.3596116
9	3838.868528	0.3209311169	47.72775608	55.61012716	255	0	136.0521918	82.53492928	255	0	26.96934758	20	118	57.44688882	12.67476824
10	6483.192867	0.2061770083	167.862011	82.37404691	255	0	94.39414706	61.48590848	255	0	84.81393526	18	108	36.90668448	13.78474528
11	1784.462613	0.1028389644	18.19059207	21.96257207	255	0	176.5591904	34.52879447	255	0	18.25218673	10	246	18.40815674	10.84361264
12	785.8834151	0.2250288483	163.0368287	49.84034698	255	0	60.88378133	41.806865	251	0	47.83265026	14	138	40.44128354	11.89447577
13	1320.169871	0.201218638	37.56804362	26.55582011	255	0	132.3368945	64.72354751	255	0	29.75871278	12	27	36.01813621	12.47975655
14	1626.904049	0.2072428408	176.2647383	94.02431705	255	0	106.5449479	91.13790617	255	0	85.04494565	17	132	36.68198282	11.30537835
15	1640.187494	0.2818685785	39.72861654	16.76846347	255	0	183.7469804	23.09601398	255	0	20.34233075	6	111	46.87447555	10.21086618
16	2684.876105	0.2860980069	63.71386758	69.53047018	255	0	123.7406482	64.35751897	255	0	68.2179486	19	180	61.21164306	12.87216044
17	340.0229881	0.2585699338	26.78777669	31.54578173	255	0	129.80625	32.67951723	254	0	20.84594326	20	17	45.82422615	11.37128625
18	389.2339991	0.1305353813	43.66540267	21.41577408	255	0	62.08117333	48.25856053	255	0	9.653067611	10	48	23.36563325	10.82698848
19	4543.967416	0.2503516852	76.52696	56.62634123	255	0	121.3755413	73.93361026	255	0	34.31540471	18	84	46.42385165	13.54138463
20	1740.611036	0.331497576	63.75419699	72.50881652	255	0	88.46784044	70.02202288	255	0	27.15711462	20	97	59.3380661	12.52376452
21	3273.580763	0.2290144578	56.65225458	45.38844823	255	0	104.1372137	61.21484069	255	0	21.40022073	3	541	40.99358795	12.43377402
22	2769.083404	0.2251612041	148.4475944	75.33106108	255	0	88.77601302	50.72722709	255	0	51.92119659	16	217	40.30206554	13.15744944
23	3121.814606	0.170010747	83.58694661	55.83494147	255	0	109.3293522	58.80736301	255	0	39.18421208	14	135	30.43192371	12.30119680
24	1124.723621	0.2190647718	71.70331894	64.18665732	255	0	148.8929084	51.08840407	255	0	51.2284675	14	84	39.21259416	12.18934376
25	188.3223723	0.3016110761	36.78452789	33.12132813	228	0	119.732819	73.24285748	255	22	21.7024965	15	88	53.98838262	10.75547046
26	1477.083201	0.2638717895	102.6814096	114.8113723	255	0	106.9987952	102.8642544	255	0	3.064395890	14	147	46.70630674	8.888408288
27	703.4140106	0.3091822179	103.2336102	73.7101406	255	0	96.68379041	64.09306492	255	0	71.31867585	20	10	55.343617	12.21874489
28	4354.452187	0.2866880793	44.95960286	53.44045637	255	0	114.8688542	69.11825192	255	0	42.59907131	20	412	51.31358519	12.29873017
29	952.155426	0.2396603579	73.09642133	51.40084715	255	0	103.9513867	54.22061875	255	0	37.07146248	18	125	42.89741406	11.93527887
30	16837.03952	0.2560344872	87.29837349	72.47033209	255	0	62.39764458	75.97364699	255	0	22.42663286	20	41	45.8301732	12.72732377
31	4774.366445	0.1669065493	188.7818173	72.71669404	255	0	42.16796106	64.67329672	255	0	46.11209813	20	43	29.87627233	11.07037375
32	7404.019332	0.3066919424	62.35178999	62.29872906	255	0	89.66992467	59.21812101	255	0	33.69074231	20	108	64.87995769	13.67233009
33	1231.614119	0.3210209126	31.89098193	50.78078497	255	0	173.5319036	66.43484853	255	0	25.93279655	12	350	57.46276335	10.16970038
34	2063.601259	0.2042309016	19.806168	24.3684147	212	0	170.2842167	63.4532253	255	22	16.80098474	10	421	36.55733139	12.13066847
35	5228.615779	0.2405720333	46.97733867	43.06726981	255	0	129.5107627	62.89701680	255	0	25.24885236	6	302	44.67339305	12.55778131
36	2783.642047	0.2702037612	83.61344332	60.86452024	255	0	106.4424526	66.37601192	255	0	59.78832404	20	62	48.36847326	13.05296298
37	3712.636803	0.1172914114	112.4720605	52.75428134	255	0	97.17611979	54.91592866	255	0	67.14263084	18	59	20.99618264	12.40858699

Feature Name	Purpose	Data Type of the feature
Sharpness	**Sharpness is a combination of two factors: resolution and acutance. Resolution is straightforward and not subjective. It's just the size, in pixels, of the image file. All other factors equal, the higher the resolution of the image—the more pixels it has—the sharper it can be. Acutance is a little more complicated. It's a subjective measure of the contrast at an edge. There's no unit for acutance—you either think an edge has contrast or think it doesn't. Edges that have more contrast appear to have a more defined edge to the human visual system.**	Float64
Contrast	**The term contrast refers to the amount of color or grayscale differentiation that exists between various image features in both analog and digital images. Images having a higher contrast level generally display a greater degree of color or grayscale variation than those of lower contrast.** **Contrast enhancement processes adjust the relative brightness and darkness of objects in the scene to improve their visibility.**	Float64
Saturation	**The term saturation describes the depth or intensity of colour present within an image.**	Float64
Brightness	**The term saturation describes the depth or intensity of colour present within an image.**	Float64
Colorfulness	**Colorfulness is the "attribute of a visual perception according to which the perceived color of an area appears to be more or less chromatic.** **The colorfulness evoked by an object depends not only on its spectral reflectance but also on the strength of the illumination, and increases with the latter unless the brightness is very high**	Float64
Hue_Histogram	**An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance.**	Float64/int64
Shannon entropy	**In Image, Entropy is defined as corresponding states of intensity level which individual pixels can adapt. It is used in the quantitative analysis and evaluation image details, the entropy value is used as it provides better comparison of the image details.** **Shannon entropy as a measure of image information is extensively used in image processing applications.**	Float64

LAB 5: Data Analysis and Modeling

- Dataset Used for the project use case is: "COCO DATASET". [Link]

The "Common Objects In Context" (COCO) dataset is a collection of challenging, quality datasets for computer vision, mostly using cutting-edge neural networks.

COCO is a large-scale object detection, segmentation, and captioning dataset.

- COCO has several features:
  - Object segmentation.
  - Recognition in context.
  - Super pixel stuff segmentation.
  - 330K images (>200K labeled).
  - 1.5 million object instances.
  - 80 object categories.

Type of Challenge addressed in the dataset:

When developing Object Detection and Image Segmentation models, common data issues include:

1. Dimensions and aspect ratios of images (especially dealing with extreme values).
2. Identifies the composition's imbalances, bounding box dimensions, and aspect ratios (for instance a lot of small objects)
3. Not the right data preparation for your dataset.
4. Modeling strategy is not consistent with the data.

General data quality:

1. Acquire a basic sense of a dataset and visually examine it.
2. Ensure sure it isn't damaged and doesn't have any glaring distortions.

3. Verify that all of the files can be read; you don't want to discover this during training.

Here is to visualize as many pictures as possible:

- **Plot them in a jupyter notebook using matplotlib**:

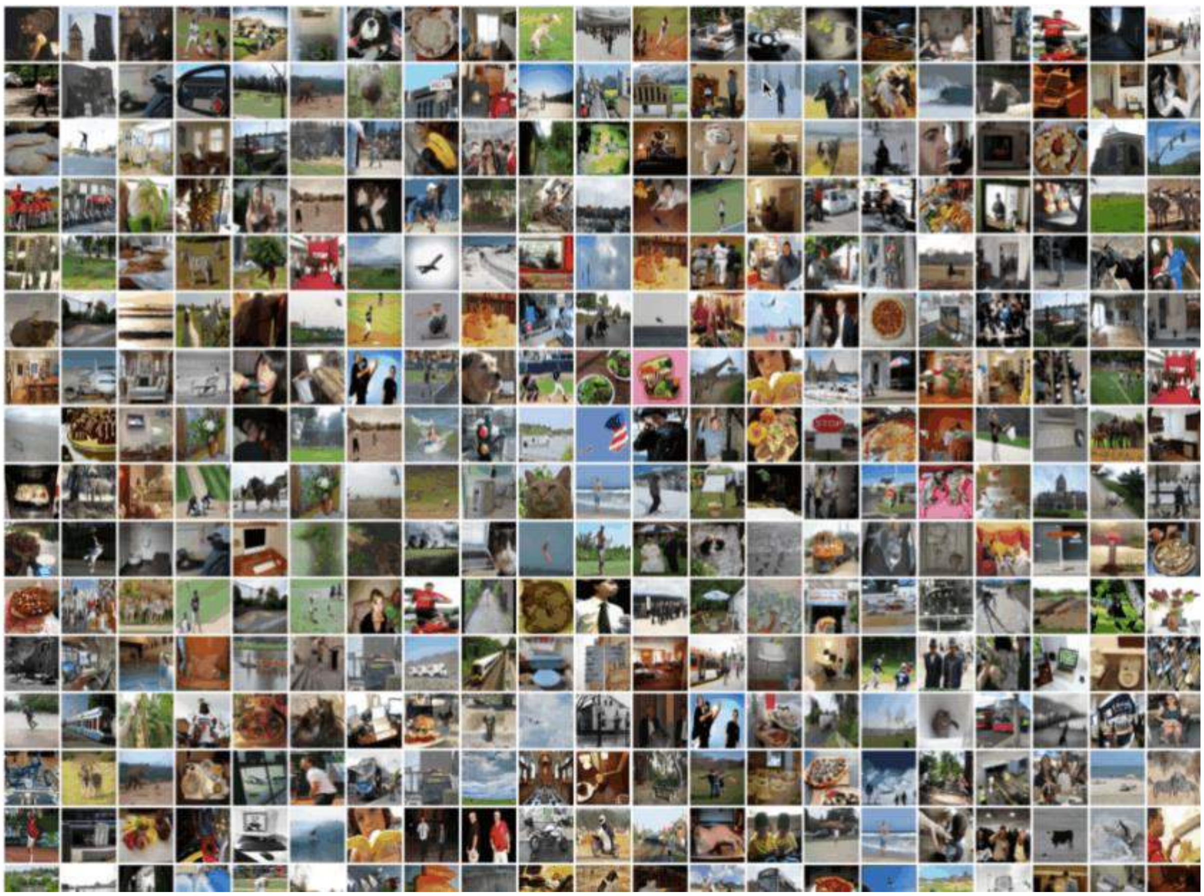
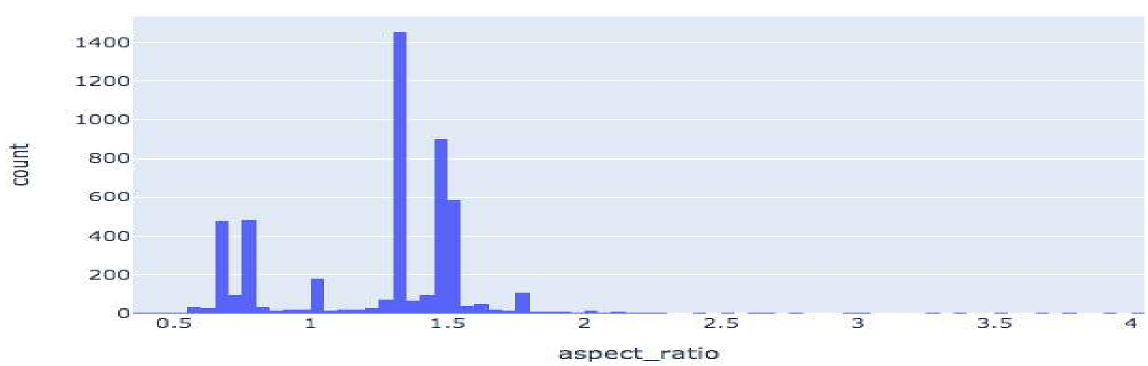


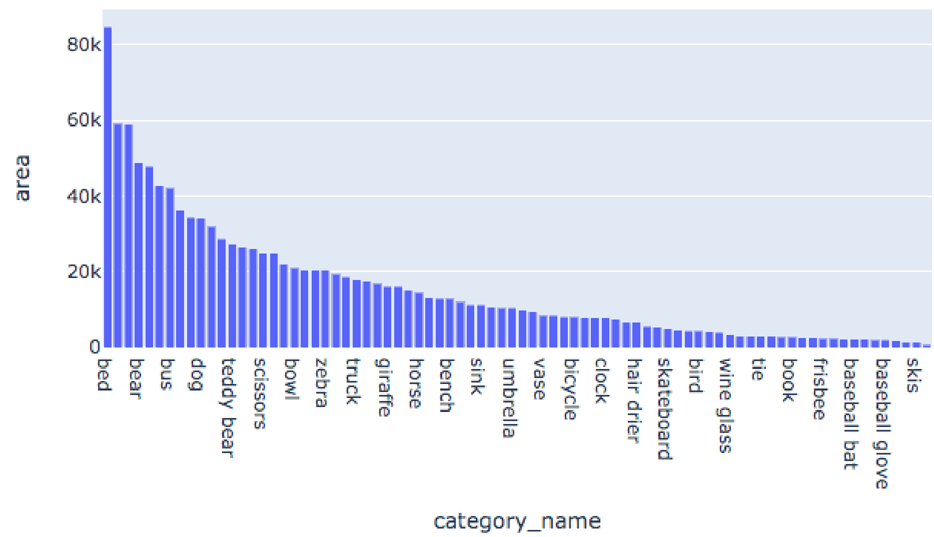
Image sizes and aspect ratios:

- Training your model on image patches (randomly selected during training or extracted before training)
- Resizing the entire dataset to avoid doing this every time you load your data.



Label (objects) sizes and dimensions :

- For instance, it might be able to greatly reduce the model if your dataset solely contains really large items.
- On the other hand, if you have photographs with small objects (10x10px, for example), it's possible that you won't be able to train the model with this configuration.
- When it comes to box or mask dimensions, the following factors should be taken into account:
    1. Aspect ratios.
    2. Size (Area).



Understanding preprocessing sequences:

Any computer vision system must include preprocessing.

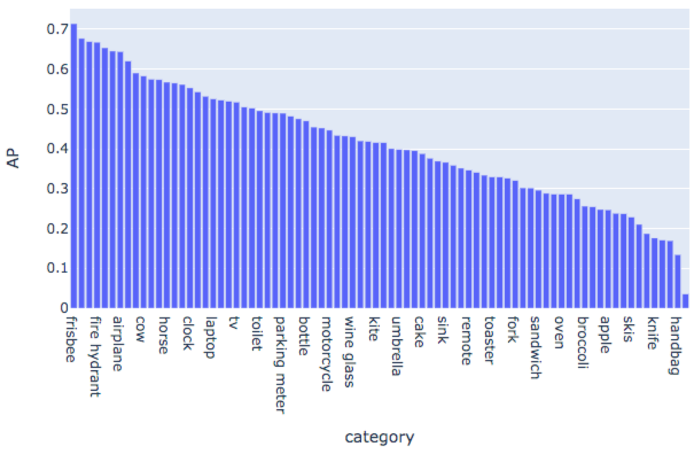
It is more difficult to apply it to object recognition and segmentation problems than to simple picture classification since various changes (like rotation or crop) must be applied to both the source and destination images (masks or bounding boxes). The following frequent transformations need a target transform:

1. Affine transformations,
2. Cropping,
3. Distortions,
4. Scaling,
5. Rotations
6. and many more.

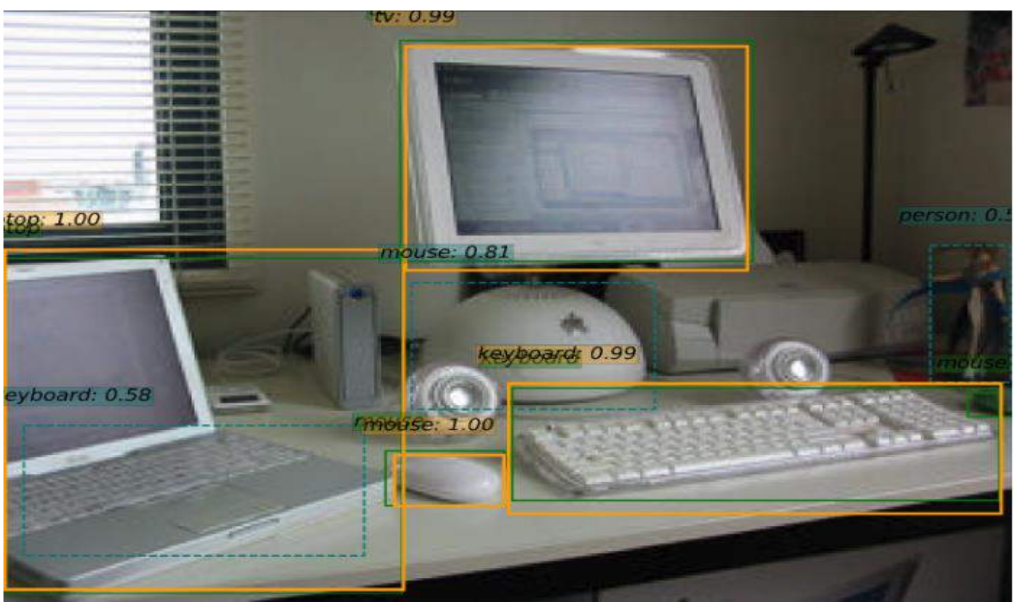
Results understanding:

Let's take a look at the formal Coco Challenge and how that system of assessment functions

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.373
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.590
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.402
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.219
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.409
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.481
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.310
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.495
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.521
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.323
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.560
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.667
```



Visualizing results:



LAB 6 : <u>Optical Flow based Deep Learning Tracking [RAFT]</u>.

<u>Aim:</u> Our aim is to estimate the motion of the image intensities over time in a video using optical-flow algorithms like RAFT. i.e. to calculate Pixel points within the images, and to provide the estimation of where the points could be in the next image sequence in our shopping video dataset.

- **<u>RAFT</u> :**

RAFT can be divided into three stages:

1. Feature extractors: The network input consists of two consecutive frames. To extract features from these two images, we use two CNNs with shared weights.

2. Visual Similarity: Visual similarity is calculated as the inner product of all pairs of feature maps.

3. Iterative update: An iterative update is a sequence of Gated Recurrent Unit (GRU) cells that combine all data we have calculated before.

Input:

Output:

Inference:

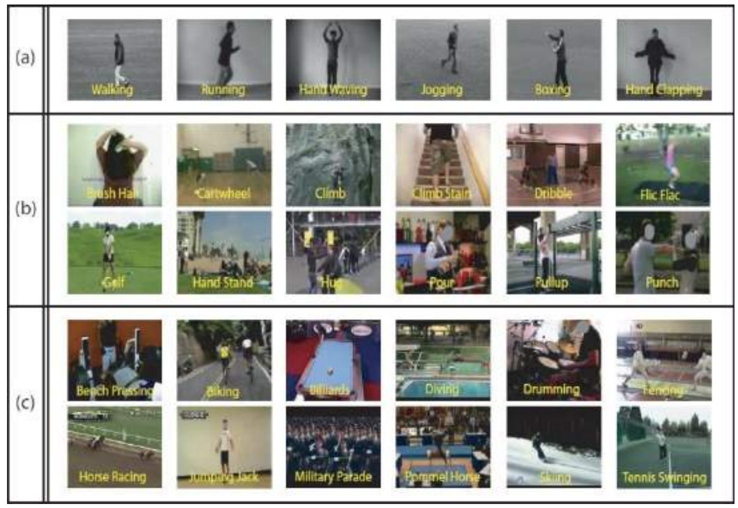
- Here the Optical flow is a per pixel prediction and the main idea is that it assumes a brightness constancy, meaning it tries to estimate how the pixels brightness moves across the screen over time.

LAB 7: Activity recognition: (CNN+LSTM,CNN+RNN)

Aim: Need to identify one or more agents' activities and objectives from a sequence of observations on those actions and the surrounding circumstances.

- **UCF50 - Action Recognition Data Set :**

UCF50 data set's 50 action categories collected from youtube are: Baseball Pitch, Basketball Shooting, Bench Press, Biking, bicycling, Billiards Shot,Breaststroke, Clean and Jerk, Diving, Drumming, Fencing, Golf Swing, Playing Guitar, High Jump, Horse Race, Horse Riding, Hula Hoop, Javelin Throw, Juggling Balls, Jump Rope, Jumping Jack, Kayaking, Lunges, Military Parade, Mixing Batter, Nun chucks, Playing Piano, Pizza Tossing, Pole Vault, Pommel Horse, Pull Ups, Punch, Push Ups, Rock Climbing Indoor, Rope Climbing, Rowing, Salsa Spins, Skate Boarding, Skiing, Skijet, Soccer Juggling, Swing, Playing Tabla, TaiChi, Tennis Swing, Trampoline Jumping, Playing Violin, Volleyball Spiking, Walking with a dog, and Yo Yo.



Outputs:

ACTION	Sample image	Description	Output
Riding a bicycle		The video of a kid riding a bicycle on a road	[bicycling]
Riding motorcycle		The video obtained from the camera fixed o the helmet of a person riding a motorcycle.	[Biking]

Doing YOGA		**This video obtained from the camera, a person doing yoga.**	**[yoga]**

LAB 8 : <u>Computer Vision based library [slovePnp() library]</u>

➢ **<u>Understand the PnP Problem</u>:**

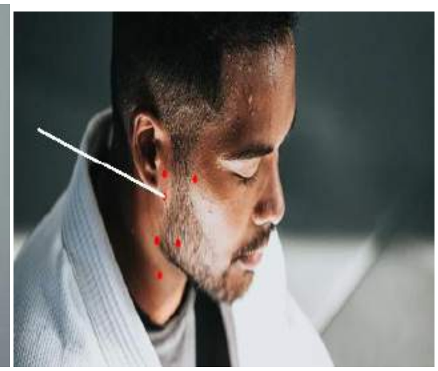
- The PnP problem is very common in Computer Vision and stands for the Perspective n-Points problem. In this issue, we cannot determine the pose of an object with respect to the camera after being provided with the 2D and 3D coordinates.
- This can be understood with the example of <u>face tracking during an online exam</u>. The pose of an object with respect can change with the change in direction.
- The following two types of motions facilitate this change:
- The first type of motion is translational motion, which can happen along any of the three axes. The object moves in a uniform motion in any particular direction, thereby changing its coordinates.
- The second type of motion is the rotational motion, in which the object can revolve around any of the three axes.

➢ **<u>Use the opencv.solvepnp() Function to Solve the PnP Problem</u>:**

- The solvepnp() function from the OpenCV library is used for the pose estimation of a given object with respect to the camera, thus solving the PnP problem. It returns rotational and translational vectors.
- It uses the 2D and 3D coordinates of the object with the camera matrix. The coordinates supplied are of the different features of the face.
- These features are the nose, corners of the mouth, chin, and both of the eyes.

<u>Input and Output:</u>






The camera matrix and the 2D and 3D points of the image's face features serve as the major parameters. It returns the vectors that identify the 3D points of the pose using these values.

Using the projectPoints() function, we can project these points in 2D with respect to the camera. We next use these points to plot a line that will serve as the determined pose's representation in the image.

Inference:

Finally, we plot a line to represent the determined pose in the image using these points.