e-ISSN: 2582-5208

International Research Journal of Modernization in Engineering Technology and Science

(Peer-Reviewed, Open Access, Fully Refereed International Journal)

Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com

IMAGE DETECTION USING OPENCV PYTHON

Kaijan Khan[*1]

[*1]B.Sc. Information Technology, B. K. Birla College, Kalyan, Maharashtra, India.

ABSTRACT

Image detection in Python involves training computers to interpret images and find objects or patterns within them. Python is popular for this task due to its versatility and libraries. The process includes acquiring, preprocessing, and extracting image features. Machine learning models like CNNs are trained, and then these models are used to detect objects in new images. Python libraries like OpenCV, TensorFlow, and PyTorch are key tools. The results are visualized and reported, making Python a powerful choice for image detection in applications like autonomous vehicles and medical imaging.

Keywords: Image Detection, Computer Vision, Machine Learning, Autonomous Vehicles, Medical Imaging.

I. INTRODUCTION

Image recognition is a digital photos or video process to identify and detect an object or feature, and AI is progressing and is being highly effective in using this technology. AI can search for images on social media platforms and equate them to several datasets to determine which ones are important in image search.

Lawrence Roberts is referred to as the real founder of image recognition or computer vision applications as we know them today. In his 1963 doctoral thesis entitled "Machine perception of three-dimensional solids "Lawrence describes the process of deriving 3D information about objects from 2D photographs.

Image detection, often referred to as image recognition or computer vision, is a field in which computers are trained to interpret and understand visual information from images. Python has emerged as a popular programming language for image detection due to its versatility, extensive libraries, and community support.

Edge detection is one of the most important tasks in the field of computer vision and image processing, and edge provides important information for many subsequent visual tasks, such as image recognition [1, 2, 3, 4, 5, 6], image segmentation [7, 8, 9], image retrieval [10, 11, 12, 13, 14], face recognition [15, 16, 17], corner detection [18, 19, 20], road detection [21], medical imaging [22], and target tracking [23, 24, 25, 26]. Each of these visual tasks requires extracting object boundaries or perceiving obvious edges from the original image, Therefore, the stability detection of image edge details in order to efficiently perform the actual task.

Here's a brief description of image detection using Python:

Image detection using Python is a process that involves analyzing images to identify and classify objects, patterns, or features within them. This technology finds applications in various domains, including autonomous vehicles, medical imaging, security systems, and more. The fundamental steps of image detection in Python typically include:

Data Acquisition: Images are captured through cameras, sensors, or collected from various sources.

Preprocessing: This step involves tasks like resizing, normalization, and noise reduction to enhance the quality of images for analysis.

Feature Extraction: Features like edges, textures, and color information are extracted from the images. Python libraries like OpenCV and scikit-image are commonly used for this purpose.

Model Selection: Machine learning and deep learning models are employed to train on labeled image datasets. Convolutional Neural Networks (CNNs) are particularly effective in image detection tasks.

Training: The selected model is trained on a large dataset of images to learn to recognize specific patterns or objects.

Inference: The trained model is applied to new, unseen images for detection. Python libraries like TensorFlow, PyTorch, and Keras facilitate this step.

Post-processing: Detected objects or regions are often further analyzed, and additional information, such as bounding boxes, is added.

Visualization and Reporting: The results are visualized and reported, which can include annotating images with detection outcomes.

Python's extensive libraries and frameworks make it a powerful tool for image detection, with OpenCV, TensorFlow, and PyTorch being the most commonly used ones. These libraries provide a wide range of functions and pre-trained models that simplify the development of image detection systems. The versatility of Python, along with its strong community support, makes it an ideal choice for both beginners and experts in the field of image detection.

II. METHODOLOGY

Methodology of analysis and conclusions:

Here's the methodology followed by this research:

1. Install OpenCV: Use pip to install the OpenCV library in your Python environment.

2. Import OpenCV: Import the necessary modules from the OpenCV library in your Python script.

3. Load the Image: Use the "imread" function to load the image you want to detect.

4. Preprocess the Image: Apply any necessary preprocessing steps, such as resizing, converting to grayscale, or applying filters.

5. Define the Object: If you want to detect a specific object, you'll need to define its characteristics or create a pre-trained model.

6. Detect Objects: Use the appropriate OpenCV function, like "cv2.detectMultiScale", to detect objects in the image.

7. Draw Boundaries: Once objects are detected, you can draw bounding boxes or other visual indicators around them.

8. Display the Result: Show the resulting image with the detected objects using the "imshow" function.

9. Handle User Input: Implement any additional functionality you need, such as saving the output or handling user input. This is just a high-level overview which was I followed there are various ways to do it.

III. MODELING AND ANALYSIS

According to my analysis image detection using Python and OpenCV, there are a few key steps which I can follow through this research:

1. Dataset Preparation: Gather a dataset of images that includes both positive samples (containing the object you want to detect) and negative samples (without the object).

2. Feature Extraction: Extract relevant features from the positive and negative samples using techniques like Histogram of Oriented Gradients (HOG) or Haar-like features.

3. Train a Classifier: Use a machine learning algorithm, such as Support Vector Machines (SVM), Random Forests, or Convolutional Neural Networks (CNNs), to train a classifier on the extracted features.

4. Evaluation: Evaluate the performance of your trained model using metrics like accuracy, precision, recall, and F1 score. You can use techniques like cross-validation or holdout validation.

5. Model Optimization: Fine-tune your model by adjusting hyperparameters, exploring different feature extraction techniques, or using data augmentation to improve its performance.

6. Object Detection: Once your model is trained, you can use it to detect objects in new images by applying a sliding window or using more advanced techniques like region proposal methods or deep learning-based object detection frameworks (e.g., YOLO, SSD).

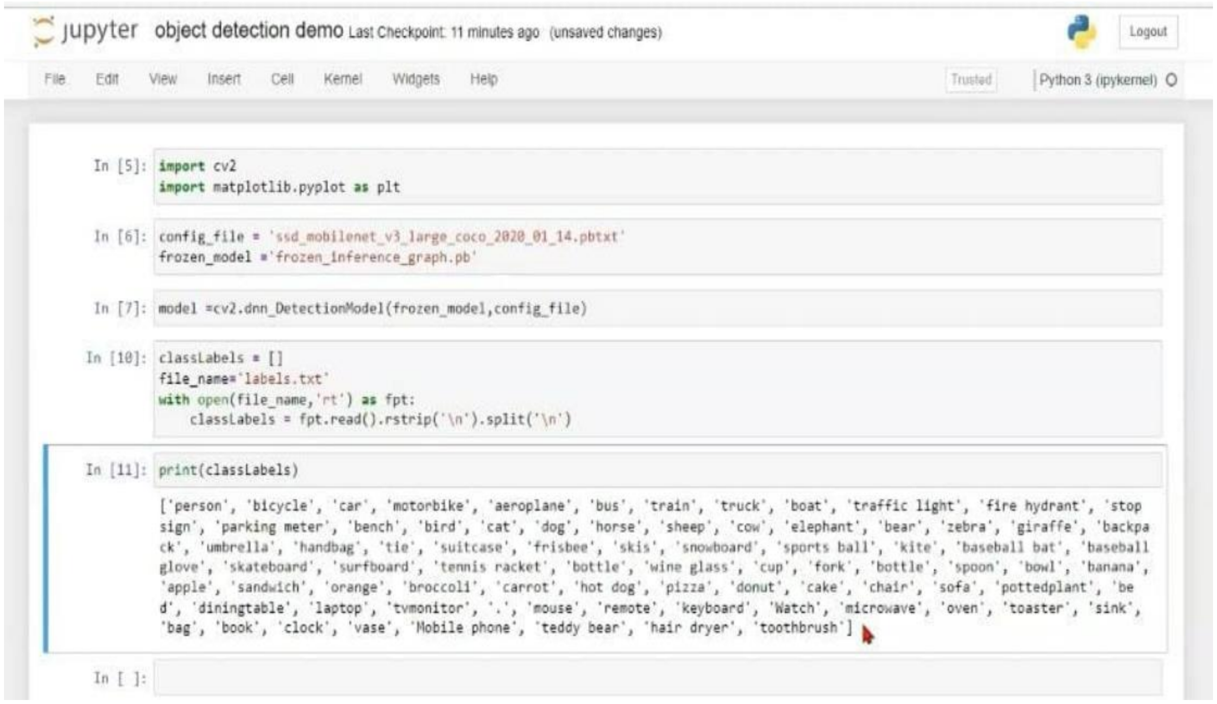
7. Post-processing: Apply post-processing techniques like non-maximum suppression to remove duplicate or overlapping detections and refine the final output.

8. Performance Analysis: Analyze the performance of your model in terms of detection accuracy, speed, and any specific requirements or constraints of your application.

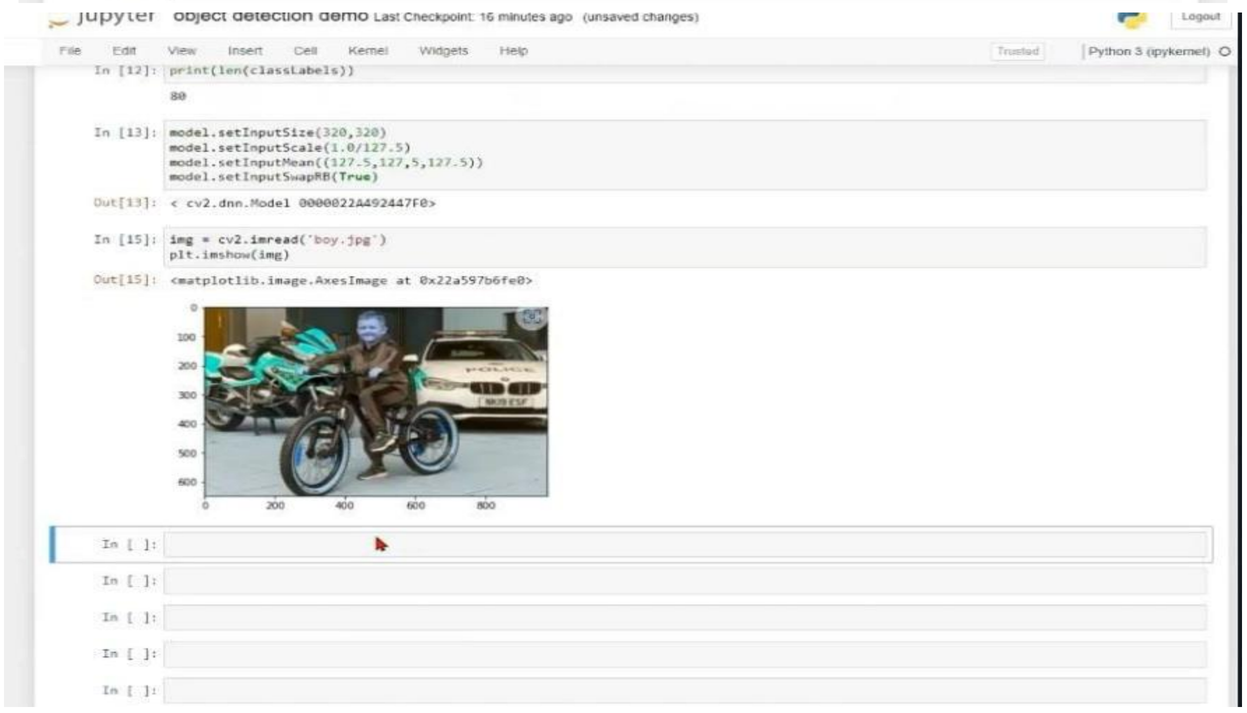
Note: Image detection is a vast field, and there are many variations and advanced techniques beyond this basic methodology.

IV. RESULTS AND DISCUSSION

Showing some results which, I did through this research using OpenCV Python. Through this research I am trying to explain OpenCV Python library. This experiment will help you to understand the full OpenCV Python.

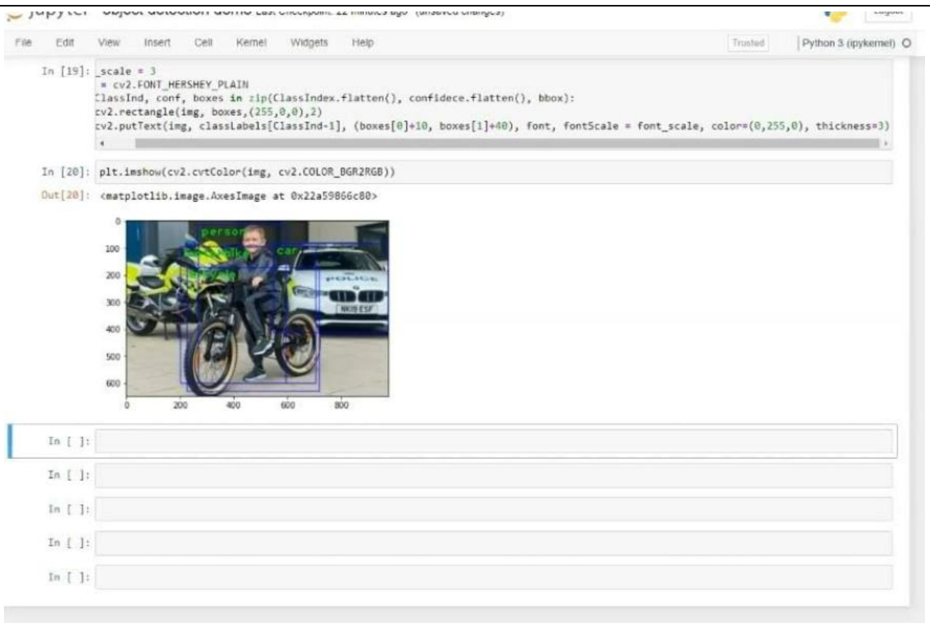

jupyter object detection demo Last Checkpoint: 11 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)

```
In [5]: import cv2
        import matplotlib.pyplot as plt

In [6]: config_file = 'ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
        frozen_model ='frozen_inference_graph.pb'

In [7]: model =cv2.dnn_DetectionModel(frozen_model,config_file)

In [10]: classLabels = []
         file_name='labels.txt'
         with open(file_name,'rt') as fpt:
             classLabels = fpt.read().rstrip('\n').split('\n')

In [11]: print(classLabels)

         ['person', 'bicycle', 'car', 'motorbike', 'aeroplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop
         sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpa
         ck', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball
         glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'bottle', 'spoon', 'bowl', 'banana',
         'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'sofa', 'pottedplant', 'be
         d', 'diningtable', 'laptop', 'tvmonitor', '.', 'mouse', 'remote', 'keyboard', 'Watch', 'microwave', 'oven', 'toaster', 'sink',
         'bag', 'book', 'clock', 'vase', 'Mobile phone', 'teddy bear', 'hair dryer', 'toothbrush']

In [ ]:
```

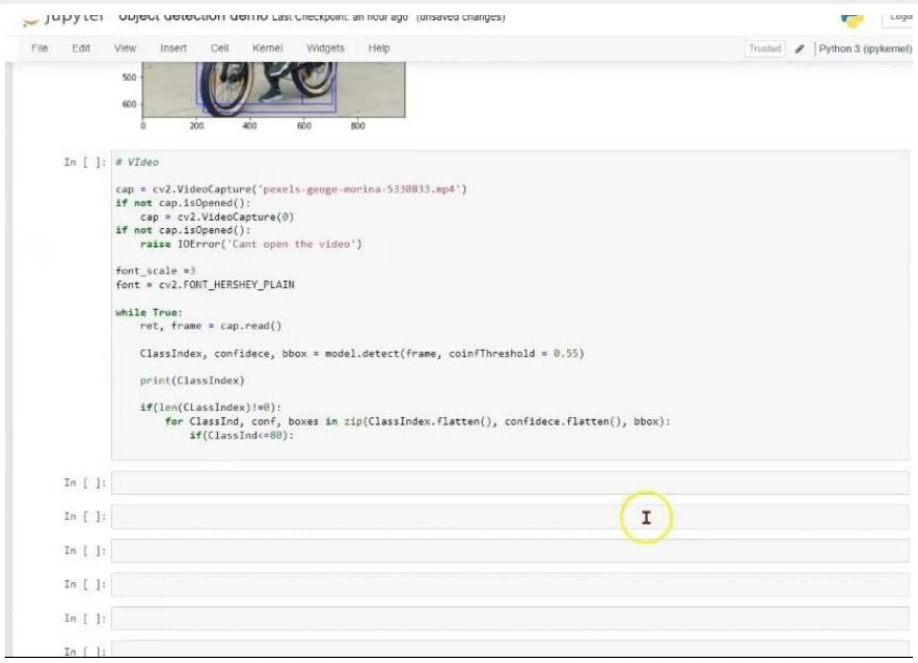

jupyter object detection demo Last Checkpoint: 16 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
0
100
200
300
400
500
600
0
200
400
600
800

```
In [12]: print(len(classLabels))

         80

In [13]: model.setInputSize(320,320)
         model.setInputScale(1.0/127.5)
         model.setInputMean((127.5,127,5,127.5))
         model.setInputSwapRB(True)

Out[13]: < cv2.dnn.Model 00000224492447F0>

In [15]: img = cv2.imread('boy.jpg')
         plt.imshow(img)

Out[15]: <matplotlib.image.AxesImage at 0x22a597b6fe0>

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```

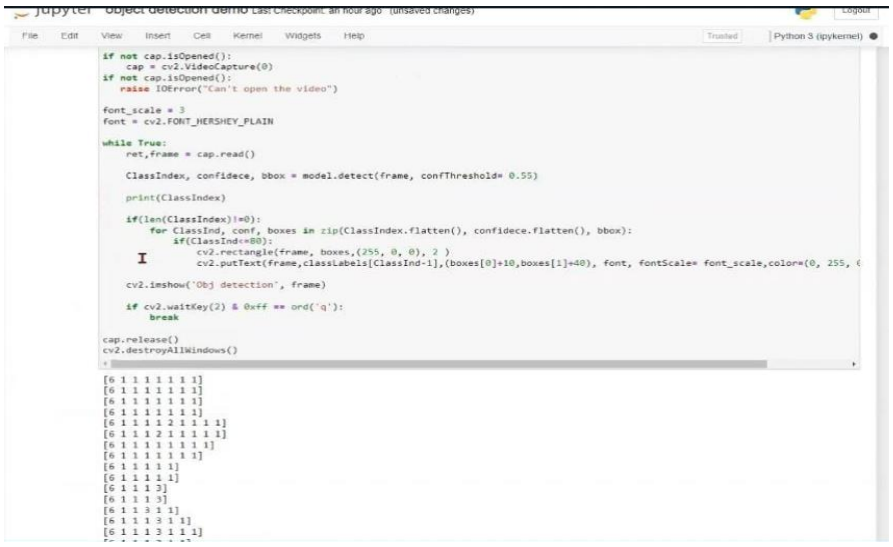
e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:05/Issue:10/October-2023 Impact Factor- 7.868 www.irjmets.com



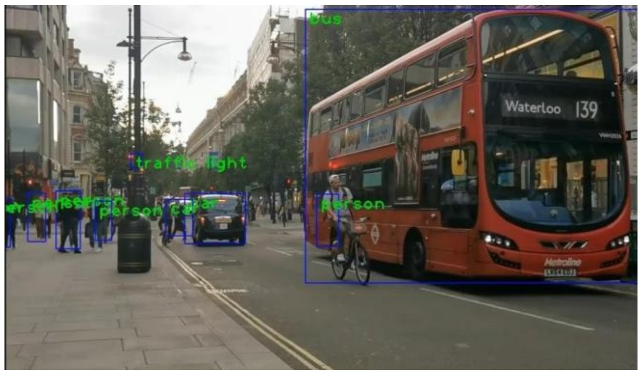
```
In [19]: _scale = 3
          = cv2.FONT_HERSHEY_PLAIN
         ClassInd, conf, boxes in zip(ClassIndex.flatten(), confidece.flatten(), bbox):
         cv2.rectangle(img, boxes,(255,0,0),2)
         cv2.putText(img, classLabels[ClassInd-1], (boxes[0]+10, boxes[1]+40), font, fontScale = font_scale, color=(0,255,0), thickness=3)

In [20]: plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

Out[20]: <matplotlib.image.AxesImage at 0x22a59866c80>

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

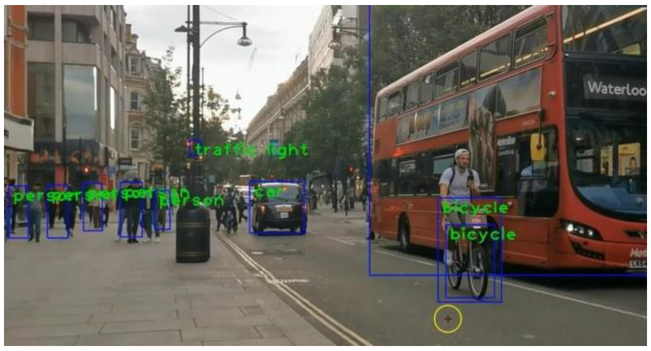


```
In [ ]: # VIdeo

        cap = cv2.VideoCapture('pexels-geoge-morina-5330833.mp4')
        if not cap.isOpened():
            cap = cv2.VideoCapture(0)
        if not cap.isOpened():
            raise IOError('Cant open the video')

        font_scale =3
        font = cv2.FONT_HERSHEY_PLAIN

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

            ClassIndex, confidece, bbox = model.detect(frame, coinfThreshold = 0.55)

            print(ClassIndex)

            if(len(CLassIndex)!=0):
                for ClassInd, conf, boxes in zip(ClassIndex.flatten(), confidece.flatten(), bbox):
                    if(ClassInd<=80):

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



```
if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
   raise IOError("Can't open the video")

font_scale = 3
font = cv2.FONT_HERSHEY_PLAIN

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

    ClassIndex, confidece, bbox = model.detect(frame, confThreshold= 0.55)

    print(ClassIndex)

    if(len(ClassIndex)!=0):
        for ClassInd, conf, boxes in zip(ClassIndex.flatten(), confidece.flatten(), bbox):
            if(ClassInd<=80):
                cv2.rectangle(frame, boxes,(255, 0, 0), 2 )
                cv2.putText(frame,classLabels[ClassInd-1],(boxes[0]+10,boxes[1]+40), font, fontScale= font_scale,color=(0, 255,

    cv2.imshow('Obj detection', frame)

    if cv2.waitKey(2) & 0xff == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()

[6 1 1 1 1 1 1 1]
[6 1 1 1 1 1 1 1]
[6 1 1 1 1 1 1 1]
[6 1 1 1 1 1 1 1]
[6 1 1 1 1 2 1 1 1 1]
[6 1 1 1 2 1 1 1 1 1]
[6 1 1 1 1 1 1 1 1]
[6 1 1 1 1 1 1 1]
[6 1 1 1 1 1]
[6 1 1 1 1 1]
[6 1 1 1 3]
[6 1 1 1 3]
[6 1 1 3 1 1]
[6 1 1 1 3 1 1]
[6 1 1 1 3 1 1 1]
```





This is the final output which I did through OpenCV Python.

V. CONCLUSION

By all the proven results, we can conclude through this statement which is Python and OpenCV provide powerful tools for image detection, allowing you to detect objects, faces, or other features in images. By leveraging machine learning algorithms and techniques, such as SVM or CNNs, you can train models to accurately detect objects in images.

ACKNOWLEDGEMENT

I would like to express my gratitude to my advisory Prof. Swapna Nikale who has give me this opportunity to publish the research paper as a part of curricular activity. Thanks to give support and deep guidance on writing this research paper.

VI. REFERENCES

[1] Mohan, K., Seal, A., Krejcar, O., Yazidi, A., 2020. Facial expression recognition using local gravitational force descriptor-based deep convolution neural networks. IEEE Transactions on Instrumentation and Measurement 70, 1–12.

[2] Olson, C.F., Huttenlocher, D.P., 1997. Automatic target recognition by matching oriented edge pixels. IEEE Transactions on Image Processing 6, 103–113.

[3] Du, J.X., Huang, D.S., Wang, X.F., Gu, X., 2007. Shape recognition based on neural networks trained by differential evolution algorithm. Neurocomputing 70, 896–903.

[4] Vu, N.S., Caplier, A., 2011. Enhanced patterns of oriented edge magnitudes for face recognition and image matching. IEEE Transactions on Image Processing 21, 1352–1365.

[5] Drolia, U., Guo, K., Tan, J., Gandhi, R., Narasimhan, P., 2017. Cachier: Edge-caching for recognition applications, in: 2017 IEEE 37th international conference on distributed computing systems (ICDCS), pp. 276–286

[6] Ramadevi, Y., Sridevi, T., Poornima, B., Kalyani, B., 2010. Segmentation and object recognition using edge detection techniques. AIRCC's International Journal of Computer Science and Information Technology 2, 153–161

[7] Wei, Y., Liang, X., Chen, Y., Shen, X., Cheng, M.M., Feng, J., Zhao, Y., Yan, S., 2016. STC: A simple to complex framework for weakly-supervised semantic segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 39, 2314–2320.

[8] Arbeláez, P., Pont-Tuset, J., Barron, J.T., Marques, F., Malik, J., 2014. Multiscale combinatorial grouping, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 328–335.

[9] Wang, X.F., Huang, D.S., Xu, H., 2010. An efficient local chan–vese model for image segmentation. Pattern Recognition 43, 603–618.

[10] Pavithra, L., Sharmila, T.S., 2018. An efficient framework for image retrieval using color, texture and edge features. Computers & Electrical Engineering 70, 580–593

[11] Gordo, A., Almazán, J., Revaud, J., Larlus, D., 2016. Deep image retrieval: Learning global representations for image search, in: European Conference on Computer Vision, pp. 241–257.

[12] Lin, K., Yang, H.F., Hsiao, J.H., Chen, C.S., 2015. Deep learning of binary hash codes for fast image retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 27–35.

[13] Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., Chum, O., 2018. Revisiting oxford and paris: Large-scale image retrieval benchmarking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5706–5715.

[14] Li, B., Huang, D.S., Wang, C., Liu, K.H., 2008a. Feature extraction using constrained maximum variance mapping. Pattern Recognition 41, 3287–3294.

[15] Zhao, Z.Q., Huang, D.S., Sun, B.Y., 2004. Human face recognition based on multi-features using neural networks committee. Pattern recognition letters 25, 1351–1358.

[16] Chen, W.S., Yuen, P.C., Huang, J., Dai, D.Q., 2005. Kernel machinebased one-parameter regularized Fisher discriminant method for face recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 35, 659–669.

[17] Li, B., Zheng, C.H., Huang, D.S., 2008b. Locally linear discriminant embedding: An efficient method for face recognition. Pattern Recognition 41, 3813–3821.

[18] Zhang, W.C., Wang, F.P., Zhu, L., Zhou, Z.F., 2014. Corner detection using Gabor filters. IET Image Processing 8, 639–646.

[19] Zhang, W., Shui, P., 2015. Contour-based corner detection via angle difference of principal directions of anisotropic Gaussian directional derivatives. Pattern Recognition 48, 2785–2797.

[20] Zhang, W., Sun, C., Breckon, T., Alshammari, N., 2019b. Discrete curvature representations for noise robust image corner detection. IEEE Transactions on Image Processing 28, 4444–4459.

[21] Dollar, P., Tu, Z., Belongie, S., 2006. Supervised learning of edges and object boundaries, in: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1964–1971.

[22] Qiu, B., Guo, J., Kraeima, J., Glas, H.H., Zhang, W., Borra, R.J., Witjes, M.J.H., van Ooijen, P., 2021. Recurrent convolutional neural networks for 3D mandible segmentation in computed tomography. Journal of Personalized Medicine 11, 492.

[23] Chi, Z., Li, H., Lu, H., Yang, M.H., 2017. Dual deep network for visual tracking. IEEE Transactions on Image Processing 26, 2005– 2015.

[24] Leal-Taixé, L., Canton-Ferrer, C., Schindler, K., 2016. Learning by tracking: Siamese CNN for robust target association, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 33–40.

[25] Xu, Y., Brownjohn, J., Kong, D., 2018. A non-contact vision-based system for multipoint displacement monitoring in a cable-stayed footbridge. Structural Control and Health Monitoring 25, e2155.

[26] Ojha, S., Sakhare, S., 2015. Image processing techniques for object tracking in video surveillance-a survey, in: 2015 International Conference on Pervasive Computing, pp. 1–6.