CS 763 Computer Vision

Task02

22M1079, 22M1177, 22M2119

February 23, 2023

Contents

1 Introduction

This task is an extension of the previous class discussions on faces and computer vision. In this task, we will be exploring face detection, identification, and even some fun makeup applications on detecting faces. The face recognition package from
https://github.com/ageitgey/face_recognition will be used for this task. While it is not part of OpenCV, it provides a user-friendly package for face recognition.
The task is divided into several sub-tasks, including image and video face detection and comparison of the Viola Jones and HOG methods for face detection. In the first sub-task, we will detect and extract individual faces from a sample face-detection image and group members' dataset images. The output will include the bounding box coordinates of each detected face, which will be used to extract all faces as sub-images. In the second sub-task, we will repeat the previous sub-task on videos, draw a bounding box for every frame detected, and save the video with face bounding boxes. Finally, in the third sub-task, we will compare the results of using the Viola Jones and HOG methods for face detection. We will copy the bounding box information from the group members dataset in Task01 into data/captured, assume it as ground truth, and output the results obtained from both methods using the intersection-over-union metric.

2 Library Used

The following libraries were used:

OpenCV: An open-source computer vision and machine learning software library that is commonly used for image processing tasks such as object detection and face recognition.
Numpy: A library for scientific computing in Python, used for processing arrays and matrices in image processing tasks.

OS: A library in Python for interacting with the operating system, such as reading or writing to the file system.

Sys: A library in Python for interacting with the Python interpreter, such as accessing arguments passed to the script.

Argparse: A library in Python for parsing command-line arguments, used for allowing the user to specify options for the script. This library allowed for greater flexibility and control over the input parameters, such as the input folder containing the images and the output folder to store the annotations. The parsed arguments were then used to perform the annotation process, where the bounding boxes were drawn around the faces in the images. The use of argparse made it easier to reuse the code for annotation on different sets of images and to easily modify the input parameters for different use cases. It also made the code more user-friendly by

allowing the user to specify the parameters from the command line rather than hard-coding them in the code.

face_recognition: The face_recognition package is a Python library for face recognition that uses deep learning algorithms to detect and recognize faces in images and videos. It can detect faces using various algorithms, recognize faces by comparing with a database of known faces, encode facial features into a vector, and detect facial landmarks. The package is easy to install and use, with a simple API and pre-trained models. It can be used for applications like security, surveillance, entertainment, and social media.

3 Face Detection

Face detection is a computer technology that is used to detect human faces in digital images and videos. It is an important task in computer vision and has a wide range of applications, including surveillance, security, entertainment, and social media.

The process of face detection involves analyzing an image or video frame to locate regions that may contain a face.

i. Images

The first step requires detecting and extracting individual faces from a sample face-detection image (located in data/samples/arnold-sylvester) and saving the resulting images with bounding box coordinates on the standard output. The extracted faces follows a specific naming convention. Next, for the second step, we used the previously created dataset (Task01), store these images in data/captured (e.g. 01.jpg, 02.jpg). For step three, we repeat the face detection and extraction process on the new dataset created in step two.

Finally, the extracted face images are saved in results/faceDetection/.

ii. Videos

The task requires the code to be written for face detection in videos. The input video path is provided as data/samples/arnold.mp4. For each frame in the video, a bounding box for every detected face should be drawn. The resulting video with face bounding boxes should be saved as results/faceDetection/arnold.mp4. This process should be repeated for the videos captured in Task01 and stored as data/captured/faceVideo0?.mp4. The output video should also be saved in results/faceDetection/faceVideoOutput0?.mp4 with an output size of 240p. The accuracy percentage of frames with successful detection are noted.

iii. VJ vs HoG

We compared Viola-Jones (VJ) and Histogram of Oriented Gradients (HoG) methods for face detection. VJ uses Haar-like features to detect faces, while HoG uses gradient orientation information to detect faces. We used the pre-trained classifier file

haarcascade_frontalface_alt.xml for VJ and face_recognition package for HoG. We compared the results obtained from these methods using the intersection-over-union (IoU) metric.

Our results show that HoG outperformed VJ in terms of accuracy. The IoU score for HoG was higher compared to VJ, indicating that the bounding boxes generated by HoG were closer to the ground truth. However, VJ had faster processing time compared to HoG, making it more suitable for real-time applications. Another advantage of VJ is that it can detect faces at different scales, while HoG is better suited for detecting faces at a fixed scale.

So, VJ is fast and efficient, performs well under varying lighting and expressions, simple feature set suitable for low-power devices but struggles with detecting faces at different scales, partially occluded faces, produces false positives whereas, HoG is robust to pose and illumination variations, detects objects at multiple scales, produces accurate and detailed boundaries but computationally expensive for multi-scale detection, requires large number of training samples, sensitive to background clutter and occlusion.

In summary, while VJ has the advantage of faster processing time and the ability to detect faces at different scales, HoG outperforms VJ in terms of accuracy. Therefore, the choice between these two methods depends on the specific requirements of the application.

4 Face Recognition

Face recognition is the process of identifying or verifying the identity of a person by analyzing and comparing patterns in their facial features with those stored in a database. It involves the capture of an image or video of a person's face, detection of facial features such as eyes, nose, and mouth, extraction of unique facial features, and comparison of those features with known faces in a database to find a match.

i. Images

This sub-task involves recognizing known faces in an image with multiple faces. Firstly, the known faces are Arnold and Sylvester, and all the faces are detected in the image using bounding boxes. Next, the detected faces are annotated with the recognized person's name and saved in a specific folder. If a detected face is unknown, it is annotated as "Unknown". This experiment is then repeated with the lowest two roll numbers in the group as known faces and the third as unknown. In this case, two query images are used, and the expected results are "Unknown" for all faces in the "arnold-sylvester-unknown.jpg" image. The output image should have thickened lines, readable text, and trimmed long names. The result is saved in the "results/faceDetection/sampleRecognized0?.jpg" folder.

ii. Video

The task involves face recognition in videos for known individuals. The first step is to read the input video path, provided in the data/samples/arnold.mp4 directory and consider Arnold as known. For each frame in the video, the faces are detected, recognized, and annotated with their corresponding names. The resulting output video is saved in the results/faceRecognition/ directory as sampleOutput.mp4. The same steps are repeated for the videos captured in Task01, with the two lowest roll numbers being considered as known. The resulting output videos are saved in the results/faceRecognition/ directory as faceVideoOutput0?.mp4.

iii. Images with Masks

The task involves updating the face recognition system to adapt to face masks. First, a test image set with face masks for each member is created and stored as masked01.jpg, masked02.jpg, etc. in the data/captured folder. Then, face recognition is performed using the same code as before, considering only two members as known. Finally, the results are displayed in the image section in a similar format as before.

5 Fun with Images

In this sub-task, we will manipulate the facial image to apply makeup to the face. This requires the identification of various facial landmarks such as eyes, eyebrows, lips, nose, etc. Detect the facial landmarks in the input image (data/samples/arnold-sylvester) and save the results in results/drawing as peopleLandmarks. Repeat the above experiment on data/samples/angelina, and using the detected facial landmarks, apply makeup to the face. Save the output image with face makeup as results/drawing/angelinaBeautiful.

The face_recognition natively supports this task and was done using PIL, but us we have done it using OpenCV.
Here is a sample output:

6 Conclusion

In conclusion, this task focused on face detection, identification, and fun makeup on faces using the face_recognition package. In subtask 2.1, the task involved detecting and extracting individual faces from sample face detection images and storing the output with face followed by the input file name in results/faceDetection/. In subtask 2.2, the task was repeated on videos, where a bounding box was drawn for every detected frame, and accuracy percentage of frames with successful detection was reported. Finally, in subtask 2.3, the goal was to compare two methods, Viola-Jones built into OpenCV and HOG built into face_recognition for face detection, using the intersection-over-union metric. The results were analyzed, and pros and cons of both methods were discussed. Overall, the task was a valuable exercise in implementing computer vision techniques for face detection and identification.