Assignment 1

COSC 2116

Agam Jammu

Introduction:

Object detection is a crucial task in computer vision that involves locating instances of objects in images or videos. The Haar Cascade classifier is a machine learning-based approach that is widely used for object detection in images and videos. It is based on a cascade of simple classifiers that can efficiently classify the sub-windows of an image or video frame. In this report, I will discuss the process of training a Haar Cascade classifier for object detection, including the collection and preparation of training data, cascade training, and classifier testing.

Data Collection:

The first step in training a Haar Cascade classifier is to collect a set of positive and negative images. Positive images are the images that contain the object we want to detect, while negative images are the images that do not contain the object. In our case, we collected 4000 positive images and 2000 negative images, each with a size of 24 x 24 pixels.

Positive images are usually obtained through web scraping or capturing images of the object of interest. Negative images can be obtained from a variety of sources, such as randomly sampled images from the internet or from an existing dataset.

Video Processing:

In addition to images, we can also collect training data from videos. This can be done by extracting frames from the video and using them as images for the classifier. We start by defining the video file paths and the size of the crops for the negative images. We then initialize an empty list to store the frames from the videos. Next, we loop over the video files, and for each video file, we initialize a video object and loop over the frames in the video. We read each frame from the video and append it to the frames list. If there is an error in reading the frame, we print an error message.

Positive and Negative Image Annotation: After collecting the frames from the videos, we loop over the frames and use the ‘selectROI’ function from the cv2 library to annotate the frames with positive and negative images. We first annotate the frame for the positive image, crop the frame to create the positive image, resize the positive image to the desired size, set the directory for the positive images, save the positive image, and append it to the list of positive images. We then annotate the frame for the negative image, crop the frame to create the negative image, resize the negative image to the desired size, set the directory for the negative images, save the negative image, and append it to the list of negative images.

Positive images are annotated with a bounding box that tightly encloses the object of interest. Negative images can be annotated by drawing a box around a region that does not contain the object. It is important to ensure that the negative images do not contain any instances of the object of interest.

Text File Generation:

We generate two text files for training the classifier, one for the positive images and one for the negative images. For the positive images, we write the filename, the coordinates of the object in the image, and the size of the object. For the negative images, we only write the filename.

Cascade Training:

Once we have generated the positive and negative samples and written them into text files, we can start the cascade training process using the OpenCV library. The training process involves specifying the parameters for the classifier, such as the number of stages, the size of the features, and the minimum number of positive and negative images required for each stage. We also specify the paths to the positive and negative text files and the output directory for the classifier.

Cascade training starts with a weak classifier, and multiple stages are used to build a strong classifier. Each stage involves applying a strong classifier to the sub-windows of the image and rejecting those that do not contain the object. The positive images are used to train the classifier, while the negative images are used to determine which sub-windows to reject. The result of each stage is a smaller set of sub-windows that are more likely to contain the object, and the process continues until the desired level of accuracy is achieved.

The cascade training process can take a significant amount of time, depending on the number of stages and the size of the dataset. However, the resulting classifier is highly accurate and can detect objects in images and videos with high precision.

Classifier Testing:

Finally, we can test the trained classifier on a video. We first load the trained classifier, open the video file, and loop through the video frames. For each frame, we convert it to grayscale, detect objects in the frame using the classifier, draw rectangles around the detected objects, and display the frame. We exit the loop on 'q' key press, release the video capture, and close the window.

The trained Haar Cascade classifier can be used for a variety of object detection applications, including face detection, pedestrian detection, and license plate detection. With its high accuracy and efficient performance, it is a powerful tool for computer vision and machine learning applications.

Conclusion:

In conclusion, the Haar Cascade classifier is a machine learning-based approach that is widely used for object detection in images and videos. It is based on a cascade of simple classifiers that can efficiently classify the sub-windows of an image or video frame. The process of training a Haar Cascade classifier involves collecting and preparing the training data, cascade training, and classifier testing.

The first step in training a Haar Cascade classifier is to collect a set of positive and negative images. Positive images are the images that contain the object we want to detect, while negative images are the images that do not contain the object. After collecting the images, we annotate them to create positive and negative samples, and then generate text files for training the classifier.

Cascade training involves specifying the parameters for the classifier, such as the number of stages, the size of the features, and the minimum number of positive and negative images required for each stage. We also specify the paths to the positive and negative text files and the output directory for the classifier. The training process starts with a weak classifier and uses multiple stages to build a strong classifier. Each stage involves applying a strong classifier to the sub-windows of the image and rejecting those that do not contain the object.

Finally, we can test the trained classifier on a video. We load the trained classifier, open the video file, and loop through the video frames. For each frame, we detect objects in the frame using the classifier, draw rectangles around the detected objects, and display the frame.

The Haar Cascade classifier is a powerful tool for object detection in images and videos. It is widely used in computer vision and machine learning applications and can be applied to a variety of object detection tasks. With its high accuracy and efficient performance, it is a valuable asset for researchers and practitioners in the field of computer vision.

Screenshots:

A screenshot of a computer

Description automatically generated with low confidenceA picture containing text

Description automatically generated