Report

The objective of this project involves the development of a Python script for gesture recognition, leveraging handshape feature extraction from video data.

Here is an in-depth breakdown of the approach and solution:

Import the following libraries: "cv2" for image manipulation, "numpy" for numerical operations, "os" for system interactions, "glob" for file handling, "handshape_feature_extractor" for extracting handshape features from images, and "frameextractor" for the retrieval of frames from video data.

Following this, the script initializes an empty NumPy array, "features," with dimensions [len(pathlist), 27]. Here, len(pathlist) represents the count of video files within the "traindata" directory. This array's purpose is to house the extracted handshape feature vectors derived from the middle frame of each video. Additionally, I establish a "frames_output" variable to store these extracted frames. Then, I construct an instance of the "HandShapeFeatureExtractor" class, which resides in the "handshape_feature_extractor" module. This class includes methods for extracting handshape features from images. Next, for each video file within the "traindata" directory, our code proceeds to extract the middle frame utilizing the "frameExtractor" function, as defined in the "frameextractor" module. Subsequently, this extracted frame is read as a grayscale image via the "cv2.imread" function, and its handshape feature vector is derived using the "hf.extract_feature" method from the "HandShapeFeatureExtractor" class. The resulting feature vector is then stored within the "features" array.

Once the training data has been processed in this manner, our script repeats a similar operation for the test data. I create an empty NumPy array named "test_features" with dimensions [len(testlist), 27], where len(testlist) represents the count of video files in the "test" directory. I also set up a "test_frames_output" variable to save the extracted frames. In every video file in the "test", the code extracts the middle frame, reads the frame as a grayscale image, and extracts the corresponding handshape feature vector. These feature vectors are then stored in the "test_features" array.

The next step is to use dot product to compute the similarity between the two sets of feature vectors. The outcome similarity matrix is saved in the variable "simil," which is a NumPy array with dimensions [len(pathlist), len(testlist)]. For each column in this matrix, we calculate the index of the maximum value using the "np.argmax" function, which returns the index of the largest element in the array.The final step is saving the result indexes to a CSV file, namely "Results.csv," by using "np.savetxt" function.