**Department of Computer Engineering**



**Cairo University**

**Faculty of Engineering**

**CMP3006 – Spring 2023**

**Pattern Recognition**

**Hand Gesture Recognition**

**Submitted to**

Dr. Abdelmonem

Eng. Mohamed Shawky

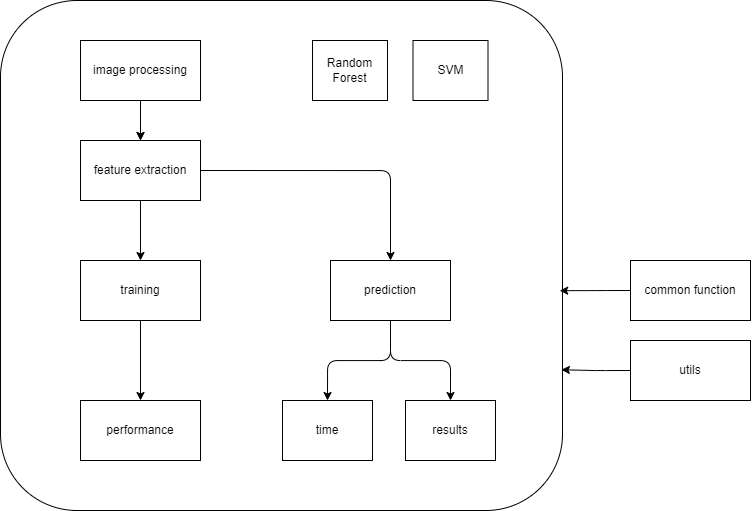
**Submitted by**

|  |  |  |
| --- | --- | --- |
| **Name** | **Sec** | **BN** |
| **Ahmed Hosny** | **1** | **2** |
| **Ahmed Sayed** | **1** | **3** |
| **Eman Mohamed Shahda** | **1** | **16** |
| **Nour Ziad Almulhem** | **2** | **31** |

**Work distribution:**

|  |  |
| --- | --- |
| **Ahmed Hosny** | **SVM**  **HOG – LBP**  **Tunning modules** |
| **Ahmed Madbouly** | **Image processing** |
| **Eman Shahda** | **Random Forest**  **Shi Thomas** |
| **Nour Ziad** | **ML Pipeline**  **ORB - SIFT** |

**Project Pipeline:**



**Preprocessing module:**

The **image\_pre\_processing** function is the main function that performs the image preprocessing steps. It takes an input image as a parameter and returns the preprocessed result.

1. Resizing: The input image is first resized to a smaller size to make the preprocessing faster and take less time while maintaining its aspect ratio.
2. Brightness Calculation: The algorithm calculates the average brightness of the resized image. This value is used to determine which method to use for binary conversion in the next step.
3. Binary Conversion: Based on the calculated brightness value, the algorithm decides whether to use a "high contrast" or "low/medium contrast" binary conversion method.
   * If the average brightness is high, indicating a well-lit image, the algorithm applies a "high contrast" binary conversion method. It converts the image to a different color space(YUV), enhances the contrast using histogram equalization, and applies a thresholding operation to obtain a binary image.
   * If the average brightness is low to medium, indicating a darker image, the algorithm uses a "low/medium contrast" binary conversion method. It converts the image to a different color space, applies thresholding operations, and performs morphological operations (such as erosion and dilation) to refine the binary image.
4. Edge detection: to perform this step firstly we reduce the size of the image a bit to avoid dealing with the whole image as an one contour , then blur the image to remove noise , finally apply canny edge detection
5. Contour Extraction: The algorithm extracts the contours from the binary image. Contours represent the boundaries of objects or regions of interest in the image. This step helps in isolating and segmenting important features.
6. Filtering and Smoothing: The algorithm applies filters and smoothing techniques to reduce noise and refine the extracted contours. This step helps to remove any unwanted artifacts and improve the overall quality of the image.
7. Result Generation: Finally, the algorithm generates the preprocessed result. It applies additional filtering or transformations to enhance the image quality. The resulting image is typically resized to a standardized size for consistency and compatibility with further analysis or processing tasks.

By performing these preprocessing steps, the algorithm aims to enhance the quality of the input image, improve contrast and clarity, and extract meaningful features.

**Feature Extraction/Selection Module:**

**Hog Transform** is a feature extraction technique to extract features from an image. Here's a breakdown of the parameters used in the hog\_features function:

orientations: This parameter specifies the number of gradient orientation bins to use when calculating the histogram for each cell. In this case, orientations=6 means that the gradients will be quantized into 6 orientation bins.

pixels\_per\_cell: This parameter specifies the size of each cell in pixels. In this case, pixels\_per\_cell=(16, 16) means that each cell will be 16 pixels wide and 16 pixels tall.

cells\_per\_block: This parameter specifies the number of cells per block. In this case, cells\_per\_block=(3, 3) means that each block will be 3 cells wide and 3 cells tall.

The hog\_features function likely takes in the image as input and returns a feature vector that represents the image. The feature vector is likely calculated by following the steps I mentioned in my previous answer, including gradient computation, cell formation, histogram calculation, block normalization, and feature vector concatenation.

**Model Selection/Training Module:**

**SVM**

We used svm classifier with a polynomial kernel of degree 4 and a low regularization parameter, which may result in a more generalized model. The coef0 parameter is set to 1, which gives more importance to higher-degree polynomials. The gamma parameter is set to 0.1, which may result in a more complex decision boundary that is sensitive to changes in the input data. The random\_state parameter is set to 0 to ensure reproducibility of the results.

Model parameter:

kernel='poly': This specifies that the SVM should use a polynomial kernel function.

C=0.1: This is the regularization parameter, which controls the trade-off between minimizing the training error and minimizing the complexity of the model. A smaller value of C results in a more regularized model that prefers simpler decision boundaries.

random\_state=0: This is the random seed used by the random number generator. Setting a fixed value for random\_state ensures that the results are reproducible.

coef0=1: This is a parameter that determines the importance of high-degree polynomials in the kernel function. A higher value of coef0 gives more importance to higher-degree polynomials.

degree=4: This is the degree of the polynomial kernel function. A higher value of degree allows the SVM to fit more complex decision boundaries.

gamma=0.1: This is a parameter that controls the sensitivity of the SVM to changes in the input data. A smaller value of gamma results in a smoother decision boundary.

**Random forest:**

this code creates a random forest classifier object with 100 decision trees and a fixed random seed of 0. The RandomForestClassifier() function also has other optional parameters that can be used to further customize the random forest, such as max\_depth to control the depth of the decision trees, max\_features to control the number of features considered for each split, and criterion to specify the function used to measure the quality of each split.

Model parameter:

n\_estimators=100: This is the number of decision trees to create in the random forest. A higher value of n\_estimators typically results in better performance, but also increases the computational cost. The default value is 100.

random\_state=0: This is the random seed used by the random number generator. Setting a fixed value for random\_state ensures that the results are reproducible.

**Voting algorithm:**

**Note: We implemented this code but didn’t use this classifier because we found out that svm comes out with larger accuracy**

**Performance Analysis Module:**

for performance analysis model we used many approaches to calculate the model performance

Confusion Matrix: This is a table that is often used to evaluate the performance of a classification model. It is a matrix that summarizes the actual and predicted classifications made by a classifier on a set of test data. A confusion matrix displays the number of instances that are correctly and incorrectly classified by the classifier, organized by true class and predicted class.

Accuracy: This metric measures the proportion of correctly classified instances out of all instances. It is calculated as (TP + TN) / (TP + TN + FP + FN), where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

Precision: This metric measures the proportion of true positives out of all predicted positives. It is calculated as TP / (TP + FP).

Recall: This metric measures the proportion of true positives out of all actual positives. It is calculated as TP / (TP + FN).

F1 score: This metric is the harmonic mean of precision and recall and is a good measure of overall performance. It is calculated as 2 \* (precision \* recall) / (precision + recall).

**Enhancements and Future work:**

Hand gesture recognition is an important application of computer vision and has various potential applications, such as in virtual reality, robotics, and human-computer interaction. Here are some possible enhancements and future work that can be done in the field of hand gesture recognition:

Improvement of accuracy: One of the main challenges in hand gesture recognition is achieving high accuracy, especially for complex gestures with subtle variations. Future work can focus on developing more accurate algorithms by using advanced machine learning techniques such as deep learning and convolutional neural networks.

Real-time gesture recognition: Real-time gesture recognition is important for many applications, such as gaming and virtual reality. Future work can focus on developing algorithms that can recognize gestures in real-time using efficient algorithms and hardware.

Robustness to lighting and background variations: Hand gesture recognition systems can be affected by variations in lighting and background, which can lead to incorrect classification. Future work can focus on developing algorithms that are robust to lighting and background variations by using advanced image processing techniques.

Multi-modal gesture recognition: Multi-modal gesture recognition involves recognizing gestures using multiple sources of input, such as depth sensors, cameras, and microphones. Future work can focus on developing algorithms that can integrate multiple sources of input to improve the accuracy and robustness of the gesture recognition system.

Gesture recognition for sign language: Hand gesture recognition can be used for sign language recognition, which can help improve communication for the hearing-impaired. Future work can focus on developing algorithms that can recognize sign language gestures with high accuracy.

Gesture recognition for robotics: Hand gesture recognition can be used for controlling robots and other devices. Future work can focus on developing algorithms that can recognize hand gestures and translate them into commands for controlling robotic devices.

Large-scale gesture datasets: The availability of large-scale gesture datasets can help improve the accuracy and robustness of hand gesture recognition algorithms. Future work can focus on developing large-scale datasets that include a wide range of hand gestures and variations in lighting and background.