Article

ROS System Facial Emotion Detection Using Machine Learning for a Low-Cost Robot Based on Raspberry Pi

Javier Martínez [1,†] and Julio Vega [2,*,†]

[1] Department of Welfare on Animal Research, University of Alcalá, Pza. San Diego s/n, 28801 Alcalá de Henares, Spain
[2] Department of Telematic Systems and Computing, Rey Juan Carlos University, Camino del Molino n.º 5, 28942 Fuenlabrada, Spain
[*] Correspondence: julio.vega@urjc.es; Tel.: +34-91-488-4731
[†] These authors contributed equally to this work.

Abstract: Facial emotion recognition (FER) is a field of research with multiple solutions in the state-of-the-art, focused on fields such as security, marketing or robotics. In the literature, several articles can be found in which algorithms are presented from different perspectives for detecting emotions. More specifically, in those emotion detection systems in the literature whose computational cores are low-cost, the results presented are usually in simulation or with quite limited real tests. This article presents a facial emotion detection system—detecting emotions such as anger, happiness, sadness or surprise—that was implemented under the Robot Operating System (ROS), Noetic version, and is based on the latest machine learning (ML) techniques proposed in the state-of-the-art. To make these techniques more efficient, and that they can be executed in real time on a low-cost board, extensive experiments were conducted in a real-world environment using a low-cost general purpose board, the Raspberry Pi 4 Model B. The final achieved FER system proposed in this article is capable of plausibly running in real time, operating at more than 13 fps, without using any external accelerator hardware, as other works (widely introduced in this article) do need in order to achieve the same purpose.

Keywords: ROS; low-cost; raspberry Pi; visual attention; facial emotion detection; human–robot interaction

check for updates

Citation: Martínez, J.; Vega, J. ROS System Facial Emotion Detection Using Machine Learning for a Low-Cost Robot Based on Raspberry Pi. *Electronics* **2023**, *12*, 90. https://doi.org/10.3390/electronics12010090

Academic Editors: Juan M. Corchado, Byung-Gyu Kim, Carlos A. Iglesias, In Lee, Fuji Ren and Rashid Mehmood

Received: 22 November 2022
Revised: 12 December 2022
Accepted: 21 December 2022
Published: 26 December 2022

1. Introduction

Service robotics is a broad area of research within robotics that is characterized by offering useful services to humans [1,2]. Recently, an exponential boom was achieved in fields such as hospitality, with customer service robots, and in the health field, with companion robots. Studies were conducted that revealed that mascot-shaped robots improve the level of arousal of dementia patients and reduce their levels of depression [3], as well as improve their facial expressions (affect) and communication with care staff (social interaction) [4]. For these robots, it is essential to empathize and interact correctly with the humans with whom they are dealing; that is why the field of study of human–robot Interaction, or HRI, was born, with the need for robots to be able to collaborate and live with humans [5,6].

Artificial vision plays a fundamental role in this area, which, added to machine learning, provides the robot with qualities such as recognition and location of hands [7] or prediction of emotions [8], providing it with capabilities that allow it to interact appropriately according to the context. In [9] the authors showed how, in any given interaction, 7% is conveyed through words, 38% through tone, and 55% through facial expressions; therefore, the correct identification of these expressions is crucial. In [10], authors proposed a strategy to create datasets with images/videos to validate the estimation of emotions in scenes and personal emotions. Both datasets were generated in a simulated environment based on the Robot Operating System (ROS) from videos captured by robots through their sensory capabilities.

However, computer vision tasks that use machine learning usually require a high computational load, and not all robot software development environments have enough money, or even space inside their robots, to do this with a large computing station. Therefore, recently there was a great boom in research to develop algorithms and lightweight applications capable of running on low-cost embedded systems.

This article presents a facial emotion detection system—detecting emotions such as anger, happiness, sadness or surprise—that was implemented under the Robot Operating System (ROS), Noetic version, and is based on the latest machine learning (ML) techniques proposed in the state-of-the-art. To make these techniques more efficient, and so that they can be executed in real time on a low-cost board, extensive experiments were conducted in a real-world environment using a low-cost general purpose board, the Raspberry Pi 4 Model B. The final achieved FER system proposed in this article is capable of plausibly running in real time, operating at more than 13 fps, without using any external accelerator hardware, as other works (widely introduced in this article) do need in order to achieve the same purpose. The ROS package is publicly available at https://github.com/jamarma/emotion_detection_ros (accessed on 1 December 2022).

The three key aspects of the system proposed in this work are described below:

- An extensive state-of-the-art study was conducted, which is described in Section 2, so as to incorporate into this work the most cutting-edge machine learning techniques for the image processing, specifically for the recognition of facial emotions.
- The system, described in Section 3, is easily usable by anyone because, first, it was implemented under the most widely used robotics programming environment worldwide, the Robot Operating System (ROS), as a ROS package, which is publicly available at https://github.com/jamarma/emotion_detection_ros (accessed on 1 December 2022); and, second, because the experiments were performed so that the system works on a low-cost general-purpose board, affordable for almost anyone, and widely used and acquirable worldwide, such as the Raspberry Pi 4.
- Several and extensive experiments, which are summarized in Section 4, were designed and conducted so that the system, integrating the latest ML algorithms, is capable of plausibly running in real time on a low-cost platform. These experiments are publicly available at https://github.com/RoboticsURJC/tfg-jmartinez/wiki (accessed on 1 December 2022).

2. Related Work

On the one hand, regarding the low-cost hardware approach, some studies—that are based specifically on the Raspberry Pi as main board—can be highlighted. Many applications can be found where Raspberry Pi devices are programmed along with ML as predictive models and proposed as low-cost hardware solutions. These common applications include image classification, occupancy detection, fault detection, IoT security, and smart energy management as promising fields. Some of these applications are described below.

In [11], the authors introduced a new image classification system based on the porting of the Rulex ML platform on the board to perform ML forecasts in an Internet of things (IoT) setup. Specifically, they explained the porting Rulex's libraries on Windows 32 Bits, Ubuntu 64 Bits, and Raspbian 32 Bits. With the aim of performing an in-depth verification of the application possibilities, they proposed to perform forecasts on five unrelated datasets from five different applications, having varying sizes in terms of the number of records, skewness and dimensionality.

An image classification system based on a low-cost system was presented in [12]. In this study, the authors proposed a scalable and less-intrusive occupancy detection method that leverages existing BLE technologies found in smartphone devices to perform zone-level occupancy localization, without the need for a mobile application. The proposed method uses a network of BLE beacons for data collection before passing the pre-processed data into a machine learning model to infer the occupants' zone-level location. A similar approach by the same authors can be found in [13], where an IoT-based occupancy-driven plug load management system in smart buildings is introduced.

Another application of a Raspberry Pi as the main board can be found in [14], where authors presented an analysis and comparison of the performances achieved by machine learning techniques for real-time drift fault detection in sensors using a Raspberry Pi. The machine learning algorithms under observation included an artificial neural network, a support vector machine, a naïve Bayes classifier, k-nearest neighbors and a decision tree classifier. Drift fault was injected into the normal data using an Arduino Uno microcontroller.

A low-cost system can also be used as a secure IoT system. In [15], to address the difficulties in verifying IoT gadgets in ML, the authors proposed using AI inside an IoT door to help secure the framework. They explored the use of artificial neural networks in a portal to distinguish oddities in the information sent from the edge gadgets. In this paper, they argued how this methodology can improve the security of IoT frameworks.

On the other hand, some recent studies that used deep learning to solve the facial emotion detection (FER) problem are described. In the article [16], three models of convolutional neural networks (CNN) were presented; the first is a fully convolutional surface network consisting of six convolution modules, the second is a double-branch CNN that extracts traditional features from LBP and deep learning in parallel and the third is a pre-trained CNN achieving accuracy results of up to 95.29% and 71.14% for the CK+ (The Extended Cohn–Kanade Dataset [17,18]) and FER2013 (https://www.kaggle.com/datasets/msambare/fer2013, accessed on 1 December 2022) datasets, respectively.

Another different approach to CNNs was proposed in the article [19], where a frequency neural network (FreNet) was presented, which, compared to a CNN, has the advantages of processing the image in the frequency domain, efficient calculation and the elimination of spatial redundancy, achieving accuracy values of up to 98.41% for the CK+ dataset. However, one of the major problems of FER is the intraclass variation caused by the identities of the subjects. In the article [20], a self-difference convolutional network (SD-CNN) was proposed to address this problem, obtaining accuracy values of 99.7% and 91.3% for the CK+ and Oulu-CASIA datasets, respectively.

In [21], the authors proposed a compact and robust service for Assistive Robotics, called Lightweight EMotion recognitiON (LEMON), which uses image processing, computer vision and deep learning (DL) algorithms to recognize facial expressions. Specifically, the proposed DL model is based on residual convolutional neural networks with a combination of dilated and standard convolution layers.

Finally, regarding the efforts made to create faster FER algorithms for low-cost hardware, many studies can be found in the literature. In [22], the authors presented an IoMT-based facial emotion detection and recognition system that was implemented in real-time by using a Raspberry-Pi with the assistance of deep convolution neural networks. The same authors proposed in [23] a Raspberry-Pi-based standalone edge device that can detect real-time facial emotions. This article was mainly crafted using a dataset of employees working in organizations. A Raspberry-Pi-based standalone edge device was implemented using the Mini-Xception Deep Network because of its computational efficiency in a shorter time compared to other networks.

In the article [24], a fast algorithm was proposed to monitor the emotions of a driver using a random forest classifier classifying geometric features extracted from dlib reference points, obtaining accuracy results of 92.6% and 76.7% for the CK+ and MMI datasets, respectively.

Continuing with the use of facial reference points to classify emotions, many articles can be found, such as [25] or [26], achieving accuracy results of up to 97% in the latter. Another method proposed in [27] is the use of Gabor filters using a lightweight Gabor convolutional network (GCN). Thanks to this, good precision results were obtained with low computational requirements because changes in emotions are strongly related to regions of interest (ROI) and Gabor filters are very efficient in extracting visual content. Results of 89.41% accuracy were obtained for the FERPlus dataset.

A hybrid model consisting of a CNN and k-nearest neighbors (KNN) algorithm for FER using transfer learning was presented in [28], by substituting the EfficientNet Softmax layer for the KNN. An accuracy value of 75.26% was obtained with the FER-2013 dataset, and the researchers managed to make it work in real time but using a Coral USB Accelerator.

In [29], an algorithm was proposed that starts by detecting faces using the Viola Jones detector [30]; such a region was processed using a Gabor filter and a median filter, ORB features were extracted and a support vector machine (SVM) classifier that predicts emotions was trained. Accuracy of 99.1% was obtained for the MMI dataset, but the algorithm has only been tested with images and not with videos. Therefore, its performance with image streams is unknown.

3. Emotion Detection System

This section describes the emotion detection system developed in this work, as well as its integration into ROS. After the study of the state-of-the-art, it was decided to use the following technique—consisting of three steps—for detecting emotions: detection of facial points, extraction of geometric information from those facial points, and classification of that information. It was decided to choose this technique due to the low computational cost it entails and the great precision it offers. Figure 1 shows a general scheme of the proposed methodology of the system, ignoring the integration in ROS, which is described in Section 3.3.

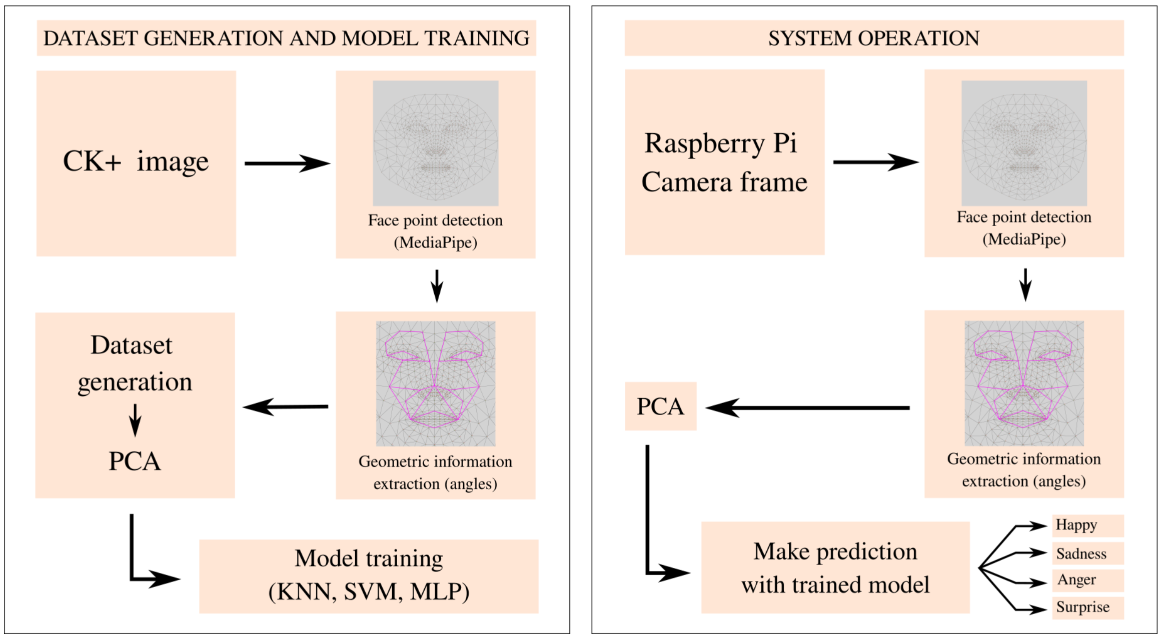
DATASET GENERATION AND MODEL TRAINING
CK+ image
Face point detection (MediaPipe)
Geometric information extraction (angles)
Dataset generation
PCA
Model training (KNN, SVM, MLP)
SYSTEM OPERATION
Raspberry Pi Camera frame
Face point detection (MediaPipe)
Geometric information extraction (angles)
PCA
Make prediction with trained model
Happy
Sadness
Anger
Surprise

Figure 1. Schema of the emotion detection system developed.

3.1. Face Point Detection and Geometric Information Extraction

The algorithm used to perform feature facial point detection is MediaPipe Face Mesh [31] (https://google.github.io/mediapipe/solutions/face_mesh.html, accessed on 1 December 2022). In Section 4.1, a study was made comparing the two most used libraries to conduct such a task (dLib and MediaPipe) and the result was that the one with the best performance and accuracy offered on the hardware used was MediaPipe, so it is the library used on the system. It provides a performance of 13.28 fps on average and provides 3D coordinate information of 468 facial points.

After obtaining data in coordinates with the characteristic facial points of a face, they need to be processed to obtain geometric information about the possible emotions that are occurring in the current face. The method chosen in this work is a novel technique proposed in the article [26]. This consists of building an *emotional mesh* using the points provided by MediaPipe (Figure 2), based on the facial coding system or FACS [32,33]. The *emotional mesh* provides information on angles of the facial expressions.

FACS is a system that classifies human facial movements based on changes in the face caused by muscle movements. These movements are defined as action units or AUs, of which there are up to 46. In Table 1, the most used AUs are displayed with their respective descriptions.

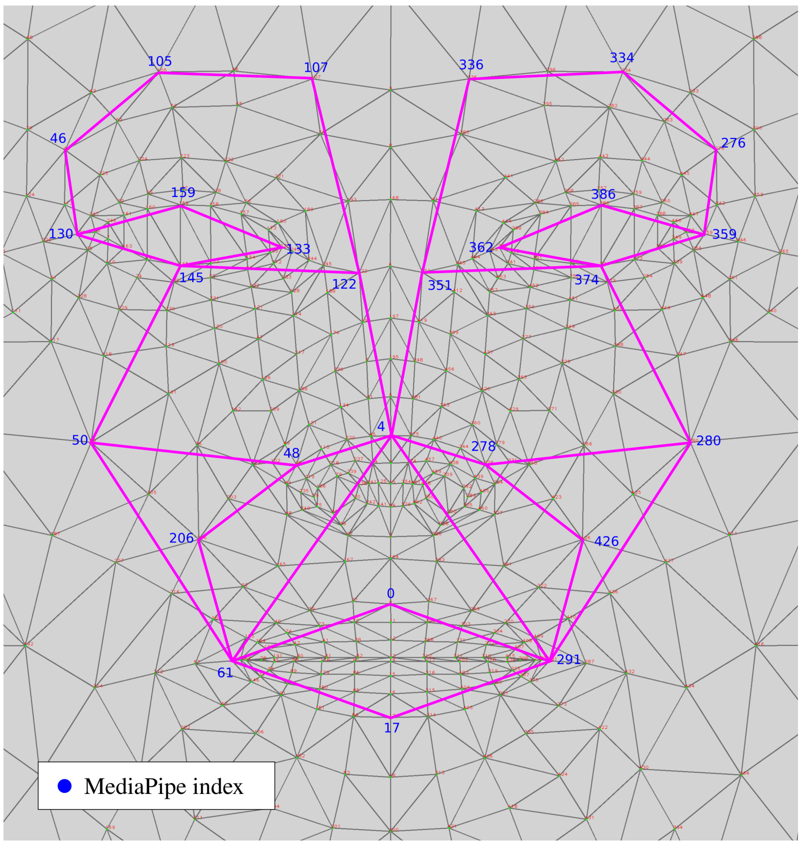

Figure 2. *Emotional mesh* proposed in [26].

The EMFACS system (emotional facial action coding system) describes simple emotions using combinations of AUs (Table 2). Therefore, assuming this, it can be affirmed that the movement of each of these facial muscles is related to seven simple emotions, and that is why each location of the points of the *emotional mesh* (Figure 2) was chosen in such a way that it is affected by an UA and in this way achieves better recognition of emotions.

Table 1. List of the most used action units and their respective FACS descriptions.

AU	FACS Descriptions
1	Inner brow raiser
2	Outer brow raiser
4	Brow lowerer
5	Upper lid raiser
6	Cheek raiser
7	Lid tightener
9	Nose wrinkler
12	Lip corner puller
15	Lip corner depressor
20	Lip stretcher
23	Lip tightener
26	Jaw drop

Table 2. List of simple emotions in terms of AUs.

Emotion	AU
Happiness	6 + 12
Sadness	1 + 4 + 15
Surprise	1 + 2 + 5 + 26
Fear	1 + 2 + 4 + 5 + 7 + 20 + 26
Anger	4 + 5 + 7 + 23
Disgust	9 + 15 + 17
Contempt	12 + 14

Each point of the *emotional mesh* is joined to others by using edges, thus forming a closed mesh of 27 vertices and 38 edges. These edges form angles between them and these will be used as information to classify emotions. Taking advantage of the fact that all edges form triangles with each other (Figure 3) where p_2, p_3 and p_4 are 2D points of the *emotional mesh* and l_1, l_2 and l_3 are edges of the *emotional mesh*, to calculate the desired angles (α) the cosine theorem (Equation (1)) is used, which uses the length of the edges to perform the calculations, this is the distance between the two points that make up the edge. This distance is calculated by Euclidean distance (Equation (2)). The variables in Equations (1) and (2) are as shown in Figure 3.

$$\alpha = \arccos \frac{l_1^2 + l_3^2 - l_2^2}{2 l_1 l_3} \tag{1}$$

$$l_3 = \sqrt{(p_{4x} - p_{2x})^2 + (p_{4y} - p_{2y})^2} \tag{2}$$

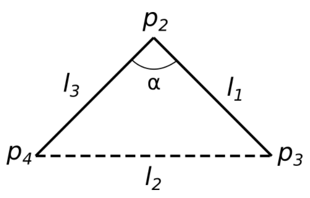

Figure 3. Example of triangle formed by 2 edges of the *emotional mesh*.

3.2. Dataset Generation and Model Training

Once the facial point angles are extracted (Section 3.1), the objective is to generate a dataset with these angles. To do that, the algorithm takes an images dataset from which geometric information is extracted using the *emotional mesh* and, therefore, a new database of angles is generated.

The image dataset used was CK+. This database contains 593 image sequences, showing a face from a neutral position to full expression. Of these 593 sequences, 327 have the last *frame* labeled with an emotion between 1 and 7 (1 = anger, 2 = contempt, 3 = disgust, 4 = fear, 5 = happiness, 6 = sadness, 7 = surprise). These last *frames* were the ones used in this work.

CK+ was chosen over other popular datasets due to the alignment of the faces in the images. Databases such as FER-2013 are focused on being used by CNNs and the position of the faces does not influence the training; moreover, it provides more generality. However, to perform the technique proposed in this article, it is necessary to have aligned faces to obtain reliable information on the angles formed by facial expressions. The CK+ dataset also does not contain faces in different light conditions, but this is not necessary because this dataset is used just to build another dataset with angular data that are invariant even if the light conditions of the faces change. In our system, the detection of faces and facial points, which provide us with angular data, is performed using MediaPipe FaceMesh (Section 3.1). This model designed and trained by Google is very robust in different situations.

Several studies were conducted and are introduced in Sections 4.2–4.4, with the objective to find the number of characteristics and classes that offer the best precision in terms of training and final performance. Finally, the dataset that has obtained the best results and, therefore, the one used to train the system, is the one comprised of four emotions (anger, happiness, sadness, surprise) and 21 angles for each sample. Table 3 shows the structure of the generated dataset. The rows represent the data extracted from each CK+ image; the columns from X0 to X20 represent each angle obtained using the *emotional mesh* and the *y* column refers to the type of class: anger, happiness, sadness or surprise.

Table 3. Dataset used in the project.

X0	X1	X2	...	X19	X20	y
54.288044	37.570711	154.655589	...	44.625203	63.887010	1.0
44.670597	35.229102	148.630240	...	47.334403	61.278073	1.0
46.613914	36.808837	161.148375	...	57.291823	64.390395	1.0
49.404349	47.407905	153.817836	...	49.880184	61.894869	1.0
42.510847	43.626048	146.891826	...	46.965439	59.971707	1.0
...	...	...	...	...	...	...
23.444336	97.667648	88.384186	...	28.011004	63.905389	7.0
24.634940	96.406625	93.413763	...	28.637606	63.551521	7.0
22.425106	105.319774	87.727242	...	30.811141	61.646881	7.0
15.966920	120.968600	60.790043	...	28.018767	56.442379	7.0
19.667632	104.568158	80.332991	...	29.088510	64.107810	7.0

Training was performed with three different classification algorithms and the results of each of them were compared. The algorithms used were KNN, SVM and a multi-layer perceptron (MLP). These algorithms are three of the most popular ML algorithms within the state-of-the-art of this study: KNN is used in article [28], SVM is used in article [23], and SVM, KNN and MLP are the most accurate algorithms compared to others such as naïve Bayes (NB) or decision trees (DT) in article [14]. Therefore, these three algorithms were considered optimal ones in terms of performance and, for this reason, they were the algorithms chosen to be used and compared in this article. Additionally, principal component analysis (PCA) was used to reduce the number of features in the dataset and, thus, improve the accuracy of emotion detection. To find the optimal number of neighbors for KNN, the optimal regulation parameter for SVM, the optimal number of layers and neurons for MLP, and the optimal number of components to be maintained by PCA, the K-fold stratified cross-validation was used with four *folds* (Figure 4). The class with the least data in the dataset has a quantity of 28 and, therefore, with 4 folds, 7 data points of that class in each fold are granted.

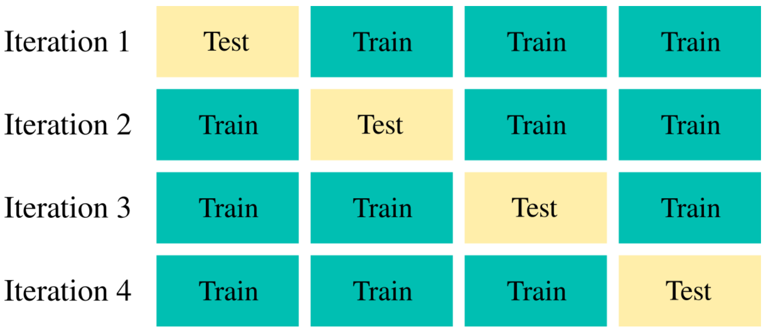

Figure 4. Splitting the database into 4 *folds* using K-fold.

An example of a simplified use of K-fold to look up parameters for KNN and PCA is found in Figure 5. The system proposed in this article was tested for PCA with numerous components between 2 and 20 for KNN, with odd values of *k* between 1 and 13 inclusive, for SVM with values of C between 1 and 999 (jumps of 10 in 10) and for MLP with 1 hidden layer of neurons between 5 and 24 included, and it was not necessary to add more layers.

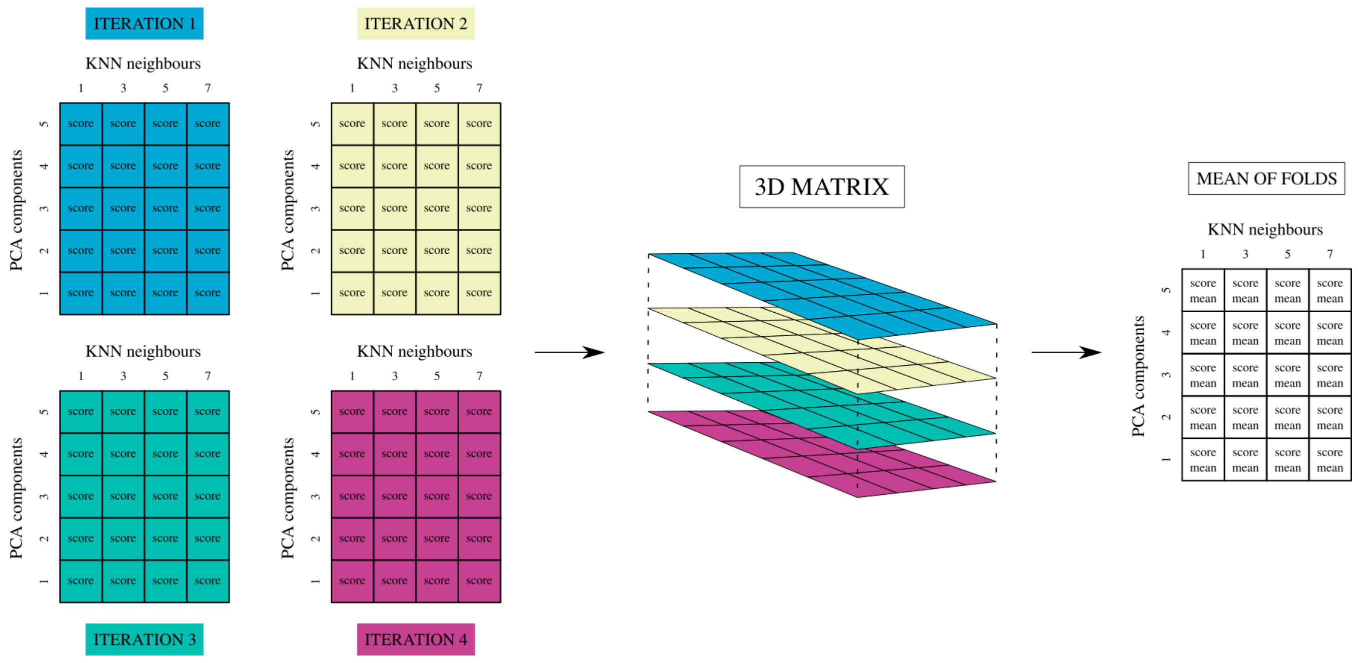

Figure 5. Example of finding the optimal number of neighbors for KNN and the optimal number of components to be maintained by PCA using K-fold.

The results of the search for optimal parameters for each algorithm and the results of the training with these optimal parameters are given in Table 4. The accuracy results have also been calculated by K-fold stratified cross-validation of four folds, showing the average of the four folds.

Table 4. Optimal parameters and precision results for each of the classifiers.

Classifier	Accuracy	Precision	Recall	F1-Score	Optimal Parameters
KNN y PCA	0.95	0.93	0.94	0.92	k = 7, n_components = 11
SVM y PCA	0.95	0.93	0.92	0.92	C = 21, n_components = 11
MLP y PCA	0.95	0.95	0.93	0.93	hidden_layer_sizes = (17), n_components = 11

3.3. System Integration in ROS

The last goal is to create a tool that abstracts the developer from the system code. The data detected by the system can easily be obtained by using ROS *topics*. To do this, ROS Noetic was installed on a Raspberry Pi board and a package called *emotion_detection_ros* (https://github.com/jamarma/emotion_detection_ros, accessed on 1 December 2022) was developed. Figure 6 shows a general scheme of the operation of the system already integrated in ROS.

The *emotion_detection* node receives the *frames* captured by the Raspberry Pi through the *topic /raspicam_node/image/compressed*, which handles messages of the type *sensor_msgs/ CompressedImage*. This node encapsulates all of the processing related to emotion detection (it can be observed that it contains the same as shown in Figure 1) and, through the *topic/emotion_detection_ros/bounding_boxes*, which handles messages of the type *emotion_detection_ros_msgs/BoundingBoxes*, sends a list of *emotion_detection_ros_msgs/BoundingBox*. Each of these last messages contains the information of each emotion detected: a value of type *float64* with the probability of the prediction, four fields of type *int64* with the coordinates of the bounding box that surrounds the emotion detected, and a *string* with the emotion detected.

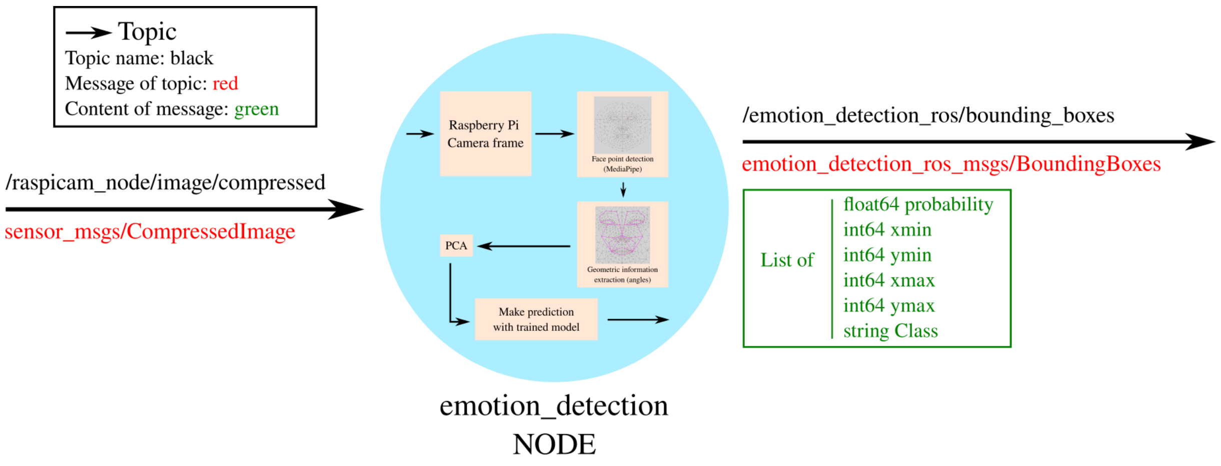
Topic
Topic name: black
Message of topic: red
Content of message: green
/raspicam_node/image/compressed
sensor_msgs/CompressedImage
Raspberry Pi Camera frame
Face point detection (MediaPipe)
Geometric information extraction (angles)
PCA
Make prediction with trained model
emotion_detection NODE
/emotion_detection_ros/bounding_boxes
emotion_detection_ros_msgs/BoundingBoxes
List of
float64 probability
int64 xmin
int64 ymin
int64 xmax
int64 ymax
string Class

Figure 6. General scheme of the ROS package.

Full instructions for using it can be found in the *README* of the *emotion_detection_ros* (https://github.com/jamarma/emotion_detection_ros, accessed on 1 December 2022) repository. To use the Raspberry Pi camera in an ROS environment, the *raspicam_node* (https://github.com/UbiquityRobotics/raspicam_node, accessed on 1 December 2022) package was used, which is responsible for efficiently reading *frames* from the Raspberry Pi camera and publishing them to the *topic /raspicam_node/image/compressed*. After compiling and launching the system, following the instructions, a graphic window is opened, showing the *frames* where the bounding boxes will be drawn with their respective labels displaying the prediction. In addition, the system frame-rate value will be displayed in the upper left corner. This and other examples are shown in the *emotion_detection_ros* official repository (https://github.com/jamarma/emotion_detection_ros, accessed on 1 December 2022).

4. Experiments

This section describes the experiments conducted during the development of the system with the aim of optimizing its different parts. Furthermore, the frame-rate performance of the emotion detection system running on ROS is shown in Section 4.5.

The platform on which the system runs and, therefore, the one used to conduct these experiments, is the Raspberry Pi 4 Model B together with the Raspberry Pi Camera Module V2.1, its official camera, and all under the official Raspberry Pi Operating System, to maximize performance. The ROS version used was ROS Noetic. The programming language used was Python 3.7.3. The library used to extract facial points was *mediapipe-rpi4* 0.8.8, and *scikit-learn* to make use of machine learning algorithms. In addition, other libraries, such as *numpy* 1.21.6, *opencv-python* 4.5.5.64, *pandas* 1.3.5 and *matplotlib* 3.5.2, were also used to support the development of the system.

As detailed in Section 4.1, studies were conducted with the aim of detecting facial points in real-time on a Raspberry Pi, successfully achieving a value of 13.28 fps, which could be considered as a real-time execution, as explained in Section 4.5.

As detailed in Sections 4.2–4.4, extensive experiments were conducted with the aim of achieving a high level of robustness so that the proposed system can be used in a real robot.

The performance tests are described in Section 4.5, and were conducted to demonstrate that the system proposed in this work runs plausibly in real-time under ROS and it achieved a better frame-rate than other state-of-the-art Raspberry Pi-based systems.

Regarding the accuracy value, in the article [26], the authors show a novel technique that was tested using different general-purpose datasets, obtaining an accuracy value of 0.94–0.97, for CK+ with SVM, KNN and MLP classifiers. However, this technique was implemented at the beginning of this work, obtaining an accuracy value of 0.79 for CK+, which is a low value to be used in a real robot. Therefore, the experiments described in Sections 4.2–4.4 were performed to optimize the technique as much as possible so as to improve the accuracy value up to 0.95 for all classifiers (SVM, KNN and MLP).

4.1. Search for the Optimal Facial Point Detector

A study was conducted to compare two facial point extraction libraries capable of working on the system platform, obtaining conclusions about which one offers better performance and accuracy. These two libraries are dlib and MediaPipe.

Tables 5 and 6 show performance tests for the dlib and Mediapipe libraries, using a resolution of 640 × 480 pixels, to study which one gives the highest frame-rate. For this purpose, the algorithms were evaluated for 30 s, with different implementations, and the FPS average obtained was calculated.

Table 5. Performance tests for different implementations of dlib.

Test Number	Implementation of Algorithm	Average FPS
Test 1	HOG and Linear face detector. dlib feature detector.	0.83
Test 2	OpenCV Face Detector (Viola). dlib feature detector.	6.28
Test 3	Test 2 detectors. Divide the reading of the frames from their processing, using threads.	8.21

Table 6. Performance tests for different implementations of MediaPipe.

Test Number	Implementation of Algorithm	Average FPS
Test 1	Use of the algorithm in the general way recommended in the official documentation.	5.91
Try 2	Split reading frames from processing them, using threads.	7.80
Try 3	Without drawing the facial mesh in the rendered frames.	13.28

Second, it was studied which of the two algorithms is more robust running on the hardware platform used and, therefore, is less prone to failures. Table 7 shows different tests that were conducted considering two failure situations: *false positives* and *no face detection*. It was assumed that there is always one face in every *frame*, so a *false positive* means there is more than one face, and *no detection* means there is none. However, MediaPipe can avoid false positives by setting the algorithm to only detect one face. Each *frame* was evaluated for 30 seconds against these criteria, and failures were counted.

Table 7. Built-in for dlib and MediaPipe.

Test	dlib Failures	MediaPipe Failures
Good lighting conditions	81	0
Poor lighting conditions	126	0
Partially covered faces	188	103
Rotated faces	183	0

Finally, it can be concluded that the algorithm that offers the best performance and accuracy on the hardware and software platform used is MediaPipe. This algorithm achieved a performance of 13.28 fps on average, compared to 8.21 fps for dlib and OpenCV. Additionally, the latter obtained 578 failures, compared to 103 failures for MediaPipe.

4.2. Study of the Most Influential Angles in Each Emotion

A study was conducted to discover which are the angles that most influence each emotion, that is, the angles that vary the most each time these facial expressions are produced. The aim of this study was obtaining information to build a reliable dataset of angles. To perform this test, the research was started using all of the angles of the right part of the face (Figure 7), and the variation in each of them and for each CK+ emotion was calculated, from a neutral position of the face to the maximum expression. This calculation was performed for each CK+ image and the results were the calculated means. The results of the study can be found in Figure 8.

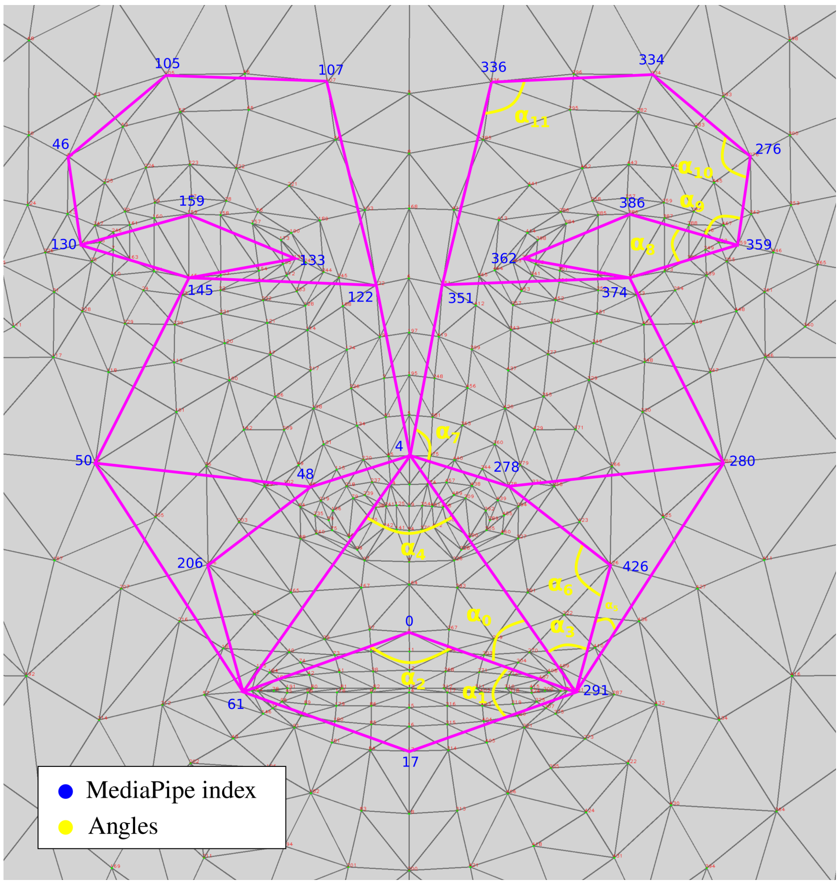

Figure 7. All angles on the right side of the *emotional mesh*.

As conclusions of this study, it is clear that the five most influential angles in each emotion are those shown in Table 8, and therefore, the angles that vary the most in total are: 2, 12 , 1, 16, 15, 4, 6, 14, 18, 8, 0 and 19.

Table 8. List of the 5 most influential angles for each emotion.

Emotion	Influential Angles
Happiness	4, 2, 18, 12, 8
Sadness	2, 1, 12, 16, 14
Surprise	1, 2, 0, 4, 19
Fear	2, 4, 6, 1, 14
Anger	2, 12, 1, 16, 15
Disgust	12, 16, 2, 15, 1
Contempt	2, 1, 4, 0, 19

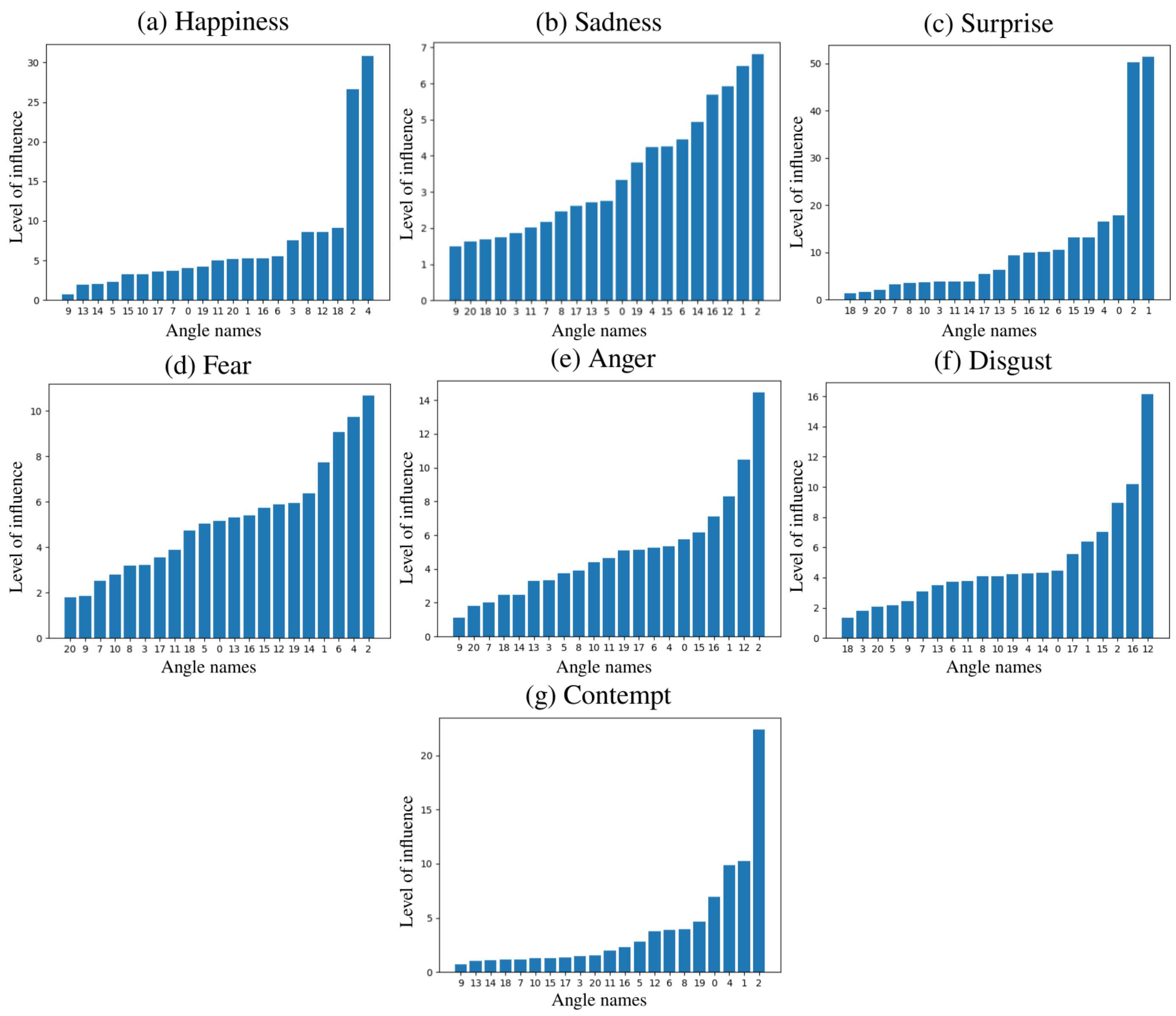

Figure 8. Study of the most influential angles for each emotion.

4.3. Study of the Symmetry of Emotions

With the aim of discovering if the angles of one half of the face could be dispensed with, a symmetry study was conducted, in which is checked whether the emotions can be considered symmetrical for both sides of the face or not. To do this, the study started from the angles obtained in the previous study (the most influential) and studied the variation that exists between the left and right sides of the face for each of these angles in each emotion. This procedure was performed on each image from the CK+ dataset and the results are the mean of what was obtained for each of them (Figure 9).

Observing the results, it can be considered that the emotions are symmetrical, since the greatest average variation between the left and right sides was five degrees, this being a minimal variation. Therefore, knowing the information from the two studies, all of the angles of one half of the face can be ruled out.

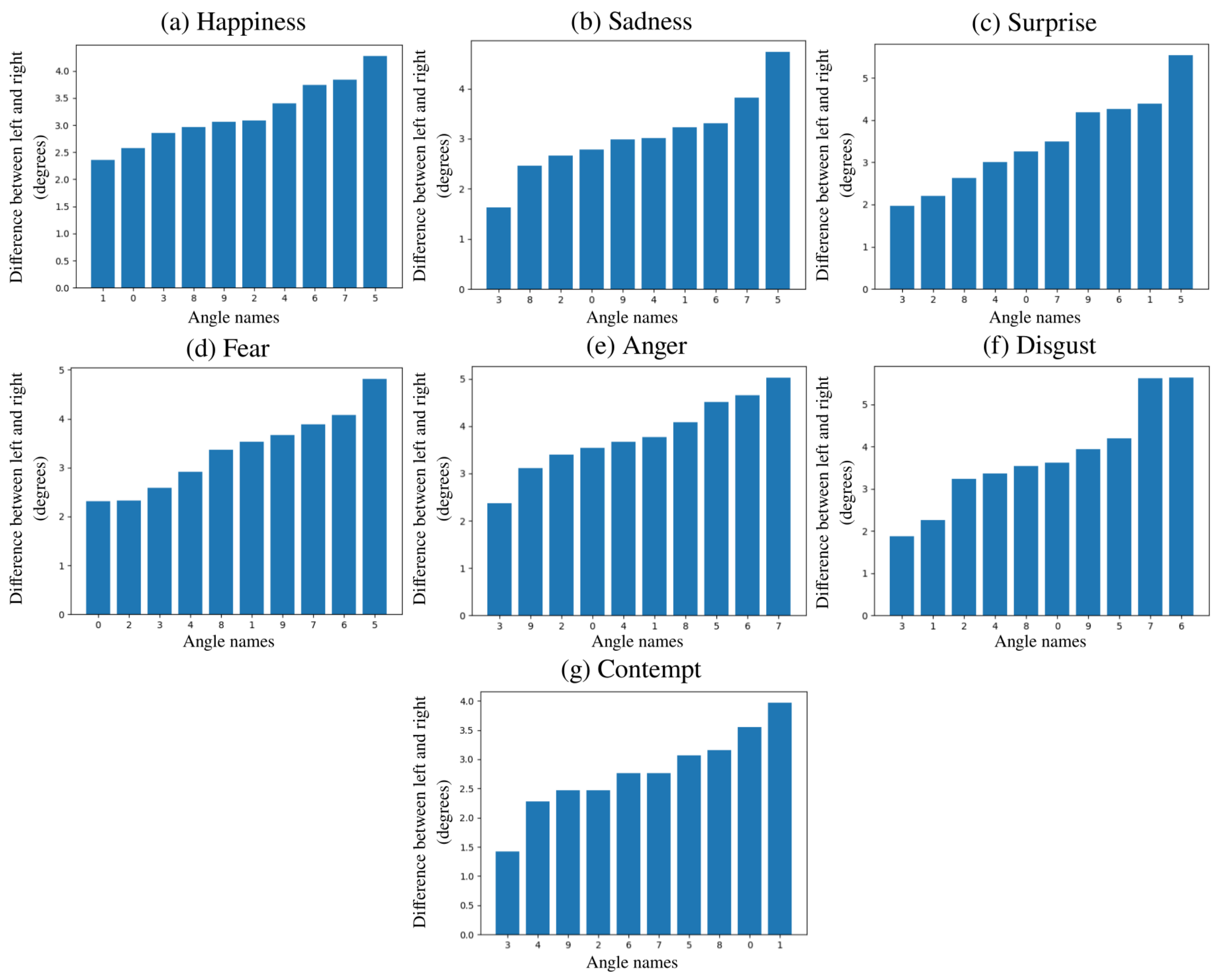

Figure 9. Symmetry study for each emotion.

4.4. Finding the Optimal Dataset for Training

Training tests were performed with several datasets to study which one offers the best results and, therefore, helps create a more robust system. This section shows what those test datasets were and the results of their training.

4.4.1. Dataset with the Most Influential Angles from the Right Side of the Face (*dataset1*)

First, a dataset with only the most influential angles of the right part of the *emotional mesh* was generated, to check if training the algorithm with this number of characteristics may be optimal since they are the ones that should have the most information. The search for optimal parameters for each classification algorithm was performed in the same way as explained in Section 3.2, as well as the calculation of precision in the training results. All of the data are shown in Table 9.

Table 9. Optimal parameters and precision results for each classifier on *dataset1*.

Classifier	Accuracy	Precision	Recall	F1-Score	Optimal Parameters
KNN	0.82	0.76	0.74	0.74	k = 11
SVM	0.84	0.79	0.78	0.77	C = 11
MLP	0.82	0.78	0.78	0.77	hidden_layer_sizes = (19)

4.4.2. Dataset with All Right Angles and Reduced with PCA (*dataset2*)

We performed a test with this dataset comprising the angles of the right side of the face to study whether it offers better results than reducing the features manually as was done in *dataset1*, or using automatic PCA. The search for optimal parameters for each classification algorithm was performed in the same way as explained in Section 3.2, as well as the calculation of precision in the training results. All of the data are shown in Table 10.

Table 10. Optimal parameters and precision results for each classifier on *dataset2*.

Classifier	Accuracy	Precision	Recall	F1-Score	Optimal Parameters
KNN y PCA	0.83	0.79	0.75	0.75	k = 11, n_components = 17
SVM y PCA	0.84	0.80	0.77	0.78	C = 11, n_components = 7
MLP y PCA	0.84	0.81	0.80	0.80	hidden_layer_sizes = (12), n_components = 11

The results obtained for each of the two previous datasets are quite similar, although a little higher for *dataset2*, especially when taking a look at the *Precision*, *Recall* and *F1-score* parameters. As for the results obtained by each classifier, they are also quite similar and it would be very difficult to consider just one. However, in general, these are not excellent results and the robotic tool built from these models would not be very robust, especially when observing the results of each class separately in each K-fold iteration; for example, those returned by SVM using *dataset2* (Tables 11–14). In these results, it can be observed that classes such as *Contempt* or *Angry* have obtained poor scores, even below 50% in some cases. Therefore, this led us to conclude that the system will not perform well regarding detecting all emotions, although on average it will have an acceptable score.

This misclassification of some emotions is due to the great similarity that exists between some of them. This means that, geometrically, they are practically indistinguishable and, therefore, it is difficult for algorithms to know how to classify them correctly. It can be observed, in the study of influential angles in each emotion (Figure 8, Table 8), that the emotions *surprise* and *contempt* have the same five influencing angles, and—similarly—the emotions *anger* and *disgust* also have the same five angles. This makes the classification very difficult to perform with a high degree of reliability.

Table 11. SVM training results for iteration 1 of K-fold stratified using *dataset2*.

Class	Precision	Recall	F1-Score
Anger	0.67	0.50	0.57
Contempt	0.67	0.50	0.57
Disgust	0.76	0.87	0.81
Fear	0.67	1.00	0.80
Happiness	0.94	0.94	0.94
Sadness	0.83	0.71	0.77
Surprise	0.95	0.95	0.95

Table 12. SVM training results for iteration 2 of K-fold stratified using *dataset2*.

Class	Precision	Recall	F1-Score
Anger	0.71	0.45	0.56
Contempt	0.57	0.80	0.67
Disgust	0.76	0.93	0.84
Fear	1.00	0.71	0.83
Happiness	0.85	1.00	0.92
Sadness	0.83	0.71	0.77
Surprise	1.00	0.95	0.98

Table 13. SVM training results for iteration 3 of K-fold stratified using *dataset2*.

Class	Precision	Recall	F1-Score
Anger	0.75	0.82	0.78
Contempt	0.67	0.80	0.73
Disgust	0.82	0.93	0.87
Fear	0.80	0.67	0.73
Happiness	0.94	0.94	0.94
Sadness	1.00	0.57	0.73
Surprise	0.95	0.95	0.95

Table 14. SVM training results for iteration 4 of K-fold stratified using *dataset2*.

Class	Precision	Recall	F1-Score
Anger	0.67	0.55	0.60
Contempt	0.50	0.25	0.33
Disgust	0.67	0.80	0.73
Fear	1.00	0.67	0.80
Happiness	1.00	1.00	1.00
Sadness	0.50	0.71	0.59
Surprise	1.00	1.00	1.00

Taking into account all of the above regarding the similarity of classes by observing the influential angles, it was proposed to eliminate the emotions that are intercepting others and, in this way, increase the robustness of the proposed system, achieving greater precision in this model. The emotions that were chosen are the following: *happiness*, *sadness*, *surprise* and *anger*. They are the four most general simple emotions and, in addition, they differ geometrically from each other. Assuming this, the winning dataset, and therefore the one used to train the system, is *dataset2*, but only with the emotions of *happiness*, *sadness*, *surprise* and *anger*. All of the results of this training are shown in Section 3.2.

4.5. System Performance

This section shows the final performance of the system and a comparison with other state-of-the-art systems. It, as mentioned at the beginning of Section 4, is running entirely on the Raspberry Pi 4 Model B and under the ROS Noetic middleware. A study of the frame-rate value thrown depending on the number of faces detected was conducted. Three tests were used, modifying the configuration parameter `max_num_faces` of the system, which allows customizing the maximum number of faces to be simultaneously detected. In this way, different tests were conducted with up to three faces. The results of the three situations are found in Tables 15–17, respectively.

Table 15. System performance in ROS with max_num_faces: 1.

Face Detected	Average FPS	Max FPS Value	Min FPS Value
1	13.18	15.13	11.04

Table 16. System performance in ROS with max_num_faces: 2.

Face Detected	Average FPS	Max FPS Value	Min FPS Value
1	11.11	11.97	9.18
2	7.53	8.35	6.96

Table 17. System performance in ROS with max_num_faces: 3.

Face Detected	Average FPS	Max FPS Value	Min FPS Value
1	11.05	11.72	9.41
2	6.77	7.61	6.15
3	5.37	6.93	5.02

The proposed system can be considered to be operating in real time because it works at 13.18 fps (Table 15), detecting emotions without suffering any delay perceptible by a human. Table 18 shows a comparison of the system proposed in this article with other state-of-the-art studies.

It can be observed that the proposed system obtained similar inference time results compared to other studies, but the system introduced in this work does not use any ML inferencing external accelerator board, such as Coral USB Accelerator or Intel Movidius NCS2, while the other state-of-the-art systems do. Therefore, the proposed system can be considered a real low-cost FER system.

Furthermore, the proposed system, with an accuracy value of 0.95, is more accurate than other Raspberry Pi-based systems. In article [29], the accuracy value was 0.9, whereas in article [28], the value was 0.7526.

Table 18. Comparison of the proposed system with previous studies.

Research	Hardware Device	Algorithm	Dataset	Accuracy (%)	Inference Time (ms)
[28]	Raspberry Pi 4 with Coral USB Accelerator	CNN + KNN	FER-2013	75.26	74.15
[29]	Raspberry Pi 3	Viola Jones face detector + ORB + SVM	CK+	90	Unmentioned
[23]	Raspberry Pi 3 B+	CNN + SVM	FER-2013	68	Unmentioned
[22]	Raspberry Pi with Intel Movidius NCS2	CNN + SVM	FER-2013	73	Unmentioned
Proposed	Raspberry Pi 4	CNN + (SVM or KNN or MLP)	CK+ (happiness, sadness, surprise, anger)	95	75.87

5. Conclusions

In this article, a facial emotion recognition system was developed, based on existing state-of-the-art techniques. This system arose with the aim of collaborating in the field of human–robot Interaction research, providing developers—with limited resources—a tool that usually requires a high computational load.

The proposed system is capable of working in real time on an embedded platform, such as Raspberry Pi 4, under the ROS middleware, and without using any ML inferencing external accelerator board, achieving a frame-rate similar to other state-of-the-art systems that do. Therefore, the proposed system can be considered the cheapest, most accurate and best performing state-of-the-art Raspberry Pi-based system to be used in a real robot.

Several performance tests were conducted to achieve a final system that can operate with single face detection at 13.18 fps, which let it be used in a real-time application. In addition, it achieved an average of 7.53 fps, for simultaneous two-face detections, and 5.37 fps, for simultaneous three-face detections. Furthermore, its integration with ROS provides a full abstraction of the code to any developer because the data detected in the frames are accessible using ROS *topics*.

The method chosen to perform the emotion detection provides a good performance under changing lightning conditions using very low-cost hardware. Nevertheless, some disadvantages can be considered; on the one hand, it is necessary to align the face to be detected in front of the camera; on the other hand, and due to the dataset used in this work, CK+, the system cannot detect a blank expression (which implies a lack of emotion), because this class is excluded in the training dataset.

Among the future developments of this line of research, it is worth mentioning the need to solve the abovementioned problems. It would also be interesting to extend the system with new functionalities, such as detecting the gender, or predicting the age of the subject. Finally, and continuing along the lines of providing the system to anyone who cannot afford a more expensive platform, the system could be ported to other low-cost platforms existing on the market.

Author Contributions: Conceptualization, J.V.; methodology, J.V.; software, J.M.; validation, J.V. and J.M.; formal analysis, J.V.; investigation, J.M.; resources, J.M.; data curation, J.V. and J.M.; writing—original draft preparation, J.V. and J.M.; writing—review and editing, J.V.; visualization, J.M.; supervision, J.V.; project administration, J.V.; funding acquisition, J.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by the Community of Madrid through the Jóvenes Investigadores project (ref. F664) of the Rey Juan Carlos University.

Data Availability Statement: All the data supporting reported results can be found at https://github.com/RoboticsURJC/tfg-jmartinez/wiki (accessed on 1 December 2022). The final ROS package can be found at https://github.com/jamarma/emotion_detection_ros (accessed on 1 December 2022).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Alonso, I.G. Service Robotics. In *Service Robotics within the Digital Home: Applications and Future Prospects*; Springer: Dordrecht, The Netherlands, 2011; pp. 89–114. [CrossRef]
2. Miseikis, J.; Caroni, P.; Duchamp, P.; Gasser, A.; Marko, R.; Miseikiene, N.; Zwilling, F.; de Castelbajac, C.; Eicher, L.; Fruh, M.; et al. Lio-A Personal Robot Assistant for human–robot Interaction and Care Applications. *IEEE Robot. Autom. Lett.* **2020**, *5*, 5339–5346. [CrossRef] [PubMed]
3. Lu, L.C.; Lan, S.H.; Hsieh, Y.P.; Lin, L.Y.; Lan, S.J.; Chen, J.C. Effectiveness of Companion Robot Care for Dementia: A Systematic Review and Meta-Analysis. *Innov. Aging* **2021**, *5*, igab013. [CrossRef] [PubMed]
4. Liang, A.; Piroth, I.; Robinson, H.; Macdonald, B.A.; Fisher, M.J.; Nater, U.M.; Skoluda, N.; Broadbent, E. A Pilot Randomized Trial of a Companion Robot for People with Dementia Living in the Community. *J. Am. Med. Dir. Assoc.* **2017**, *18*, 871–878. [CrossRef] [PubMed]
5. Bartneck, C.; Belpaeme, T.; Eyssel, F.; Kanda, T.; Keijsers, M.; Šabanović, S. *Human–Robot Interaction: An Introduction*; Cambridge University Press: Cambridge, UK, 2020. [CrossRef]
6. Mohebbi, A. human–robot Interaction in Rehabilitation and Assistance: A Review. *Curr. Robot. Rep.* **2020**, *1*, 131–144. [CrossRef]
7. Gao, Q.; Liu, J.; Ju, Z. Robust real-time hand detection and localization for space human–robot interaction based on deep learning. *Neurocomputing* **2020**, *390*, 198–206. [CrossRef]
8. Rawal, N.; Stock-Homburg, R. Facial emotion expressions in human–robot interaction: A survey. *Int. J. Soc. Robot.* **2022**, *14*, 1583–1604. [CrossRef]
9. Mehrabian, A. Communication without words. In *Communication Theory*; Routledge: Brighton, UK, 1968.
10. Quiroz, M.; Patiño, R.; Diaz-Amado, J.; Cardinale, Y. Group Emotion Detection Based on Social Robot Perception. *Sensors* **2022**, *22*, 3749. [CrossRef]
11. Daher, A.W.; Rizik, A.; Muselli, M.; Chible, H.; Caviglia, D.D. Porting Rulex Software to the Raspberry Pi for Machine Learning Applications on the Edge. *Sensors* **2021**, *21*, 6526. [CrossRef]
12. Tekler, Z.D.; Low, R.; Gunay, B.; Andersen, R.K.; Blessing, L. A scalable Bluetooth Low Energy approach to identify occupancy patterns and profiles in office spaces. *Build. Environ.* **2020**, *171*, 106681. [CrossRef]
13. Tekler, Z.D.; Low, R.; Yuen, C.; Blessing, L. Plug-Mate: An IoT-based occupancy-driven plug load management system in smart buildings. *Build. Environ.* **2022**, *223*, 109472. [CrossRef]
14. Saeed, U.; Ullah Jan, S.; Lee, Y.D.; Koo, I. Machine Learning-based Real-Time Sensor Drift Fault Detection using Raspberry Pi. In Proceedings of the 2020 International Conference on Electronics, Information, and Communication (ICEIC), Barcelona, Spain, 19–22 January 2020; pp. 1–7. [CrossRef]
15. Babu, R.G.; Karthika, P.; Rajan, V.A. Secure IoT Systems Using Raspberry Pi Machine Learning Artificial Intelligence. In Proceedings of the International Conference on Computer Networks and Inventive Communication Technologies, Coimbatore, India, 23–24 July 2020; pp. 797–805. [CrossRef]
16. Shao, J.; Qian, Y. Three convolutional neural network models for facial expression recognition in the wild. *Neurocomputing* **2019**, *355*, 82–92. [CrossRef]

17. Kanade, T.; Cohn, J.; Tian, Y. Comprehensive database for facial expression analysis. In Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580), Grenoble, France, 28–30 March 2000; pp. 46–53. [CrossRef]
18. Lucey, P.; Cohn, J.; Kanade, T.; Saragih, J.; Ambadar, Z.; Matthews, I. The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition—Workshops, San Francisco, CA, USA, 13–18 June 2010; pp. 94–101. [CrossRef]
19. Tang, Y.; Zhang, X.; Hu, X.; Wang, S.; Wang, H. Facial Expression Recognition Using Frequency Neural Network. *IEEE Trans. Image Process.* **2021**, *30*, 444–457. [CrossRef] [PubMed]
20. Liu, L.; Jiang, R.; Huo, J.; Chen, J. Self-Difference Convolutional Neural Network for Facial Expression Recognition. *Sensors* **2021**, *21*, 2250. [CrossRef] [PubMed]
21. Devaram, R.R.; Beraldo, G.; De Benedictis, R.; Mongiovì, M.; Cesta, A. LEMON: A Lightweight Facial Emotion Recognition System for Assistive Robotics Based on Dilated Residual Convolutional Neural Networks. *Sensors* **2022**, *22*, 3366. [CrossRef] [PubMed]
22. Rathour, N.; Alshamrani, S.S.; Singh, R.; Gehlot, A.; Rashid, M.; Akram, S.V.; AlGhamdi, A.S. IoMT Based Facial Emotion Recognition System Using Deep Convolution Neural Networks. *Electronics* **2021**, *10*, 1289. [CrossRef]
23. Rathour, N.; Khanam, Z.; Gehlot, A.; Singh, R.; Rashid, M.; AlGhamdi, A.S.; Alshamrani, S.S. Real-Time Facial Emotion Recognition Framework for Employees of Organizations Using Raspberry-Pi. *Appl. Sci.* **2021**, *11*, 10540. [CrossRef]
24. Jeong, M.; Ko, B.C. Driver's Facial Expression Recognition in Real-Time for Safe Driving. *Sensors* **2018**, *18*, 4270. [CrossRef]
25. Happy, S.L.; Routray, A. Automatic facial expression recognition using features of salient facial patches. *IEEE Trans. Affect. Comput.* **2015**, *6*, 1–12. [CrossRef]
26. Siam, A.; Soliman, N.; Algarni, A.; Abd El-Samie, F.; Sedik, A. Deploying Machine Learning Techniques for Human Emotion Detection. *Comput. Intell. Neurosci.* **2022**, *2022*, 8032673. [CrossRef]
27. Jiang, P.; Wan, B.; Wang, Q.; Wu, J. Fast and Efficient Facial Expression Recognition Using a Gabor Convolutional Network. *IEEE Signal Process. Lett.* **2020**, *27*, 1954–1958. [CrossRef]
28. Ab Wahab, M.N.; Nazir, A.; Zhen Ren, A.T.; Mohd Noor, M.H.; Akbar, M.F.; Mohamed, A.S.A. Efficientnet-Lite and Hybrid CNN-KNN Implementation for Facial Expression Recognition on Raspberry Pi. *IEEE Access* **2021**, *9*, 134065–134080. [CrossRef]
29. Sajjad, M.; Nasir, M.; Ullah, F.U.M.; Muhammad, K.; Sangaiah, A.K.; Baik, S.W. Raspberry Pi assisted facial expression recognition framework for smart security in law-enforcement services. *Inf. Sci.* **2019**, *479*, 416–431. [CrossRef]
30. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, Kauai, HI, USA, 8–14 December 2001; Volume 1, p. I. [CrossRef]
31. Kartynnik, Y.; Ablavatski, A.; Grishchenko, I.; Grundmann, M. Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs. *arXiv* **2019**. [CrossRef]
32. Ekman, P.; Friesen, W.V. *Facial Action Coding System: A Technique for the Measurement of Facial Movement*; American Psychological Association: Washington, DC, USA, 1978.
33. Ekman, P.; Friesen, W.V. *Manual of the Facial Action Coding System (FACS)*; Consulting Psychologists Press: Palo Alto, CA, USA, 1978.