Sérgio Chevtchenko

Multi-Objective Optimization for Real-Time Hand Posture Recognition

Brasil

2017

Sérgio Chevtchenko

Multi-Objective Optimization for Real-Time Hand Posture Recognition

Undergraduate work presented to the program of Bachelor's of Computer Science of the Department of Statistics and Informatics of Federal Rural University of Pernambuco as partial requirement to obtaining of Bachelor's of Computer Science degree.

Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Bacharelado em Ciência da Computação

Supervisor: Valmir Macario

Brasil

2017

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por Sérgio Chevtchenko às 10 horas do dia 25 de agosto de 2017, no Auditório do CEAGRI-02 – Sala 07, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado **Multi-Objective Optimization for Real-Time Hand Posture Recognition**, orientado por Valmir Macário Filho e aprovado pela seguinte banca examinadora:

Valmir Macário Filho
DEINFO/UFRPE

Filipe Rolim Cordeiro
DEINFO/UFRPE

Péricles Barbosa Cunha de Miranda
DEINFO/UFRPE

This work is dedicated to my grandfather, electrical engineer Sergej Denisovitch Chevtchenko.

Acknowledgements

Special thanks go to my advisor, Valmir Macário (UFRPE) for his patience, motivation and dedication. I would like to show my sincere gratitude to Oleg Krasilnikov (UFPE, *in memoriam*), Paulo Sérgio Brandão de Nascimento (IFPE), Diogo Roberto Raposo de Freitas (UNINASSAU) and several other teachers, both in schools and universities. Last but by no means least, thanks to my family and friends for support and encouragement!

Abstract

Hand gestures are an intuitive way for humans to interact with computers. They are becoming increasingly popular in several application, such as smart houses, games, vehicle infotainment systems, kitchens and surgery rooms. An effective human-computer interaction system should aim for both good recognition accuracy and speed. This work proposes a new approach for static hand gesture recognition. A benchmark database is used for performance evaluation. It contains 2425 images with variations in scale, illumination and rotation. Several common image descriptors, such as Fourier, Zernike moments, pseudo-Zernike moments, Hu moments, complex moments and Gabor features are comprehensively compared by accuracy and speed for the first time. Gesture recognition is made by a multilayer perceptron with a flexible structure. In order to achieve improved accuracy and minimize computational cost, both the feature vector and the neural network are tuned by a multi-objective evolutionary algorithm based on NSGA-II. The proposed method is compared against state-of-the-art methods. Experimental results showed good recognition rate, using a descriptor with low computational cost and reduced size. A real-time gesture recognition system based on the proposed descriptor is constructed and evaluated.

Key-words: Hand posture, MOEA, Neural networks.

Resumo

O uso de gestos de mão é uma maneira intuitiva para humanos interagirem com computadores. Tal forma de comunicação tem aumentado sua popularidade em diversas aplicações, como casas inteligentes, jogos, sistemas de *infotainment* em veículos, cozinhas e salas de cirurgia. Um sistema eficiente de interação humano-computador deve visar obter tanto boa acurácia quanto velocidade de reconhecimento. Este trabalho propõe uma nova abordagem para reconhecimento de gestos estáticos de mão. Uma base de dados de *benchmark* é utilizada para avaliação de desempenho. A base contém 2425 imagens com variações em escala, iluminação e rotação. São implementados diversos descritores de imagens comumente utilizados, como descritores de Fourier, momentos de Zernike e pseudo-Zernike, momentos de Hu, momentos complexos e características de Gabor. Esses descritores são comparados extensivamente, pela primeira vez, em termos de acurácia e tempo computacional. O reconhecimento de gestos é feito por um perceptron multicamada (MLP), contendo uma estrutura parametrizável. Para aumentar a acurácia e minimizar o tempo de reconhecimento, a rede neural e o conjunto de características são otimizados por um algoritmo genético multi-objetivo baseado em NSGA-II. O método proposto é comparado com o estado da arte. Resultados experimentais mostram uma boa taxa de reconhecimento, utilizando um descritor com baixo custo computacional e quantidade reduzida de características. Um sistema em tempo real é construído com base no resultado de otimização.

List of Figures

List of Tables

Contents

1 Introduction

Static hand gestures, also known in the literature as hand postures, can provide an intuitive human-computer interaction (HCI), in addition to being a non-verbal form of communication between humans. A hand gesture recognition system can be used for an effective and natural human-machine interaction (WANG; WANG, 2008). It can also be beneficial for the deaf or hearing impaired, who use mostly gestures for communication (BIRK; MOESLUND; MADSEN, 1997). Other potential applications for this system include:

- Virtual object manipulation.
- Interaction with multimedia and games.
- Smart houses.
- Infotainment systems in vehicles.
- Use in places with critical sanitary conditions, such as surgery rooms and kitchens.

The chart in Figure 1 shows the growth in interest in hand gesture recognition. Despite the increasing effort there are still open challenges, some of which will be discussed further and addressed in this work.

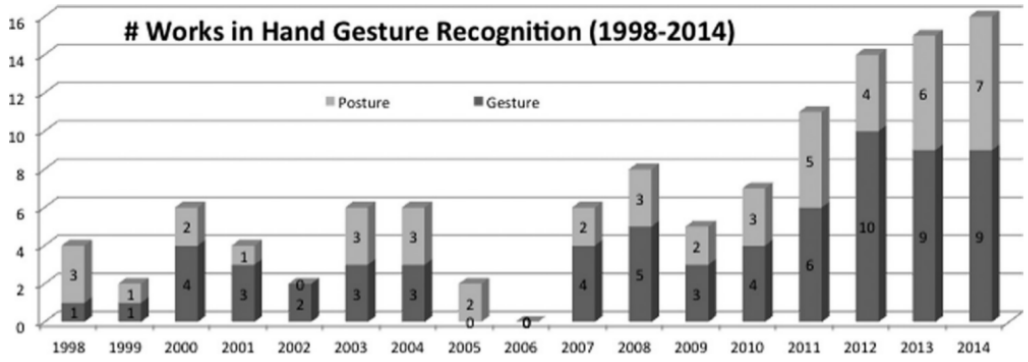


Figure 1 – Chart depicting the growing research efforts in hand gesture and posture recognition (PISHARADY; SAERBECK, 2015).

Overall, a hand posture recognition system relies on acquisition, segmentation, feature extraction and recognition of specific visual features characterizing the posture in the image (RASINES; REMAZEILLES; BENGOA, 2014). Once the hand acquisition and segmentation steps are completed, the problem consists in extracting a good feature set and recognizing the hand posture with an effective classifier. Furthermore, an effective hand

posture recognition method has to be both accurate and real time capable. Although a variety of feature extraction and recognition methods focuses on improving the recognition rate of static hand gestures, few of them, such as Rasines, Remazeilles e Bengoa (2014) and Guo, Liu e Gong (2015), also consider the computational cost involved.

1.1 Objectives

The purpose of this work is to demonstrate a novel combination of features and optimization techniques for fast and accurate hand posture recognition. The proposed methodology has not been used before for hand posture recognition. A multi-objective genetic algorithm (NSGA-II) is designed to improve accuracy, minimize the number of features and simplify connections in an artificial neural network (ANN). Extensive experimentation is conducted with a hand posture database (BARCZAK et al., 2011) and the proposed method is compared with existing ones in terms of recognition accuracy, computational cost and feature vector size. Finally, a real-time gesture recognition system based on the proposed descriptor is implemented and evaluated.

1.2 Contributions

Commonly used image descriptors, namely Fourier, Zernike moments, pseudo-Zernike moments, Hu moments, complex moments and Gabor features, are comprehensively compared by accuracy and speed for the first time. For better performance the algorithms are implemented in C++. A hand posture database (BARCZAK et al., 2011) with 2425 images and 36 classes is used for benchmarking and the results are statistically compared. Thus the results presented here are a good stating point for further work with these descriptors.

Furthermore, a multilayer perceptron is implemented for the recognition step. The neural network is designed to allow feature selection and adjustment in the number of hidden layers. This way it is possible to decrease the number of features and the overall complexity of the classifier, as this makes overfitting of the test data less likely to occur (SOYEL; TEKGUC; DEMIREL, 2011). Reduced complexity is also important for real-time systems as it usually means decreased computational cost.

As previously stated, hand posture recognition is a multi-objective problem: the system should be designed to maximize accuracy and minimize recognition time. This is addressed in this work by wrapping the feature vector and the classifier in a multi-objective genetic algorithm based on NSGA-II. The initialization step of this algorithm is adapted for feature selection and the whole optimization system is implemented on a small computer cluster.

Finally, while this work's main focus is feature extraction and classification, other steps, such as image retrieval and preprocessing are also implemented and evaluated. A real-time system is implemented based on the optimization results, as described further. A paper has been submitted to a journal for review in March, 2017.

2 Background

This section provides basic information about hand posture recognition and multi-objective optimization. Related work on hand posture recognition is also reviewed.

2.1 Hand Posture Recognition

A hand posture recognition system is illustrated in Figure 2 and consists essentially of the following steps: (a) image acquisition; (b) hand segmentation; (c) feature extraction; (d) hand posture classification. Although hand segmentation is as much challenging as feature extraction and classification, it is usually treated as a separate problem, with proposed solutions varying from special gloves and 3D cameras to purely computational (BADI; HUSSEIN; KAREEM, 2014). Since this work is focused on feature extraction and recognition, the database is already segmented, requiring only a binarization step. In this case, this step is straightforward and consists of assigning white color to all pixels with values greater than zero. A more detailed explanation about image processing before feature extraction is presented in Section 4, where a real time gesture recognition system is described.

After extraction from a segmented frame, the image descriptors are used to compose the feature vector. The descriptors have to be carefully selected in order to achieve good recognition rate without causing a significant delay in the human-machine interaction. In the proposed method, an evolutionary algorithm is used to iteratively select the best combination of features so that the two following goals are met: optimal recognition accuracy and minimum computational cost. The resulting feature vector has a reduced complexity and therefore is less likely to cause overfitting.

Similarly to the feature extraction step, gesture classification has to be both fast and accurate. A multilayer perceptron (MLP) network is used to recognize the hand

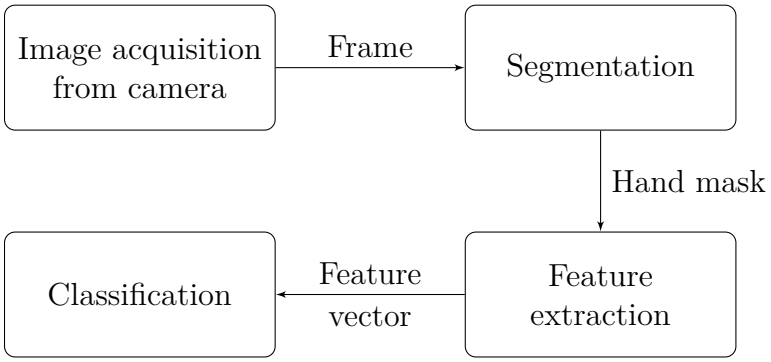


Figure 2 – Block diagram of a standard hand posture recognition system.

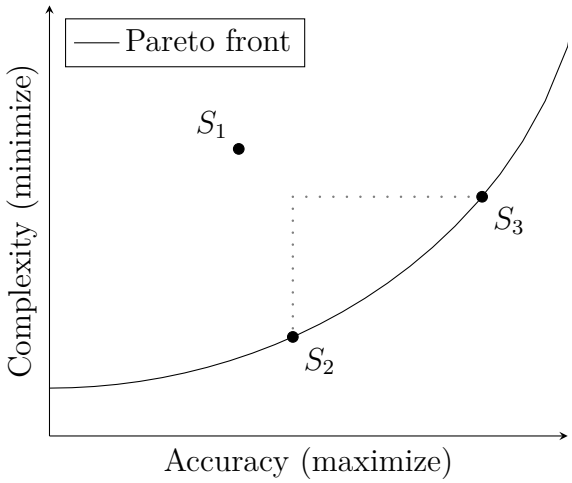


Figure 3 – Multi-objective feature selection problem.

posture, having the feature vector as input. An MLP is a standard method for nonlinear classification and can approximate any continuous function to an arbitrary accuracy (LEUNG et al., 2003). However, the network's performance within a given training period is dependent on its structure. A small network may have limited processing capability, while a large one may be too slow and have redundant connections. Thus, the same evolutionary algorithm is also used to select the optimal layout for the neural network.

2.2 Multi-Objective Optimization

Multi-objective problems occur when an optimal decision is a trade-off between two or more concurrent objectives. Hand posture recognition is inherently a multi-objective problem, where any given solution can be evaluated by two criteria: the complexity of selected features and the corresponding recognition rate. It is desirable to minimize the complexity of the feature vector and to maximize accuracy.

In the feature selection problem addressed in this work, one solution A is said to dominate another solution B if the following criteria are met:

- A has lower feature complexity than B and/or yields better accuracy.
- A is not worse than B, neither in terms of complexity nor of accuracy.

As an example, let us consider three different solutions in Figure 3: S_1, S_2 and S_3. S_1 is considered a dominated solution by both S_2 and S_3, while neither S_2 nor S_3 can be considered better or worse than another without the introduction of a new criteria. The set of all nondominated solutions, like S_2 and S_3, is called a *Pareto front*. A multi-objective optimization is designed to search for an optimal *Pareto front* for a given problem.

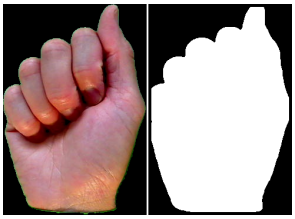


Figure 4 – Original and binary images.

In this work a Multi-Objective Genetic Algorithm (MOGA) is used to search for Pareto-optimal solutions. A Pareto front is a useful tool to evaluate a trade-off between speed and accuracy and support the decision for choosing a final model. Since the Pareto front contains multiple solutions, one of them is manually selected to be used for a real-time recognition system.

2.3 Feature Extraction

All features are extracted from binary and grayscale images containing only the hand palm, as shown in Figure 4.

The following feature descriptors are implemented and evaluated in this work: Hu moments, complex moments, Zernike moments, pseudo-Zernike moments, Fourier descriptors and Gabor features. Combinations of these features are also considered. A detailed description of the implementation of evaluated feature descriptors is given below.

2.3.1 Hu moments

The Hu set of moment invariants (HU, 1962) is one of the oldest and best established image descriptors. They remain roughly constant under scale, rotation and translation. The seven moments used in this work are extracted from a binary image and are defined

as follows:

$$
\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
&\quad + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{2.1}
$$

where each η_{ij} is a scale invariant moment, as defined by (HU, 1962).

2.3.2 Complex moments

Introduced by Abu-Mostafa e Psaltis (1985), the complex moments are a simple and computationally inexpensive form of moment invariants. As defined by Hasan e Abdul-Kareem (2014), a complex moment of order m over an image intensity function $f(x, y)$ is

$$C_m = \int \int (x + iy)^m f(x, y)\, dx\, dy \tag{2.2}$$

where i is an imaginary unit and x, y are image coordinates. Each complex moment has two parts: real and imaginary. Since the neural network is designed for real values, we take the modulus of a complex moment, $|C_m|$, as input to the neural network. As in the work of Hasan e Abdul-Kareem (2014), the feature vector is obtained from the first 10 complex moments.

2.3.3 Zernike and pseudo-Zernike moments

The Zernike polynomials were first proposed in 1934 by Frits Zernike. The corresponding moments are known to be somewhat invariant to rotation, and can be modified to also be stable under scale and translation (TEH; CHIN, 1988). The pseudo-Zernike formulation (PZMs) proposed by Bhatia e Wolf (1954) has been shown to have better feature representation and to be more robust to image noise than conventional Zernike moments (TEH; CHIN, 1988).

The Zernike moments (ZM) implementation is based on Hse e Newton (2004). Complex ZM of a 2D image of order n and repetition m, over an intensity image $f(x, y)$, is

$$Z_{n,m} = \frac{n+1}{\pi} \sum_x \sum_y f(x,y) V_{nm}(x,y)^*$$
$$n - |m| \text{ even}, \quad |m| \leq n \tag{2.3}$$

where $x^2 + y^2 \leq 1$, so the original image coordinates are scaled for a unit disk $x^2 + y^2 = 1$, and * denotes the complex conjugate. The Zernike polynomial $V_{nm}(x, y)$ is defined as

$$V_{nm}(x,y) = V_{nm}(r,\theta) = R_{nm}(r) e^{jm\theta}$$
$$R_{nm}(r) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} r^{n-2s} \tag{2.4}$$

where j is the imaginary unit and $0 \leq r \leq 1$. The normalized image coordinates (x, y) transformation to the domain of the unit circle (r, θ) is given by

$$r = \sqrt{x^2 + y^2}, \quad \theta = tan^{-1}\left(\frac{y}{x}\right) \tag{2.5}$$

Since $Z_{n,-m} = Z_{n,m}$ and therefore $|Z_{n,-m}| = |Z_{n,m}|$, only $|Z_{n,m}|$ is used for the feature vector. Also, $|Z_{0,0}|$ and $|Z_{1,1}|$ are the same for all normalized images and are not used.

PZMs are calculated the same way as Zernike moments, based on a set of orthogonal polynomials, defined over the polar space inside a unit circle. The difference is in the definition of the radial polynomial $R_{nm}(r)$:

$$R_{nm}(r) = \sum_{s=0}^{n-|m|} (-1)^s \frac{(2n+1-s)!}{s!(n+|m|-s)!(n+|m|+1-s)!} r^{n-s} \tag{2.6}$$

where m can now assume positive and negative values, subject only to the restriction $|m| \leq n$. This implies that a set of pseudo-Zernike moments of order n is of size $(n+1)^2$, whereas a set of Zernike moments of the same order contains only $(n+1)(n+2)/2$ polynomials, due to the additional constraint of $n - |m|$ being even (CHONG; RAVEENDRAN; MUKUNDAN, 2003).

A disadvantage of pseudo-Zernike moments is in the computational cost involved in the calculation of radial polynomials. A faster, p-recursive method for calculation of $R_{nm}(r)$, proposed by Chong, Raveendran e Mukundan (2003), is used in this work.

Standard PZMs, applied on an $N \times N$ image, have a computational complexity of $O(N^2 n^3)$ and the p-recursive method used in this work has a complexity of $O(N^2 n^2)$ (CHONG; RAVEENDRAN; MUKUNDAN, 2003). In order to further increase computational efficiency, PZMs are calculated with fixed repetition ($m = 0$). Therefore, in this

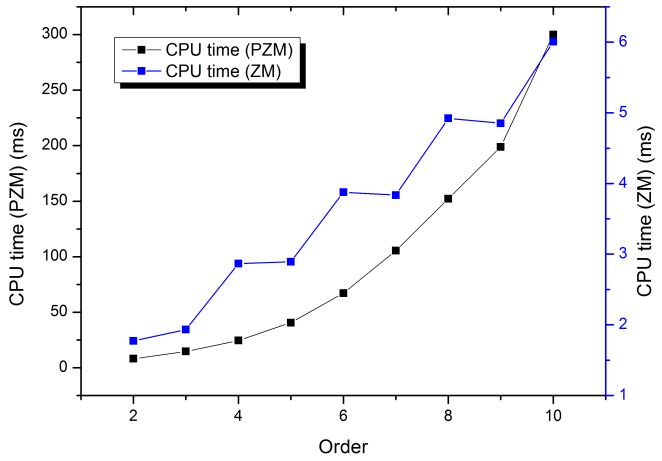


Figure 5 – Computation time for Zernike and pseudo-Zernike moments up to order 10.

implementation the complex term $e^{jm\theta}$ is constant and equal to one. Figure 5 shows the average CPU time for calculation of Zernike and pseudo-Zernike moments up to order 10, from a set of 2425 grayscale images with 64×64 pixels. Zernike moments are shown to be significantly faster than pseudo-Zernike, thus being more widely used.

2.3.4 Fourier descriptors

These features are based on the boundary (contour) coordinates, $x(t)$ and $y(t)$, extracted from a segmented shape. Two different representations for the contour function are used: complex coordinates and centroid distance function. The following implementation is based on Zhang e Lu (2001).

In the first representation each coordinate of the contour is converted to a complex function

$$z(t) = [x(t) - x_c] + i[y(t) - y_c] \tag{2.7}$$

where (x_c, y_c) are the centroid coordinates of the contour. This shift is used in order to make *z(t)* invariant to translation. Fourier transform is then applied to the magnitude of *z(t)*.

The second method is based on the centroid distance function, defined below. As with *z(t)*, the centroid distance function is also invariant to translation.

$$r(t) = \sqrt{[x(t) - x_c]^2 + [y(t) - y_c]^2} \tag{2.8}$$

A discrete Fourier transform (DFT) is applied to the contour functions described above. This transformation is defined as

$$u_n = \frac{1}{N} \sum_{t=0}^{N-1} s(t) \left(\cos\left(\frac{2\pi nt}{N}\right) - j \sin\left(\frac{2\pi nt}{N}\right) \right) \tag{2.9}$$

where N is the number of contour points and *s(t)* is the contour function, namely *r(t)* or *z(t)*, as defined in Equations 2.7 and 2.8. Usually u_n is called the Fourier descriptor of the shape.

When *s(t)* is a complex function *z(t)*, the first component u_0, also called a *dc* component, is not used, since it is dependent only on the position of the shape. The feature vector f is then composed of $N-2$ descriptors (ZHANG; LU, 2001).

$$f \stackrel{def}{=} \left[\frac{|u_2|}{|u_1|}, \frac{|u_3|}{|u_1|}, ..., \frac{|u_{N-1}|}{|u_1|}\right] \tag{2.10}$$

For a Fourier descriptor based on the centroid distance *r(t)*, Equation 2.8 uses only real numbers, so that the Fourier transform yields $N/2$ different values. Therefore, the feature vector size is limited to $N/2$, as defined below (ZHANG; LU, 2001).

$$f \stackrel{def}{=} \left[\frac{|u_1|}{|u_0|}, \frac{|u_2|}{|u_0|}, ..., \frac{|u_{N/2}|}{|u_0|}\right] \tag{2.11}$$

2.3.5 Gabor features

The response of this filter is considered to be a representation of simple visual cells in mammals. It is widely used to recognize texture features by convolving an image with the filter kernel (JAIN; FARROKHNIA, 1991). A two-dimensional Gabor filter is given by

$$\begin{aligned} G(x,y) &= \exp\left(-\frac{1}{2}\left[\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right]\right)\cos\left(\frac{2\pi x_\theta}{\lambda} + \psi\right) \\ x_\theta &= x\cos(\theta) + y\sin(\theta) \\ y_\theta &= -x\sin(\theta) + y\cos(\theta) \\ \sigma_x &= \sigma, \quad \sigma_y = \frac{\sigma}{\gamma} \end{aligned} \tag{2.12}$$

with parameters explained below:

- standard deviation of the gaussian envelope (σ);
- orientation of the normal to the parallel stripes of a Gabor function (θ);
- wavelength of the sinusoidal factor (λ);
- spatial aspect ratio (γ);
- phase offset (ψ).

This filter performs a low pass filtering along the orientation of θ and a band pass filtering orthogonal to its orientation θ. Therefore, by choosing the parameters above, it is possible to enhance visual properties of an image, such as spatial frequency and orientation, as illustrated in Figure 6.

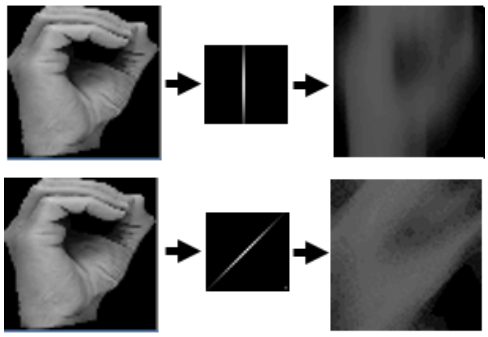


Figure 6 – A Gabor filter with $\theta = 0°$ and $\theta = 45°$.

A feature vector is constructed as follows: first a Gabor filter is applied to a grayscale image containing only the hand palm; the resulting image is then rescaled to 8×8 pixels and the feature vector is composed of the corresponding array of 64 integers. The resulting 64 integers are the feature vectors of the Gabor filter.

Remark: the descriptors defined previously can be used on other arbitrary forms, thus increasing the range of potential applications for the proposed scheme.

2.4 Multiclass Classification

Once features are extracted from the image, a classifier is used to evaluate the hand posture. The classifier receives a numerical string called a feature vector and outputs the class corresponding to that input. In this case, a class represents one of the possible hand postures. Several classifiers can be considered for this task, such as:

- k-nearest neighbors (k-NN) – a popular algorithm to decide the class of a sample based on its feature vector's distance to other training samples. The k least distant samples are said to be the neighbors, and a majority vote determines the class label of the sample.

- Decision trees – a tree structure in which leaves correspond to classes in the problem and nonterminal nodes are used to make decisions of which path to take. These decisions are based on the attribute values of a problem sample. When dealing with continuous numerical data, decision trees are known as regression trees.

- Random Forest – an ensemble of multiple decision trees, each constructed by resampling the sample data with replacement, a technique also known as bootstrapping (LIAW; WIENER et al., 2002). Feature bagging is also be performed by using a random subset of the feature set. Classification is made by a majority vote, which aids in reducing overfitting of the training set. The number of trees as well as the methods to aggregate the results can be adjusted for a specific task.

- Neural networks – this class of algorithms uses biological neural networks as inspiration. Multilayer perceptrons (MLP) are a common example of neural network. Training an MLP is performed with a learning algorithm called backpropagation, whose objective is to minimize the difference between the network's output compared to the expected output. Rumelhart et al. (1988) first described neural networks using backpropagation for training. MLPs will be more detailed in Subsection 4.3.2.

- Multi-class Support Vector Machines (SVM) – a learning algorithm originally meant for binary classification that separates two classes with a plane (BOSER; GUYON; VAPNIK, 1992). This method aims to obtain a maximum decision boundary, that is, the margin between training samples of different classes is the largest possible. To apply this algorithm to a multiclass problem, it is possible to perform pairwise classification or, alternatively, one label can be selected as the first class and all the remaining label can be combined as a single class, turning a multiclass classification problem into a binary classification problem.

2.5 Multi-Objective Optimization

As seen in previous sections, both feature descriptors and classifiers usually have several parameters that need to be adjusted for a specific task. It is common to employ optimization techniques to find a good set of hyperparameter for a classifier and feature extractor.

Since hand posture recognition has at least two competing goals, namely speed and accuracy, a multi-objective search is preferable. If another or more criteria for the system are defined, such as robustness to noise, a many-objective optimization technique may be used.

Regardless of the specific algorithm for optimization, this task will consist in the following steps.

- Define objective functions – the optimization goals are made into measurable functions.

- Define encoding method for the solution – each possible solution to the problem is going to be represented in a certain code.

- Separate the dataset into test and train subsets – in order to avoid overfitting, the optimization algorithm cannot interact with train data. It is usual to separate the training data further using k-folds, this way the optimizer is evaluated with an average performance for the k folds.

- Run optimization – the optimizer performs search in the space defined by hyperparameters, looking to approximate the Pareto front of the problem.
- Test solutions – the Pareto optimal solutions returned by the optimizer are evaluated on the Test subset, previously 'unseen' by the optimizer.

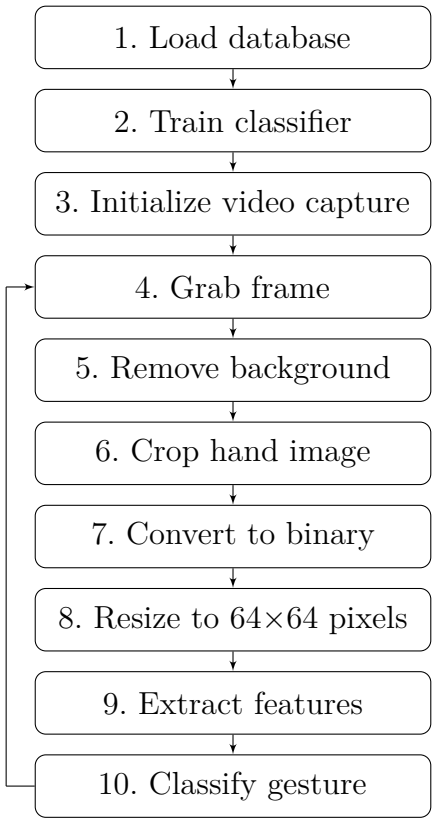
When a single solution has to be selected from the Pareto set of solutions, a decision can be made according to an experts subjective preferences or by an algorithm. For example, a decision maker can choose a solution that maximizes only the accuracy, because speed is acceptable for all the Pareto optimal solutions. A decision maker can also rank the various optimization by preference, thus converting a multi-objective problem into a single-objective. On other hand, an election method, such as Borda count, can be applied to algorithmically select a single solution.

2.6 Real-Time Gesture Recognition

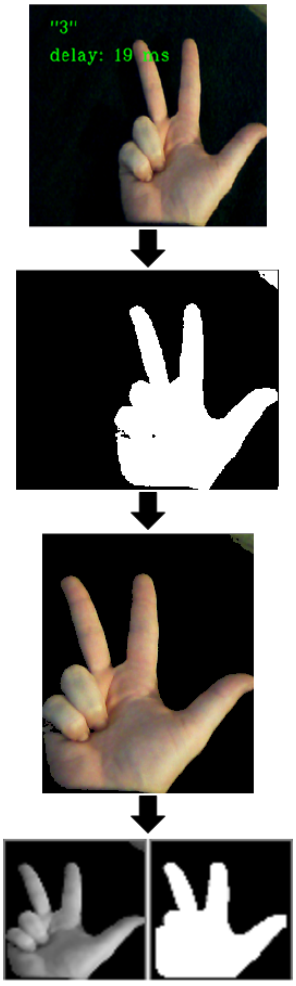
A real-time static gesture recognition system is implemented. As described in the flowchart in Figure 7a, the system consists of the following sequence of steps:

- Load database – a list of features is extracted from the dataset.
- Train classifier – the classifier is created based on the selected chromosome and is trained on the extracted features.
- Initialize video capture – the OpenCV library is used to initialize video capture from a webcam. Real-time recognition starts here.
- Grab frame – a single frame from the webcam is obtained and converted to grayscale.
- Remove background – the background and the wrist are removed by simple threshold.
- Crop hand image – only the hand is left for further processing.
- Convert to binary – the hand image is converted to black and white. The original grayscale image is also preserved.
- Resize to 64×64 pixels – both binary and grayscale images are resized to 64×64 pixels.
- Extract features – features are extracted from grayscale and the binary image.
- Classify gesture – the feature vector is presented to the previously trained classifier and the recognized gesture is printed on-screen. This process is repeated for each frame.

An illustration of steps 4 to 10 of this flowchart is found in Figure 7b, where the gesture "3" is being processed and recognized.



(a) Flowchart of the system.



(b) The main steps involved in gesture preprocessing.

Figure 7 – Real-time gesture recognition system.

3 Related Works

Numerous methods have been recently developed to perform a successful hand posture classification. Chaudhary et al. (2013) compare different applications of hand gesture recognition using computer vision. The main focus of this survey are applications of Artificial Neural Networks, Fuzzy Logic and Genetic Algorithms. While evolutionary algorithms have been successfully applied to hand posture recognition, they have not been used for feature selection and gesture classifier optimization. Several hand gesture acquisition and recognition techniques are surveyed by Rautaray e Agrawal (2015). Robustness and real time capability are identified as important directions for improvement. These are also considered as main objectives of this work.

Hasan e Abdul-Kareem (2014) and Badi, Hussein e Kareem (2014) present a hand gesture recognition system. A total of 6 different gestures are classified with an 86.38% recognition rate, using a multilayer perceptron and complex moments. In this work, complex moments are compared against other common image descriptors with respect to accuracy and speed.

Aowal et al. (2014) have extensively tested discriminative Zernike moments (DZMs) and compared them with standard Zernike moments, principal component analysis (PCA) and Fourier descriptors (FDs). Ng e Ranganath (2002) develop a real-time system that uses Fourier descriptors to represent hand blobs. An RBF network is used for hand pose classification. Hu moments are compared with Zernike moments for general object recognition (SABHARA; LEE; LIM, 2013) as well as for static hand gestures (OTINIANO-RODRÍGUEZ; CÁMARA-CHÁVEZ; MENOTTI, 2012). In general, Zernike moments have been found to be more accurate than Hu moments (OTINIANO-RODRÍGUEZ; CÁMARA-CHÁVEZ; MENOTTI, 2012; SABHARA; LEE; LIM, 2013). Guo, Liu e Gong (2015) propose an improved version of Zernike moments, measuring accuracy as well as computational cost on a common static hand gesture database. The same databse is used in this work. Avraam (2014) proposes a combination of graph-based characteristics and appearance-based descriptors such as detected edges for modeling static gestures. This method is tested on 10 numerical gestures from the Massey database (BARCZAK et al., 2011). An average recognition rate of 83% with a standard deviation of 10% is achieved. Local Binary Patterns (LBP), Speeded Up Robust Features (SURF), and Super Pixels (SP) are compared by Kumar e Manjunatha (2017) on a proprietary dataset. SURF features are found to yield better recognition rate, although the feature extraction time is not considered. A sequential forward feature selection technique (SFS) is implemented by Rasines, Remazeilles e Bengoa (2014) to select the most promising feature set among a large variety of features, such as Gabor features, histogram of oriented gradient (HOG),

Shannon entropy, Hu moments and others. The results are compared to those of Avraam (2014), improving the average recognition rate on 10 gestures of the same database to 98% with a standard deviation of 7%. The Massey database is also used in this work, with results presented for the entire set of gestures, including 26 letters and 10 digits.

Although the feature set is very important for any classification task, the computational cost involved in this task is just as critical for real-time systems. For example, despite being successfully used in diverse image recognition problems, Zernike moments have the disadvantage of a higher CPU time compared to other common image descriptors. In this work a significantly faster descriptor, based on Gabor filter, is implemented.

It is also necessary to consider how a neural network's topology influences recognition accuracy and speed. A larger neural network may be better at recognizing a certain pattern, but it will have greater training and classification costs. Genetic algorithms (GAs) have been successfully used to improve performance in neural networks. Leung et al. (2003) propose a neural network with switches in its links, using a genetic algorithm to tune its structure and parameters. The number of hidden nodes is chosen manually, starting from a small number and increasing it until a good recognition rate is reached. The result of genetic tuning is a smaller, partially connected network, reducing computation, hardware and memory costs. A similar approach is developed by Tsai, Chou e Liu (2006), using a hybrid Taguchi-genetic algorithm (HTGA) and improving the GA's capability to find the global optimum. The genetic algorithm presented in this work simultaneously optimizes the number of hidden neurons and the configuration of inputs in the neural network.

Considering the importance of choosing the best feature set to represent hand postures and the computational cost of the best classifier to perform this task in real time, this work compares several commonly used features, namely Zernike and pseudo-Zernike moments, Fourier descriptors, complex moments, Hu features and Gabor features. These image descriptors are compared based on recognition accuracy and computational cost. Given that feature selection is a multi-objective problem, this work uses a well known multi-objective evolutionary algorithm (NSGA-II) in order to select the best combination of features and optimize the performance of the neural network.

Although artificial neural networks and genetic algorithms have been used separately in this problem, the presented work is unique as it combines both. Also, as far as the authors know, the feature descriptors tested in this work are compared for the first time with respect to accuracy and recognition speed, providing a good starting point for future works.

4 Experimental Evaluation

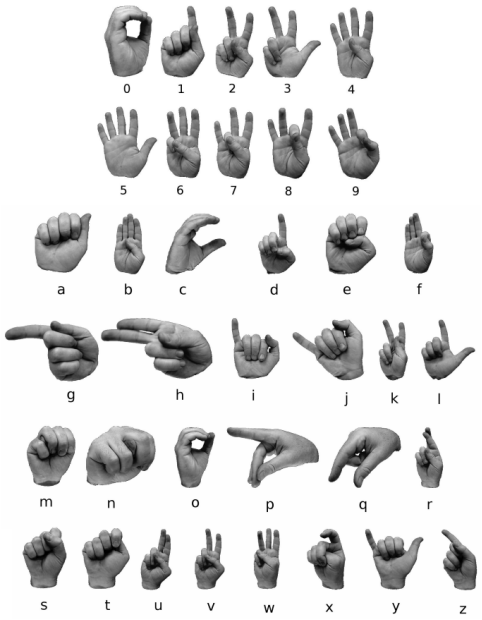
4.1 Hardware Setup and Validation Technique

The method above is implemented and tested on several computers with i5 processors and 4 GB RAM. The optimization is performed on a cluster formed by 28 such computers. The MPJ library is used for message parsing (CARPENTER et al., 2000). For better performance, all algorithms are implemented in C++ with the OpenCV library (BRADSKI; KAEHLER; PISAREVSKY, 2005).

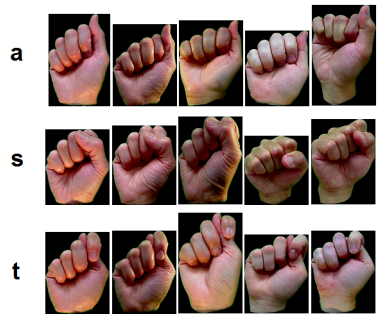
There are several strategies for experimental design involving multi-objective feature selection and presentation of the results. In this work we follow the strategy used by Xue, Zhang e Browne (2013), Hamdani et al. (2007) and Ishibuchi, Nojima e Doi (2006). In order to increase statistical robustness, the results are averaged across a total of 50 experiments. The dataset is always divided randomly in training (80%) and testing (20%).

4.2 Benchmark Dataset

The Massey University Gesture Dataset (BARCZAK et al., 2011) is a well known American Sign Language (ASL) database. It was published in 2011 and contains 2425 color images of 36 classes (26 letters and 10 numeric gestures). With the exception of Guo, Liu e Gong (2015) and Kumar, Nandi e Kala (2014), other related works provide results for only a part of this dataset (AOWAL et al., 2014; AVRAAM, 2014; RASINES; REMAZEILLES; BENGOA, 2014). Although experiments are carried with parts of the Massey dataset as well, the results are presented for the entire database.



(a) Samples from the database.



(b) Three similar letters: “a”, “s” and “t”, showing variations in scale, tone, illumination and slight rotation within the same class.

Figure 8 – The Massey hand posture database (BARCZAK et al., 2011).

Figure 8a shows samples of different classes. Note that some gestures are rather difficult to distinguish. For example “a” and “e”, “d” and “1”, “m” and “n” or “i” and “j”. Furthermore, the database contains variations in scale, illumination and rotation, as illustrated in Figure 8b with three similar gestures: “a”, “s” and “t”. In order to minimize the influence of scale and increase computational speed, the images are normalized to 64×64 pixels prior to the feature extraction step.

4.3 Implementation

This section describes details in implementation of methods found in Section 2.

4.3.1 Gabor features

Since the assignment of values for Gabor parameters is application dependent, experimental adjustments are made considering the database used in this work. Gabor features with different parameters are extracted from subsets of the Massey database. A Bayesian classifier is used for comparison, due to increased training and testing speed. Only one parameter is varied at a time, while the rest is kept constant. The following ranges of values are considered:

- w: set to 64 from $\{16, 32, 64, 72, 128\}$.
- σ: set to 0.02 or 0.2 from 0.01–1.

- θ: set to 0° from {0, 45, 90, 135, 180}.
- λ: set to 1 from 0.5–2.
- γ: set to 0.02 from 0.01–0.04.
- ψ: set to 0 from 0–1.

Only standard deviation σ is found to influence the recognition rate, with the best values between 0.02 and 0.2, depending on the subset under consideration.

4.3.2 Neural network

The artificial neural network used is this work is called a multilayer perceptron (MLP). It is constructed by multiple layers of fully connected neurons. An artificial neuron is a simple unit of computation, having multiple inputs $i_1, i_2, ..., i_n$ and a single output o. Each input has an associated weight w_n. An output can be expressed in terms of inputs as $o = f(\sum i_n w_n)$. The function f is called an activation function. The sigmoid function $f(x) = 1/(1 + e^{-x})$ is commonly used as activation and is also used in this work.

A multilayer perceptron is usually trained by a supervised algorithm called backpropagation. Assuming there is a dataset with input vectors and desired outputs, the error of the network can be easily computed by subtracting the obtained output from the desired one. Backpropagation then calculates the gradient of the error function in terms of the weights of the network. The weights are then adjusted iteratively in order to minimize this error.

A configurable three-layer perceptron network is used to classify the feature vector. A network with l inputs, n neurons in the hidden layer and m outputs is shown in Figure 9. This network has two parameters that can be optimized for a specific problem:

- an array of l switches, connecting the feature vector f_k with the input layer i_k. The arrangement of the switches defines the number of active neurons in the input layer, i.e., the combination of features actually used for training and classification.
- the number of neurons in the hidden layer (n). This parameter is used to configure the number of connections within the network.

Since each output neuron corresponds to one class, the size of the output layer is determined by the number of classes (m).

The following parameters of the artificial neural network are experimentally adjusted:

- maximum number of iterations: set to 300, from a range between 100 and 400.

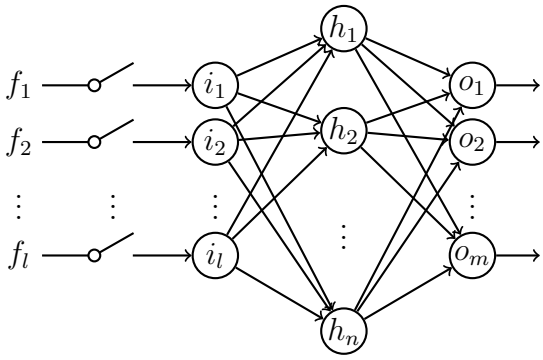


Figure 9 – An MLP with variable inputs and hidden layer size.

- admissible error: set to 10^{-5}, from a range between 10^{-6} and 10^{-4}.
- gradient decent step size: set to 0.002, from a rage between 0.001 and 0.1.
- back propagation momentum value: set to 0.01, from a range between 0.05 and 0.1.

4.3.3 Optimization algorithm

A multi-objective evolutionary algorithm known as NSGA-II (Nondominated Sorting Genetic Algorithm) is adapted for this application, which aims to achieve an increased gesture recognition accuracy while reducing the number of features used to that end. NSGA-II is a popular algorithm introduced by Deb et al. (2002). It has recently been used for a variety of optimization problems but has been proposed for the first time for hand posture recognition in this work.

As in a standard binary GA, there is a population of solutions and each solution is encoded by a chromosome. As illustrated in Figure 10, each chromosome determines the number n of neurons in the hidden layer (9 bits) and the number of available features (l bits), which correspond to the array of switches in Figure 9.

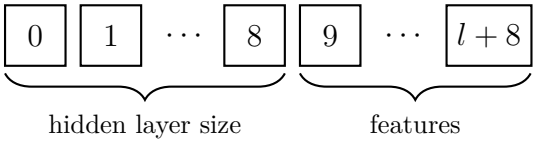


Figure 10 – Binary chromosome structure associated to an ANN.

The main steps of NSGA-II are described in Algorithm 1. A detailed explanation about *nondomination* and *crowding distance* selection criteria can be found in Deb et al. (2002).

In order to avoid bias during optimization, the feature database is divided in Training (80%) and Testing (20%) subsets, as described in Algorithm 1. During the optimization loop, each chromosome represents one possible solution. The accuracy of

this solution is evaluated by 5-fold cross-validation on the Training set. Note that cross-validation is performed in the inner loop of the optimization algorithm to evaluate the performance of a single chromosome. After the optimization loop, the final Pareto front is tested on the Testing subset. This is called a wrapper approach to feature selection and is discussed in more details by Kohavi e John (1997).

Algorithm 1 Multi-objective optimization method based on NSGA-II.

```
Extract features from a segmented dataset
Randomly divide dataset into Training and Testing subsets
Initialize genetic population P_0 of size N
t = 0
while t < Maximum iterations do
    Population Q_t of size N is obtained from P_t through crossover and mutation
    R_t = P_t ∪ Q_t
    for each chromosome in R_t do
        Generate MLP with encoded inputs and hidden layer size
        Evaluate on Training subset, using cross validation
    end for
    New population P_{t+1} of size N is selected from R_t based on nondominance and
    crowding distance
    t = t + 1
end while
Evaluate the final Pareto front of P_t on Testing subset
Return the solutions in the Pareto front, along with training and test classification
accuracies
```

In NSGA-II the representation of each feature is the same as used in other binary feature selections, where l features are encoded in a binary string. A bit has value "1" to indicate that the feature is selected, and "0" otherwise.

The most common way to initialize a binary chromosome is by simply flipping a coin for each bit (CARR, 2014). In the case of feature selection each bit represents a specific feature, and so by flipping a coin every feature will have a 50% chance of being present in a chromosome. This initialization technique has an undesirable result of creating a population of chromosomes with, on average, half of the available features. In other words, solutions with very few features, like "000010000", or with most of the available features, like "111101111", will not be present in the initial population, thus restricting the searching capability of the algorithm. In order to create a population with a uniform distribution with respect to the number of selected features, we perform a uniform random initialization, described in Algorithm 2. A comparison of both initialization methods can be seen in Figure 11 and 12. Each method of initialization is applied to a population of 200 chromosomes, encoding 169 features.

As commonly found in the literature for similar problems involving evolutionary algorithms, a bit-flip mutation is used with a rate of $1/n$, where n is the number of

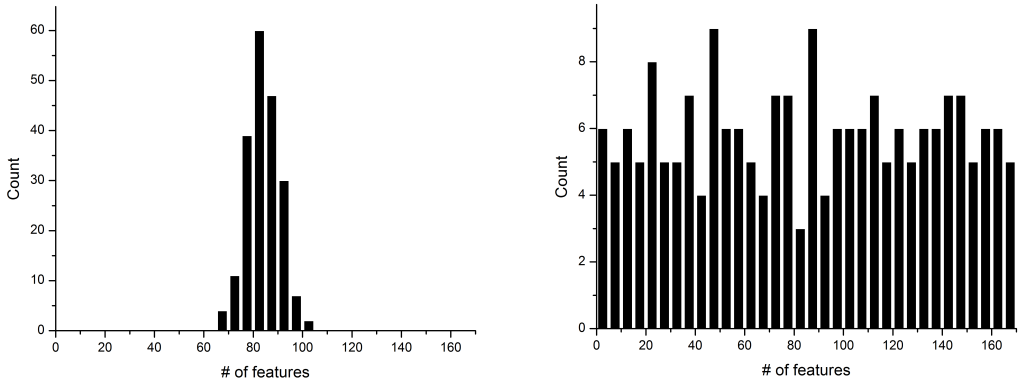


Figure 11 – A frequency count of number of features for a population of 200 chromosomes initialized by flipping a coin for each bit (left) and by uniform random initialization presented in Algorithm 2 (right).

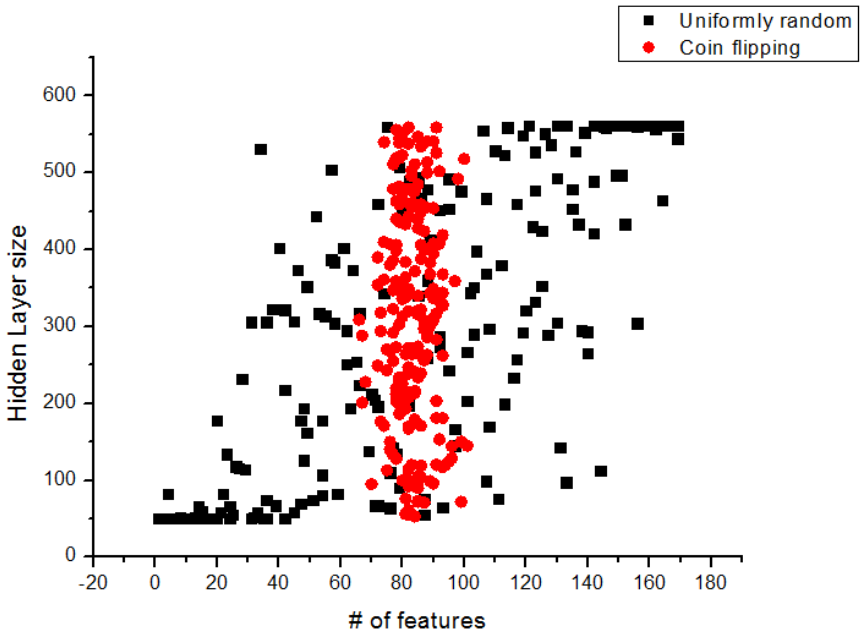


Figure 12 – A distribution of number of features and neurons in the hidden layer encoded by a population of 200 chromosomes.

Algorithm 2 Uniform random initialization of a population of binary chromosomes.

Let n be the number of bits in a chromosome
Let m be the minimum number of features per chromosome
for each chromosome C_i **do**
    C_i = binary string of n "0" /* this creates a clean cromosome with no selected features */
    $r = max(m, i)$
    Create a *list* with r random numbers between 1 and n
    for each number in the *list* **do**
        Set the corresponding bit of C_i to "1" /* as a result, only r features are used */
    end for
end for
Return population

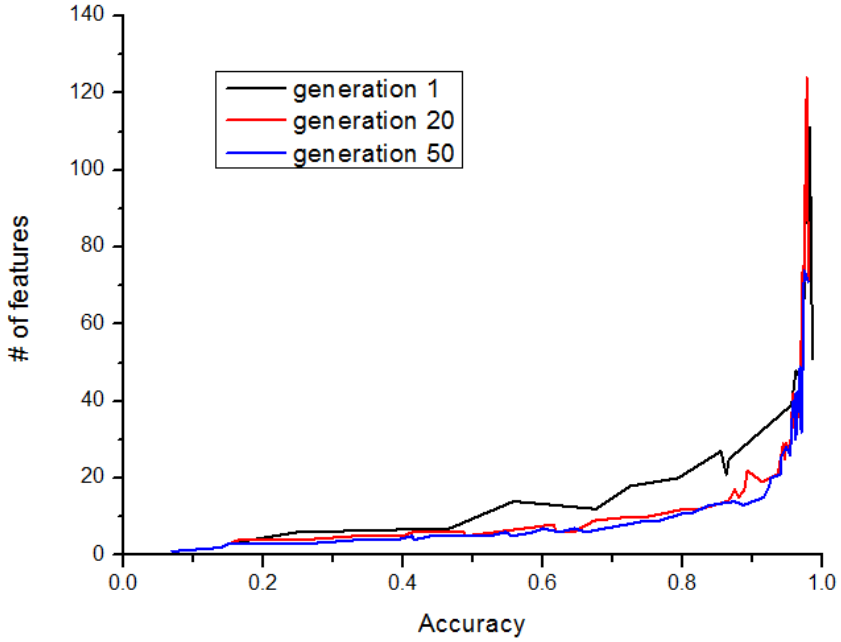


Figure 13 – A Pareto front achieved by NSGA-II after 1, 20 and 50 generations.

bits in the chromosome and the crossover probability is set to 90%. Population size is experimentally set to 50 (from 25, 50 and 100) and the maximum number of iterations to 20 (from 1 to 50). As can be seen in Figure 13, no significant advantage is found in running the optimization further than 20 generations, as the Pareto front does not visibly improve.

4.3.4 Feature vectors

The feature extraction methods defined in Section 2 are implemented as stated below. The combinations of different feature descriptors are presented here for the first time in the context of hand posture recognition.

- **HU** – Implemented as in Otiniano-Rodríguez, Cámara-Chávez e Menotti (2012). An unmodified OpenCV implementation of Hu moments is used, resulting in a vector of 7 features.
- **FD** – Implemented as in Ng e Ranganath (2002) and Zhang e Lu (2001). Fourier descriptor using centroid distance function, extracted from a contour with 120 points. The feature vector is composed of the first 56 items in the vector f, as defined in Equation 2.11.
- **FD2** – Implemented as in Ng e Ranganath (2002) and Zhang e Lu (2001). Absolute value of a Fourier descriptor using complex coordinates, extracted from a contour with 120 points. The feature vector is composed of 56 first items in the vector f, as defined in Equation 2.10.
- **CM** – 10 complex moments, implemented as in (HASAN; ABDUL-KAREEM, 2014) and Badi, Hussein e Kareem (2014).
- **PZM** – Implemented as in Chong, Raveendran e Mukundan (2003). Absolute value of complex pseudo-Zernike moments up to order 10, ignoring the first two moments, resulting in a vector of 64 features.
- **ZM** – Implemented as in Otiniano-Rodríguez, Cámara-Chávez e Menotti (2012). Zernike moments up to order 10, ignoring the first two moments, resulting in a vector of 34 features.
- **ZM_25** – Zernike moments up to order 25, ignoring the first two moments. Implemented as in Guo, Liu e Gong (2015).
- **GB**$_{\sigma=0.02}$ and **GB**$_{\sigma=0.2}$ – 64 features based on a Gabor filter with the following parameters: $w = 64$, $\theta = 0$, $\lambda = 1$, $\gamma = 0.02$, $\psi = 0$. Two different values of standard deviation σ are used: 0.02 and 0.2.
- **GB2** – Concatenation of GB$_{\sigma=0.02}$ (64 features) and GB$_{\sigma=0.2}$ (64 features), resulting in a vector with 128 features.
- **GB2_HU** – Concatenation of GB2 (128 features) and HU (7 features), resulting in a vector with 135 features.
- **GB2_HU_FD** – Concatenation of GB2 (128 features), HU (7 features) and FD, resulting in a vector with 191 features.
- **GB2_ZM** – Concatenation of GB2 (128 features) and ZM (34 features), resulting in a vector with 162 features.
- **GB2_HU_ZM** – Concatenation of GB2 (128 features), HU (7 features) and ZM (34 features), resulting in a vector with 169 features.

- **GB2_HU_ZM PCA_95%** – A common feature reduction technique called Principal Components Analysis (PCA) is applied to **GB2_HU_ZM** feature vector. In order to avoid bias, the PC attributes are obtained from the Testing dataset, retaining enough to account for 95% of the variance. The resulting vector contains 48 features.

- **GB2_HU_ZM PCA_100%** – This is just a linear combination of **GB2_HU_ZM** feature vector, aimed to increase variance and with the same number of features (169).

5 Results and Analysis

The experiments are carried in the following sequence: evaluation of feature descriptors and optimization. They are described in the following subsections.

5.1 Evaluation of feature descriptors

In order do assess the discrimination capability of the feature vectors described above, we test each with an MLP classifier. Hidden layers with 50, 100, 200, 300, 400 and 500 neurons are considered for each experiment. The results are analyzed with respect to recognition accuracy and computational cost.

Table 1 shows the classification accuracies and CPU time for different descriptors. The hidden layer size indicates the best network configuration for a given descriptor. The CPU time during feature extraction is averaged across 700 images.

Succeeding, the accuracies are evaluated as described in Section 2.5: 20% random holdout, repeated 50 times for each hidden layer size (50, 100, 200, 300, 400 and 500). The averaged results are presented with standard deviation (σ).

As can be seen from Table 1, Gabor features achieve better recognition rates than Zernike features. Although the feature vector based on Gabor filter has higher dimension, its computational cost is only a fraction of Zernike or pseudo-Zernike moments. The

Table 1 – Comparison of performance for different feature descriptors.

Feature vector	# of features	Hidden layer	CPU time (ms)	Accuracy (%)
HU	7	300	0.70	37.02 ($\sigma = 5.26$)
FD2	56	50	0.90	40.39 ($\sigma = 1.45$)
CM	10	300	2.10	48.09 ($\sigma = 3.43$)
FD	56	300	0.9	58.69 ($\sigma = 3.05$)
PZM	64	500	109.95	87.77 ($\sigma = 2.96$)
ZM	34	400	36.30	91.08 ($\sigma = 1.95$)
ZM_HU	41	500	37.30	91.41 ($\sigma = 1.79$)
ZM_25	180	400	252.00	92.08 ($\sigma = 1.37$)
GB$_{\sigma=0.2}$	64	400	1.88	95.38 ($\sigma = 1.27$)
GB2_HU_ZM PCA_95%	48	300	43.34	95.86 ($\sigma = 1.20$)
GB2_HU_FD	191	200	3.81	95.88 ($\sigma = 1.00$)
GB2_HU	135	400	2.82	97.09 ($\sigma = 0.80$)
GB2	128	300	2.37	97.15 ($\sigma = 0.84$)
GB$_{\sigma=0.02}$	64	400	1.88	97.16 ($\sigma = 0.96$)
GB2_HU_ZM PCA_100%	169	400	43.34	97.54 ($\sigma = 0.66$)
GB2_HU_ZM	169	400	43.34	97.56 ($\sigma = 0.79$)
GB2_ZM	162	400	39.01	97.63 ($\sigma = 0.76$)

accuracies of the Fourier descriptor, Hu moments and complex moments are not as good as those of other descriptors for this database. It is worth mentioning that the Fourier descriptor based on centroid distance (**FD**) provides better results than the one based on complex coordinates (**FD2**), as reported by Zhang e Lu (2001). It should also be noted that an increased number of features does not necessarily yield better recognition rate, due to a phenomena known as the curse of dimensionality (PRIDDY; KELLER, 2005). For example, recognition rate associated with the **GB2_HU_FD** descriptor is lower than of the **GB2** descriptor alone.

Principal Components Analysis, retaining 95% of variance, proves effective in reducing the dimensionality of the feature vector. However, this technique, known as a filter approach to feature selection, does not necessarily improve recognition accuracy. Since the reduced vector is just a linear combination of the original, CPU time is not affected either. A wrapper approach to feature selection, such as the one presented in this work, is usually better than filters in optimizing recognition rate (KOHAVI; JOHN, 1997).

Table 2 shows the result of the t-test between feature descriptors, compared with **GB2_HU**. A confidence interval of 95% is used in the test and "+" (or "–") indicates that the accuracy achieved by a descriptor is statistically better (or worse) than that of the **GB2_HU** descriptor. Furthermore, "=" means that there is no significant difference. The data has been tested for normality by Shapiro-Wilk test with a 5% significance level.

A combination of Gabor and Zernike features provides statistically better accuracy than either one of the descriptors by itself. On the other hand, Zernike moments have a much higher computational cost. The feature vector **GB2_HU** provides fast computation, good recognition rate and a large, possibly redundant, number of features, thus being selected for further optimization.

Considering the **GB2_HU** feature vector, an experiment is performed to verify the influence of the number of neurons in the hidden layer on the recognition rate. In this experiment only the hidden layer is varied from 10 to 500 neurons. Figure 14 illustrates the relation between the number of neurons in a hidden layer and recognition rate accuracy. Despite an approximately linear growth in training time, the trained network does not take more than 1 ms to classify a single vector. Also note that recognition rate is sensible to slight adjustments in hidden layer size, which is also encoded in the NSGA-II chromosome.

Table 2 – t-test between feature descriptors

Feature vector	t-test on GB2_HU
HU	–
FD2	–
CM	–
FD	–
PZM	–
ZM	–
ZM_HU	–
ZM_25	–
GB$_{\sigma=0.2}$	–
GB2_HU_ZM PCA_95%	–
GB2_HU_FD	–
GB2_HU	=
GB2	=
GB$_{\sigma=0.02}$	=
GB2_HU_ZM PCA_100%	+
GB2_HU_ZM	+
GB2_ZM	+

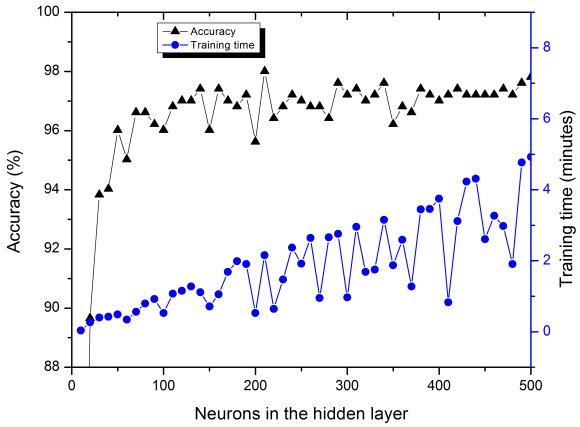


Figure 14 – Recognition rate and training time as a function of hidden layer size. The recognition time for a single feature vector varies insignificantly and is below 1 ms for any tested number of neurons in the hidden layer.

5.2 Optimization

Feature selection and neural network tuning are applied on the best feature vector, considering recognition accuracy and computational cost.

The data from Table 1 is obtained by averaging a single solution over fifty runs. On the other hand, the adapted NSGA-II algorithm described in Section 2.5 returns a set of nondominated solutions, with corresponding testing and training accuracies after each run. In order to compare these results, the fifty sets of solutions returned by the multi-objective algorithm are first combined in a single union set. In the union set, results with the same

number of features are combined into one by averaging the accuracy. Therefore, a set of average solutions is obtained from all the fifty runs. This set is called the *average* Pareto front. Besides the *average* Pareto front, the nondominated solutions in the union set are also presented.

Figure 15 compares the output of NSGA-II with the average results of the original feature vector (GB2_HU) and the most accurate feature vector (GB2_ZM). It contains the nondominated solutions (GB2_HU – A) and the *average* Pareto front (GB2_HU – B) returned by the evolutionary algorithm.

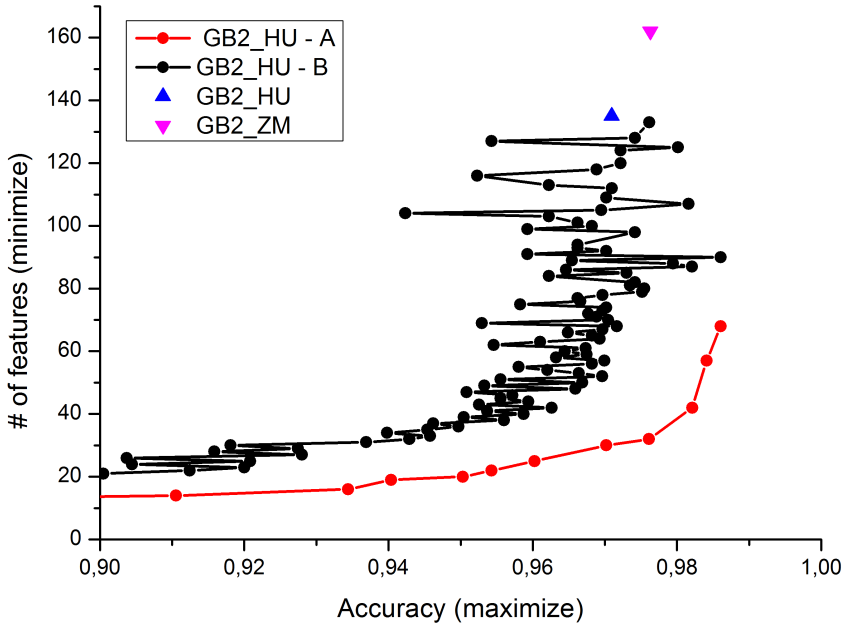


Figure 15 – Comparison of the NSGA-II output with full feature vectors.

Comparing the output from NSGA-II with GB2_HU and GB2_ZM, it can be seen that the accuracy of the *average* Pareto front (GB2_HU – B) is similar to the results contained in Table 1, but the number of features in GB2_HU – B is smaller. Also, the nondominated set includes solutions with less features and higher accuracy. The results suggest that a multi-objective technique can effectively explore the Pareto front of this problem and obtain solutions with a reduced number of features and better classification performance than those presented in Table 1.

5.3 Comparison with Existing Methods

A benchmark database (BARCZAK et al., 2011) is used to evaluate several feature descriptors with respect to accuracy and computational cost. A detailed comparison is pre-

Table 3 – Comparison with existing methods

Feature descriptor	Work	# of features	Accuracy (%)	CPU time (ms)	Strengths	Weaknesses
HU	Otiniano-Rodríguez et al. (2012)	7	37.02	0.70	Very fast and easy to implement	Low accuracy
CM	Badi et al. (2014), Hasan and Abdul-Kareem (2014)	10	48.09	2.10	Fast	Low accuracy
ZM	Otiniano-Rodríguez et al. (2012)	34	91.08	36.30	Good accuracy	High CPU time
ZM_25	Guo et al. (2015)	180	92.08	252.00	Good accuracy	Very high CPU time
GB2_HU	This paper	135	97.09	2.82	Low CPU time; high accuracy	More adjustable parameters
GB2_ZM	This paper	162	97.63	39.01	High accuracy	More adjustable parameters; high CPU time

sented in Table 3, including main strengths and weaknesses found though experimentation. The feature vectors proposed in this work outperform recent static gesture recognition techniques based on Zernike and complex moments. Gabor features are used to extract texture information from the image and have been shown to be a good combination to Zernike moments for hand posture recognition. The main weakness of Gabor features is that the Gabor filter has parameters that have to be adjusted for the specific application. While combination of Zernike and Gabor features results in good recognition rate, multi-objective analysis in section 5.2 suggests that a combination of Gabor and Hu descriptors can achieve better performance in terms of both speed and accuracy.

5.4 Real-Time Gesture Recognition

In order to apply the results obtained by multi-objective analysis to a real-time gesture recognition system, a single nondominated solution from the union is chosen. This solution has the following parameters:

- accuracy on the Test subset: 98.61%.
- number of features: 68 (from 135), including the first, fourth, sixth and seventh Hu moments.
- number of neurons in the hidden layer: 550.

The real-time system is implemented in C++, as described in Section 2.6 using the OpenCV library. It runs on a notebook with an i5 processor. As previously stated, hand segmentation is a difficult problem and is usually treated separately. In order to simplify this step, the hand is filmed against a black background, wearing a black wristband.

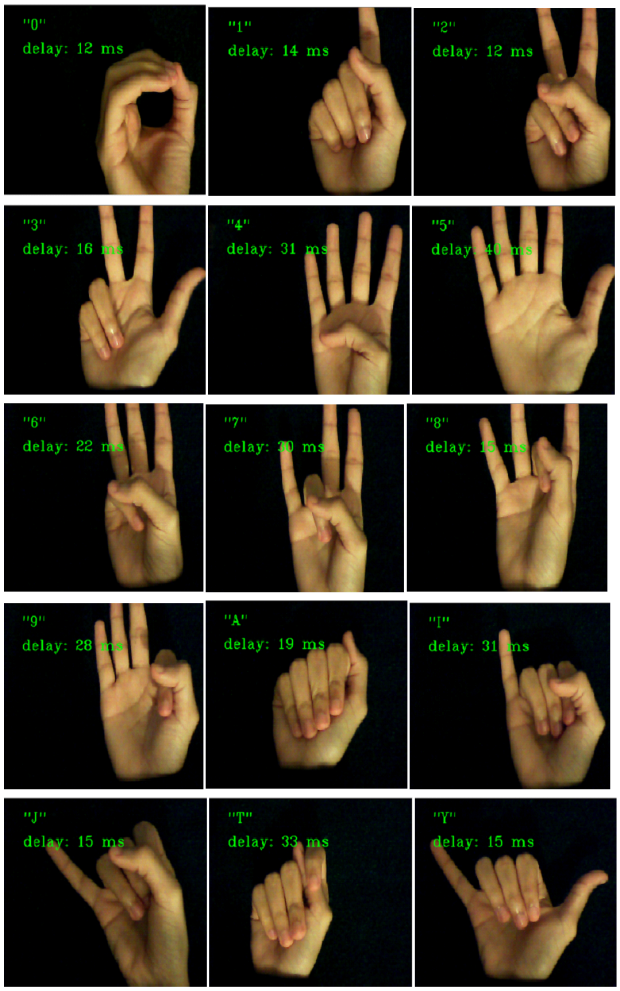


Figure 16 – Several gestures recognized in real time.

Figure 17 – An illustration of sensitivity to rotation. The gesture "Y" is only recognized correctly in the middle image.

As illustrated in Figure 7b, the delay is measured from steps 5 to 10, because step 4 is hardware dependent. On average, once the frame is acquired, preprocessing (steps 5 to 8) takes 28ms, feature extraction (step 9) takes 2ms and the neural network (step 10) takes less than 1ms to recognize the feature vector. This means that, when using a common 30fps camera, a single frame can be processed and recognized before the next frame is acquired, resulting in no noticeable delay for the user. Note that no special effort is made to optimize the image processing steps, as this is not in the scope of this paper. Additionally, since the recognition step takes less than 1ms, a more advanced classifier, such as an ensemble of classifiers, can be used.

While most of the time gestures are correctly recognized, as illustrated in Figure 16, some are very sensible to rotation. For example, the letter "Y" is recognized only when the thumb and the little finger are horizontally aligned, as can be seen in Figure 17. The recognition accuracy is also greatly affected by quality of hand segmentation. For example, a gesture is not correctly recognized if the hand is segmented with the wrist. This suggests that a larger and more robust database would be needed. It is important to notice that while the Massey database is good for benchmarking, it is not ideal for a real world application.

6 Conclusion

A hand posture recognition system is proposed, based on Gabor and Hu feature descriptors. It has a main advantage of high speed and is shown to outperform other state-of-the-art methods. The following are the main contributions of this work:

- Several commonly used feature extraction methods are tested on a benchmark dataset and compared by accuracy and computational cost for the first time.
- A multi-objective optimization scheme is presented and applied to a real-time hand posture recognition system with promising results. It is shown that both neural network size and feature vector are important for a good recognition rate.
- A real-time gesture recognition system is implemented based on the optimized configuration returned by feature selection.

Recognition accuracy of up to 98.61% is achieved for 36 gestures, while using a feature vector with low computational cost (extraction time of 2.82 ms for a 64×64 image). The proposed method outperforms other recent methods in terms of accuracy and speed. This feature vector is used to make a fast gesture recognition system.

6.1 Future work

Future work may be developed in several directions. Firstly, it is possible to add more descriptors to the feature vector and more degrees of freedom to the configuration of the neural network. The adjustment of Gabor parameters can also be included in the optimization algorithm. Other candidates for feature descriptors include Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP), Speeded up robust features (SURF) and cascade detectors.

Due to its speed, the proposed system for static hand gestures can be extended to recognize dynamic gestures. This can be done by adding dynamic information, such as position, orientation and velocity to the classified posture. A sequence of postures can be recognized as a dynamic gesture by a probabilistic method, such as Hidden Markov Model (HMM).

While NSGA-II is widely applied to multi-objective problems, other optimization algorithms have been recently proposed and could be compared to NSGA-II in the context of hand posture recognition. Unified evolutionary optimization procedure (SEADA; DEB,

2015) and improved multi-directional local search (LIAN; MILBURN; RARDIN, 2016) are suggested for this comparison.

The classification system described in this work is not specific to hand gestures. Thus, different datasets could also be evaluated and the proposed method can be applied to detection of other objects.

Finally, different classifiers are going to be evaluated for gesture recognition. Convolutional Neural Networks (CNNs) have achieved state-of-the-art results in several applications, including hand posture/gesture recognition. They have the advantage of dispensing the traditional feature extraction step, as the classifier and the convolutional feature extractor are trained simultaneously. One of the drawbacks of CNNs is the amount of memory required to store the parameters and its real-time performance. For example, a classical network for image recognition, Alex-net (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), requires about 250 MB of RAM and 1.5 million floating-point operations. Training also usually takes much longer than other machine learning methods, but this can be mitigated by parallel processing in a GPU. The performance of this network is going to be compared with traditional methods in terms of recognition accuracy and speed.

Bibliography

ABU-MOSTAFA, Y. S.; PSALTIS, D. Image normalization by complex moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, n. 1, p. 46–55, 1985. Citado na página 17.

AOWAL, M. A. et al. Static hand gesture recognition using discriminative 2d zernike moments. In: IEEE. *TENCON 2014-2014 IEEE Region 10 Conference*. [S.l.], 2014. p. 1–5. Citado 2 vezes nas páginas 25 and 27.

AVRAAM, M. Static gesture recognition combining graph and appearance features. *International Journal of Advanced Research in Artificial Intelligence (IJARAI)*, v. 3, n. 2, 2014. Citado 3 vezes nas páginas 25, 26, and 27.

BADI, H.; HUSSEIN, S. H.; KAREEM, S. A. Feature extraction and ml techniques for static gesture recognition. *Neural Computing and Applications*, Springer, v. 25, n. 3-4, p. 733–741, 2014. Citado 3 vezes nas páginas 14, 25, and 34.

BARCZAK, A. et al. A new 2d static hand gesture colour image dataset for asl gestures. Massey University, 2011. Citado 6 vezes nas páginas 7, 12, 25, 27, 28, and 39.

BHATIA, A.; WOLF, E. On the circle polynomials of zernike and related orthogonal sets. In: CAMBRIDGE UNIV PRESS. *Mathematical Proceedings of the Cambridge Philosophical Society*. [S.l.], 1954. v. 50, n. 01, p. 40–48. Citado na página 17.

BIRK, H.; MOESLUND, T. B.; MADSEN, C. B. Real-time recognition of hand alphabet gestures using principal component analysis. In: PROCEEDINGS PUBLISHED BY VARIOUS PUBLISHERS. *Proceedings of the Scandinavian Conference on Image Analysis*. [S.l.], 1997. v. 1, p. 261–268. Citado na página 11.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: ACM. *Proceedings of the fifth annual workshop on Computational learning theory*. [S.l.], 1992. p. 144–152. Citado na página 22.

BRADSKI, G.; KAEHLER, A.; PISAREVSKY, V. Learning-based computer vision with intel's open source computer vision library. *Intel Technology Journal*, v. 9, n. 2, 2005. Citado na página 27.

CARPENTER, B. et al. Mpj: Mpi-like message passing for java. 2000. Citado na página 27.

CARR, J. An introduction to genetic algorithms. *Senior Project*, p. 1–40, 2014. Citado na página 31.

CHAUDHARY, A. et al. Intelligent approaches to interact with machines using hand gesture recognition in natural way: a survey. *arXiv preprint arXiv:1303.2292*, 2013. Citado na página 25.

CHONG, C.-W.; RAVEENDRAN, P.; MUKUNDAN, R. An efficient algorithm for fast computation of pseudo-zernike moments. *International Journal of Pattern Recognition*

and Artificial Intelligence, World Scientific, v. 17, n. 06, p. 1011–1023, 2003. Citado 2 vezes nas páginas 18 and 34.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, IEEE, v. 6, n. 2, p. 182–197, 2002. Citado na página 30.

GUO, Y.; LIU, C.; GONG, S. Improved algorithm for zernike moments. In: IEEE. *Control, Automation and Information Sciences (ICCAIS), 2015 International Conference on*. [S.l.], 2015. p. 307–312. Citado 4 vezes nas páginas 12, 25, 27, and 34.

HAMDANI, T. M. et al. Multi-objective feature selection with nsga ii. In: SPRINGER. *International Conference on Adaptive and Natural Computing Algorithms*. [S.l.], 2007. p. 240–247. Citado na página 27.

HASAN, H.; ABDUL-KAREEM, S. Static hand gesture recognition using neural networks. *Artificial Intelligence Review*, Springer, v. 41, n. 2, p. 147–181, 2014. Citado 3 vezes nas páginas 17, 25, and 34.

HSE, H.; NEWTON, A. R. Sketched symbol recognition using zernike moments. In: IEEE. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. [S.l.], 2004. v. 1, p. 367–370. Citado na página 18.

HU, M.-K. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, IEEE, v. 8, n. 2, p. 179–187, 1962. Citado 2 vezes nas páginas 16 and 17.

ISHIBUCHI, H.; NOJIMA, Y.; DOI, T. Comparison between single-objective and multi-objective genetic algorithms: Performance comparison and performance measures. In: IEEE. *2006 IEEE International Conference on Evolutionary Computation*. [S.l.], 2006. p. 1143–1150. Citado na página 27.

JAIN, A. K.; FARROKHNIA, F. Unsupervised texture segmentation using gabor filters. *Pattern recognition*, Elsevier, v. 24, n. 12, p. 1167–1186, 1991. Citado na página 20.

KOHAVI, R.; JOHN, G. H. Wrappers for feature subset selection. *Artificial intelligence*, Elsevier, v. 97, n. 1, p. 273–324, 1997. Citado 2 vezes nas páginas 31 and 37.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 44.

KUMAR, B. P.; MANJUNATHA, M. A hybrid gesture recognition method for american sign language. *Indian Journal of Science and Technology*, v. 10, n. 1, 2017. Citado na página 25.

KUMAR, V.; NANDI, G. C.; KALA, R. Static hand gesture recognition using stacked denoising sparse autoencoders. In: IEEE. *Contemporary Computing (IC3), 2014 Seventh International Conference on*. [S.l.], 2014. p. 99–104. Citado na página 27.

LEUNG, F. H. et al. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *Neural Networks, IEEE Transactions on*, IEEE, v. 14, n. 1, p. 79–88, 2003. Citado 2 vezes nas páginas 15 and 26.

LIAN, K.; MILBURN, A. B.; RARDIN, R. L. An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. *IIE Transactions*, Taylor & Francis, v. 48, n. 10, p. 975–992, 2016. Citado na página 44.

LIAW, A.; WIENER, M. et al. Classification and regression by randomforest. *R news*, v. 2, n. 3, p. 18–22, 2002. Citado na página 21.

NG, C. W.; RANGANATH, S. Real-time gesture recognition system and application. *Image and Vision computing*, Elsevier, v. 20, n. 13, p. 993–1007, 2002. Citado 2 vezes nas páginas 25 and 34.

OTINIANO-RODRÍGUEZ, K.; CÁMARA-CHÁVEZ, G.; MENOTTI, D. Hu and zernike moments for sign language recognition. In: *Proceedings of international conference on image processing, computer vision, and pattern recognition*. [S.l.: s.n.], 2012. p. 1–5. Citado 2 vezes nas páginas 25 and 34.

PISHARADY, P. K.; SAERBECK, M. Recent methods and databases in vision-based hand gesture recognition: A review. *Computer Vision and Image Understanding*, Elsevier, v. 141, p. 152–165, 2015. Citado 2 vezes nas páginas 7 and 11.

PRIDDY, K. L.; KELLER, P. E. *Artificial neural networks: an introduction*. [S.l.]: SPIE Press, 2005. v. 68. Citado na página 37.

RASINES, I.; REMAZEILLES, A.; BENGOA, P. M. I. Feature selection for hand pose recognition in human-robot object exchange scenario. In: IEEE. *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*. [S.l.], 2014. p. 1–8. Citado 4 vezes nas páginas 11, 12, 25, and 27.

RAUTARAY, S. S.; AGRAWAL, A. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, Springer, v. 43, n. 1, p. 1–54, 2015. Citado na página 25.

RUMELHART, D. E. et al. Learning representations by back-propagating errors. *Cognitive modeling*, v. 5, n. 3, p. 1, 1988. Citado na página 22.

SABHARA, R. K.; LEE, C.-P.; LIM, K.-M. Comparative study of hu moments and zernike moments in object recognition. *SmartCR*, v. 3, n. 3, p. 166–173, 2013. Citado na página 25.

SEADA, H.; DEB, K. U-nsga-iii: A unified evolutionary optimization procedure for single, multiple, and many objectives: Proof-of-principle results. In: SPRINGER. *International Conference on Evolutionary Multi-Criterion Optimization*. [S.l.], 2015. p. 34–49. Citado na página 44.

SOYEL, H.; TEKGUC, U.; DEMIREL, H. Application of nsga-ii to feature selection for facial expression recognition. *Computers & Electrical Engineering*, Elsevier, v. 37, n. 6, p. 1232–1240, 2011. Citado na página 12.

TEH, C.-H.; CHIN, R. T. On image analysis by the methods of moments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 10, n. 4, p. 496–513, 1988. Citado na página 17.

TSAI, J.-T.; CHOU, J.-H.; LIU, T.-K. Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. *Neural Networks, IEEE Transactions on*, IEEE, v. 17, n. 1, p. 69–80, 2006. Citado na página 26.

WANG, C.-C.; WANG, K.-C. Hand posture recognition using adaboost with sift for human robot interaction. In: *Recent progress in robotics: viable robotic service to human*. [S.l.]: Springer, 2008. p. 317–329. Citado na página 11.

XUE, B.; ZHANG, M.; BROWNE, W. N. Particle swarm optimization for feature selection in classification: a multi-objective approach. *IEEE transactions on cybernetics*, IEEE, v. 43, n. 6, p. 1656–1671, 2013. Citado na página 27.

ZHANG, D.; LU, G. A comparative study on shape retrieval using fourier descriptors with different shape signatures. In: *Proc. International Conference on Intelligent Multimedia and Distance Education (ICIMADE01)*. [S.l.: s.n.], 2001. Citado 4 vezes nas páginas 19, 20, 34, and 37.