A Real-Time System to Recognize Static Gestures of Brazilian Sign Language (Libras) alphabet using Kinect

Mauro dos Santos Anjo, Ednaldo Brigante Pizzolato, Sebastian Feuerstack
Universidade Federal de São Carlos, Rodovia Washington Luís, km 235
São Carlos - SP - Brasil
{mauro_anjo, ednaldo, sfeu}@dc.ufscar.br

ABSTRACT
In this paper we present a system - called Gesture User Interface (GestureUI) - to recognize static gestures of the Brazilian Sign Language (Libras) in real-time out of a continuous video stream using the depth information captured by a Kinect controller. We focus on handling small sets of gestures (A,E,I,O,U) and (B,C,F,L,V), processing them in two steps: Segmentation and Classification. For the Segmentation we propose a Virtual Wall and Libras-specific heuristics to improve the hand tracking. For the classification we use a Multi-Layer Perceptron trained by the system and present an arm cutting algorithm that improved the recognition rate from 67,4% and 75,4% to 100% for both gesture sets. Finally, we evaluated the processing performance of the overall system and proof that it is able to process frames at 62.5Hz with an Intel i7 processor, which is more than twice as fast as the Kinect frame capturing rate.

Keywords
Gesture Recognition, Brazilian Sign Language, Kinect, Depth Cameras.

1.INTRODUCTION

Natural interaction through gestures is becoming popular due to recent hardware and software developments. Multi-Modal Interfaces can be significantly enhanced by using gestures to complement the already common inputs like touch, keyboard, and mouse, since it is a natural form of communication like speech.

According to the survey by Mitra [16], gesture recognition is a complex task that can be divided in two categories: static gestures and dynamic gestures. The former has to recognize postures without movement whilst the latter includes movements. The systems in the latter situation have to track the user hands or any other significant limbs over time to output a recognized gesture.

Static gestures recognition has been investigated by many researchers. Bretzner [2] presented a scale-space blob model for the hands to recognize 4 static postures. Chen [5] and Fang [8] used Haar-Like features and AdaBoost training to recognize 10 static gestures.

Moreover, Phu [16] and Wysosky [23] investigated the use of Multi-Layer Perceptron for classification, where the former obtained 92,97% with 10 postures and the latter achieved 96,7% recognition rate for 26 postures of the American Sign Language.

What all these approaches have in common is that they focused on the evaluation of the overall performance (mean recognition rate) of the system with large datasets, without providing information about real-time or individual gestures recognition rates along with the feasibility of their segmentation approaches in real world applications.

Furthermore, especially for multimodal interaction, real-time processing of gesture is a basic requirement to sync it with other modes like voice or body movements for instance. Some researchers have addressed real-time in different applications like human robot interaction at 5-10fps [], dynamic gestures [], dynamic and static gestures at 10fps [], but unfortunately they achieved low frame rates.

With the recent development of Microsoft's Kinect for XBox gaming console and its PC version, segmenting and tracking parts of the human body has become easier, and a great variety of systems has been implemented in different applications like: Healthcare [4]; Augmented Reality Environments [21]; Gesture Recognition [18]; among others.

In this paper we focus on presenting a system that can be trained to recognize static gestures out of a video stream and report them to any interface through a TCP/IP protocol in real-time. To evaluate the performance of the system we chose the Brazilian Sign Language (Libras) alphabet as use case.

Libras is recognized as an official language in Brazil since the Brazilian government enacted the law nº 10.436 in 2002. Furthermore, Libras has recently become obligatory as subject in Teachers Education through the enactment nº 5626 of 2005. This enactment also states that until 2015 all Brazilian undergraduate courses must offer Libras as an elective subject.

The paper is structured as follows: In section 2 we explain the challenges of segmentation and our solution to this problem. Thereafter, section 3 covers our static gesture classifier. Next, in section 4 the Gesture User Interface

(GestureUI) is presented followed by section 5 that presents our methodology to evaluate our approach. In section 6 the results are presented and discussed. Finally, section 7 concludes our discussion and points some future investigations.

Fig. 1 - a) Microsoft Kinect schematic; b) Kinect's Infra-Red projection pattern.

2.SEGMENTATION

The interpretation of human actions in computer systems is a complex task due to problems of segmenting human figures in a scene and tracking body parts with a high degree of freedom.

The difficulty of the segmentation is the attempt to handle all the variables or issues that rise in vision systems, so called environment and application domain variables, like: illumination; cluttered backgrounds; clothing; occlusions; velocity of movement; quality and configuration of cameras and lens; among many others.

In order to make the systems simpler, the most common approach used by researchers is to eliminate some of the variables by stating system constraints or using extra devices like in the following situations:

- **Controlled environments:** to eliminate some environment variables, for instance avoiding cluttered backgrounds by using a plain wall or Chroma Key and constraining user clothing permits segmentation by simple thresholding [17].
- **Colored markers:** to track a specific body part a common approach is the use of a colored marker. For instance a colored glove to track a user hand and segmenting it by thesholding in a color space model [1, 9, 7, 10].
- **3D Gloves:** sometimes a great precision of tracking is needed, like finger movements. In this case an extra device is needed, like a 3D Glove that embeds gyroscopes and accelerometers to track users movements [22, 24].

Since our objective is to interact naturally we must avoid extra wearable devices or specific restrictions of environment.

To do so, researchers tried to abstract how humans perceive objects in space. They found out that what eases human perception of objects is the Stereopsis or impression of depth that we can perceive due to human binocular vision. Researches tried to replicate that in computer systems with an approach called Stereo-Vision.

The Computer Stereo-Vision is obtained by a pair of visual cameras trying to estimate depth information using stereo correspondence algorithms [25].

The major problem of this approach is that it relies on visual properties (corners, edges, color), and all of them are prone to problems due to illumination variation, camera and lens quality, environment, clustered backgrounds, among others. Further on, this reduces the accuracy of the estimation and also requires hardware level code, like GPU or even Field-Programmable Gate Array (FPGA), to achieve real-time data.

Another solution is to use depth cameras, that instead of visual spectrum they use the infra-red spectrum to avoid illumination and other environment constraints. These cameras are a recent development, and have achieved good results in depth estimation in a range of 80cm to 4m.

Microsoft Kinect[1] is one example of such devices. It is a product initially developed for the XBox 360 gaming console with the purpose of enabling the user to interact with the games using nothing but body movements. Kinect is a device with an Infra-Red (IR) projector, a VGA camera and an IR camera like shown in Figure 1 (a).

Kinect estimates the depth of the objects in the scene doing a stereo-vision approach, like aforementioned, in the IR spectrum. This technology was developed by the Israeli company PrimeSense[2].

The IR Projector projects a known dotted pattern in the environment, shown in Figure 1 (b), and the IR camera (along with the embedded algorithm patented by PrimeSense) can estimate depth with a resolution of 640x480 pixels.

2.1. Virtual Wall and Hand Tracking algorithm

In our approach we use the Kinect controller to enable user interaction in indoor non-controlled environments.

[1] http://www.xbox.com/Kinect/

[2] http://www.primesense.com/

PrimeSense provides a middleware called NITE that enables the system to segment the users figures in the scene and to obtain: a) a mask with all user pixels and b) the user Center of Mass (CoM).

Before we explain our algorithm for hand tracking and segmentation we need to state some domain information about Libras. When the person is performing the gesture, the hands are always in front of the torso and sometimes there is contact with the users face or even the torso itself. Furthermore, in Libras the person can perform a gesture with the dominant hand or both hands. The dominant hand is the right one for right-handed people and the left one for left-handed people.

In this context we developed a simple algorithm to segment user hands called Virtual Wall that sets a depth threshold that works like an invisible wall in front of the user. The position of this wall is calculated using the users CoM in the following way:

$$VW_{depth} = CoM_{depth} - \alpha$$

where α is an offset to avoid that the Virtual Wall is transposed by the face or torso. It can be set with an empiric value depending on how the application will be used.

With this approach the user can move around and the Virtual Wall will follow him/her.

After obtaining only the blobs that are in front of the Virtual Wall by thresholding the depth map, the image is binarized.

We then extract the blobs that correspond to the hands of the user from the binary image and eliminate some undesired noise blobs.

First one has to find the blobs, or connected components, positions and calculate their area. To do so a linear-time component labeling algorithm [3] is applied.

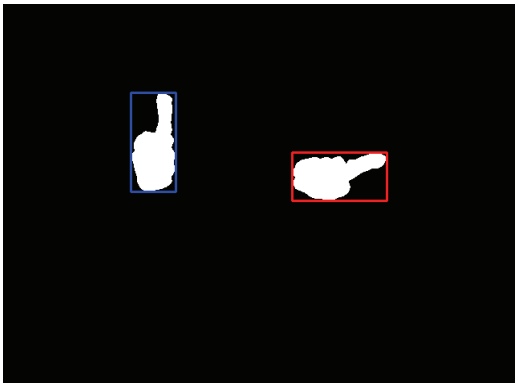

Fig. 2 – Output of the Segmentation algorithm with Right and Left hands Regions of Interest.

After obtaining the list of blobs, we created a simple algorithm to track the hands based on the aforementioned domain restriction of Libras:

1) Sort blobs detected in descending order of area;
2) Select the one or two major blobs as hands;
3) If there is only one blob, label it as dominant hand;
4) If two blobs are visible, label the leftmost as left hand and the rightmost as the right hand;
5) Define the Region Of Interest (ROI) of the present hands as the minimum rectangle that encloses the blobs, and calculate the CoM of each hand.

The outcome of the complete algorithm can be seen in Figure 2 that shows one user with both hands transposing the Virtual Wall.

2.2. Aspect Ratio Hand Cropping Heuristic

One problem of classification that arises when applying the Virtual Wall method is the appearance of the arm as a gesture is performed. To tackle this issue we introduced a heurist to eliminate or at least reduce the visible arm. We call this method Aspect Ratio Hand Cropping Algorithm (ARHCA) and describe it in the following paragraphs.

This algorithm tries to eliminate the arm by a simple aspect ratio check. When performing static gestures from Libras alphabet, the users' arm is, most of the time, positioned orthogonally in relation to the horizontal axis.

Considering this we defined a fixed aspect ratio to reduce the Region of Interest (ROI) in the following way:

$$Aspect\ Ratio = \frac{ROI_{height}}{ROI_{width}}$$

$$ROI_{height} = \begin{cases} CoM_y + \alpha, & Aspect\ Ratio > \beta \\ ROI_{height}, & otherwise \end{cases}$$

where β is the aspect ratio threshold desired and α is an offset in relation to the y coordinate of the Center of Mass of the ROI (CoM_y). These parameters can be changed in real-time but empirically we defined $\beta = 1{,}34$ and $\alpha = 1{,}05 * ROI_{height}$.

After recalculating ROI_{height}, the ROI_{width} needs to be adjusted so the resulting ROI keeps being the smallest rectangle enclosing the blob of the hand.

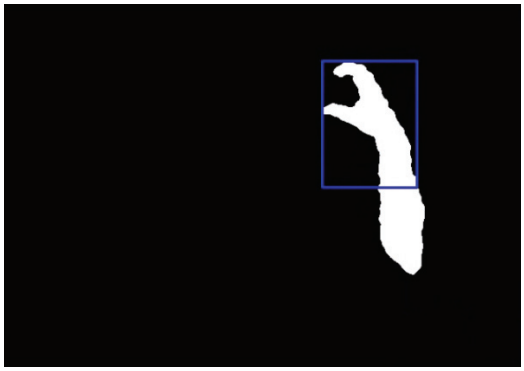

Fig. 3 – Sample output of the Hand Cropping Heuristic

A sample result is shown in Figure 3, where most of the arm is visible and the algorithm redefined the ROI.

3.STATIC GESTURE CLASSIFIER

Artificial Neural Networks (ANNs) have been widely used to solve pattern recognition problems in a great variety of applications like time forecasting [11], medical diagnosis [12], astronomy [6], gesture recognition [14], among others.

The Multi-Layer Perceptron (MLP) [20] is a feed-forward ANN that is an evolution of the Standard Perceptron [19]. The purpose of this statistical model is to learn to distinguish between non-linearly separable data.

MLP is composed by neurons disposed in input, hidden and output layers; its structure is of a directed graph with all neurons of one layer fully connected with the next layer.

Each neuron (n) has an activation function that processes all the weighted inputs (s_n). This function must be non-linear, and as stated in Karlic work [13] the best one is the hyperbolic tangent function ($\tanh(x)$). So the activation of a neuron ($f_n(\cdot)$) in our software is calculated as follows:

$$s_n = bias_n + \sum_{j=0}^{N} w_{nj} * y_{nj}$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$f_n(\cdot) = \tanh(s_n)$$

where N is the number of inputs in the previous layer.

The MLP learns to solve the pattern classification problem by the use of a learning algorithm that tries to model the mapping of inputs, with a predefined dimension, in a subset of possible outputs.

The Learning algorithms are manifold, so here we focus in the one used in our experiments that is a supervised learning algorithm, we chose it to reduce training time and to improve results of training since all data in training sets was already labeled.

This learning algorithm is called Supervised, because when in training mode each pattern presented has a label corresponding to the expected output of the network. All the knowledge of the network is stored in the weights of the connections between neurons, and in the learning process they are updated until convergence is reached. We use the Backpropagation algorithm [13] to update these weights.

Backpropagation consists in presenting the patterns to the network and does a forward pass in which every neuron is processed until the final layer is reached. With the final layer results we can calculate the error given the desired outputs, and backpropagate weight corrections (Δw_{nj}) using the following formulas:

$$\Delta w_{nj} = \eta * o_n * \delta_n$$

$$\delta_n = o_n * (1 - o_n) * (d_n - o_n) \quad (output\ layer)$$

$$\delta_n = o_n * (1 - o_n) * \sum_{j=1}^{N} w_{nj} * \delta_j \quad (hidden\ layers)$$

where η is the learning rate, o_n is the output associated with the weight, and d_n is the desired output.

With this approach we penalize weights that maximize the error and reward weights that minimize the error. This is done until the total average error of the training inputs is minimized. This may take some time depending on the learning rates defined.

To avoid local minima stagnation when increasing the learning rate to reduce time of training, Rumelhart [20] proposes a modification to the weight update formula, to take into account the weight update of the previous iteration using the termum momentum (μ):

$$\Delta w_{n(j+1)} = \eta * o_n * \delta_n + \mu * \Delta w_{nj}$$

with it we avoid great steps in the gradient descent direction that could lead the MLP to a local minima.

The learning rates and momentums used in our experiments are defined in Table I.

Table I – Layers Learning Rates and Momentums

Layer	η	μ
Input	0.15	0.5
Hidden	0.09	0.6
Output	0.05	0.7

Before the execution of the training algorithm the weights of the network must be initialized. The most promising initialization method is the Nguyen-Widrow [15], in which the weights are first initialized with random values between -1 and 1. Then β and the Euclidian Norm (n) are calculated for all randomly initialized weights to finally update them as follows:

$$\beta = 0.7 * h^{\frac{1}{i}}$$

$$n = \sqrt{\sum_{i} w_i^2}$$

$$w_{t+1} = \frac{\beta * w_{t+1}}{n}$$

where h is the number of hidden layers and i the number of inputs.

We based our classifier on previous work [17] where 27 signs of the Libras alphabet were classified by a MLP to feed Hidden Markov Layer models trained to recognize fingerspelling of words. The mean recognition rate in the aforementioned work was 90,7% for the static gestures.

In our previous work we used binary images of 25x25 pixels as input due to characteristics of that study. Here we use the same structure, despite not being the best statistical option due to the high dimensionality of the inputs. This is not a problem for small sets of gestures, as we describe later on.

The basic structure of our classifier, represented by figure 4, is an input layer of 625 inputs, followed by one hidden-layer with 100 neurons and an output layer with 5 possible classes.

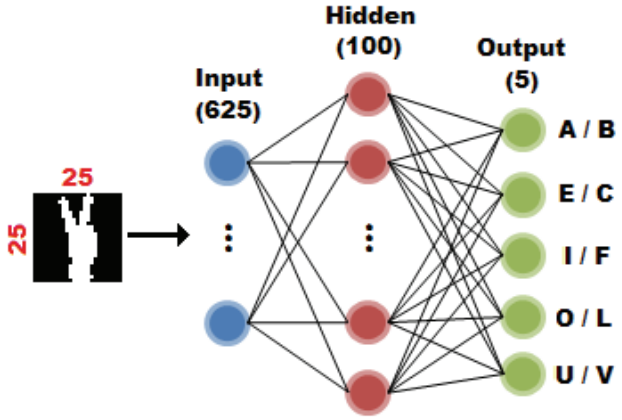

Fig. 4 – Architecture of the MLP Classifier

4.GESTURE USER INTERFACE

To ease our research activities we developed a tool to encapsulate all the work that has been done and also to ease the implementation of additional applications.

This tool is called Gesture User Interface (GestureUI) and it is able to create and configure a complete system for gesture recognition.

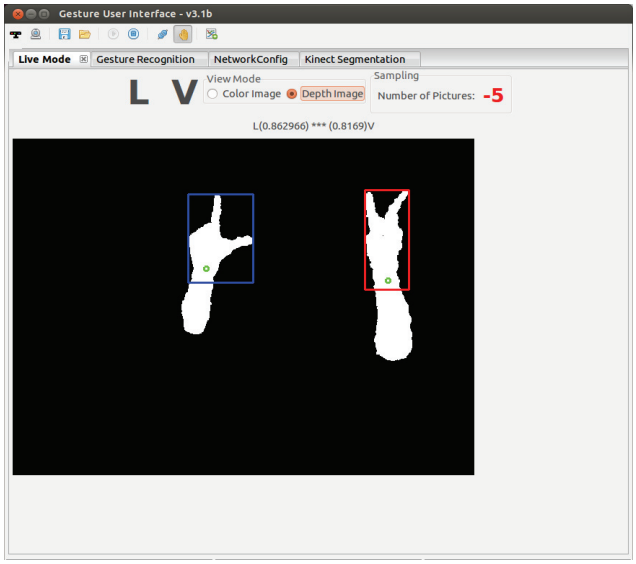

Fig. 5 – GestureUI recognizing static gestures L and V.

When creating a new interface in GestureUI the user has to choose the Segmentation approach between two categories:

- **Visual Camera:** In this mode, the software uses a monocular system that tracks colored gloves. The user can define a different color for each hand. The system is able to detect the markers by thresholding the HSV color-space model and has been presented in [17].
- **Kinect:** This mode provides automatic segmentation of the user, and uses the Virtual Wall algorithm (detailed in previous sections) to segment and track user hands.

Afterwards the user can train new models for static gesture recognition, or use previously trained ones.

To train new models the training and testing set must be representative within possible variations of user input. To ease this task the software is able to collect samples in real-time using a sequential trigger that snaps the images of the postures and organize them in folders by gesture name.

These samples can be used to train a new MLP model, letting the user configure all the properties of the ANN.

With the Segmentation and Gesture Recognition configured the user can start the Real-Time execution mode, that feeds the MLP with the current hands postures, and the outputs with the recognized static gestures are shown in the screen. In Figure 5, a user presents two signs (one on each hand) and the system outputs the corresponding letters (at the top of the figure). In the aforementioned example, the leftmost sign is a L while the rightmost is V.

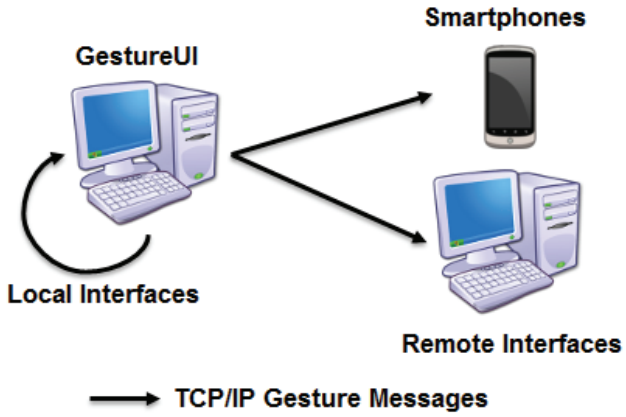

Fig. 6 – Possible applications of GestureUI TCP/IP Protocol

GestureUI also provides a TCP/IP Frame-Based Protocol in which the user can define the IP and Port of the Host where the controlled interface will be, and a message will be sent for each image frame analyzed. With this protocol one can handle gesture interactions in any device inside a network like shown in figure 6.

The message defined in the static gesture recognition protocol is structured with a string separated by semicolons that has the information about left hand followed by the info about the right hand, as follows:

$$G^{right};P_x^{right};P_y^{right};P_z^{right};G^{left};P_x^{left};P_y^{left};P_z^{left}$$

where G stands for the gesture recognized for each hand and P_x, P_y, P_z represent the 3D position of the hand that is the same as the hand center of mass.

When the hand is present and a valid gesture is recognized the system returns the label of the gesture in G. If the hand is present but the posture is not valid the system returns "UNKNOWN". Finally, if the hand is not present G is set with "MISSING" and $P_x = -1$, $P_y = -1$, $P_z = -1$.

The system has already been evaluated, like the MINT framework [1, 11], where GestureUI was used to control a web interface using static hand postures. That version used the colored gloves segmentation system and a set of 6 static gestures.

GestureUI was implemented with multi-plataform frameworks and tools. For image processing and visual spectrum cameras frame-grabbing we used OpenCV[3]. The interface was developed in C++ using wxWidgets[4] in the Code::Blocks IDE[5]. For integration with Kinect or any other device with the PrimeSense depth sensor we used the middleware OpenNI[6].

[3] http://opencv.willowgarage.com

[4] http://www.wxwidgets.org/

[5] http://www.codeblocks.org/

[6] http://openni.org/

The Next Section presents the methodoly used to evaluate our approach.

5.METHODOLOGY

In this paper we focus on the evaluation of the system in the task of segmenting and recognizing static gestures in real-time using Kinect.

To evaluate time and recognition performance we used a sample application developed to present our work of Libras Recognition in the Educomp[7] realized in 2011. In the seminar we presented to Libras students and teachers our system of static postures recognition working in real-time.

As state before we base our classifier in a previous work [17] that achieved 90,7% recognition rate. However, as this value is a is a mean of 27 gestures recognition rates, individual gestures with low recognition rates will not perform well in a Real-Time system due to misclassification. Furthermore, some static gestures only differ from each other by small details and to solve this problem in our previous work we used a two level architecture in which the second-level MLPs were responsible for this finer detail classification using other visual cues like Edge Detection. As our focus is to evaluate the performance on real-time, we chose small groups of gestures to train our classifiers, avoiding higher complexities in classification.

Therefore we trained two MLP models to recognize the following groups: 1) A, E, I, O, U and 2) B, C, F, L, V. The former is a set containing all the vowels of the alphabet. The latter we chose only 5 consonants, so the set has the same size as the one with vowels. Furthermore, these consonants do not need finer detail classification simplifying our MLP architecture.

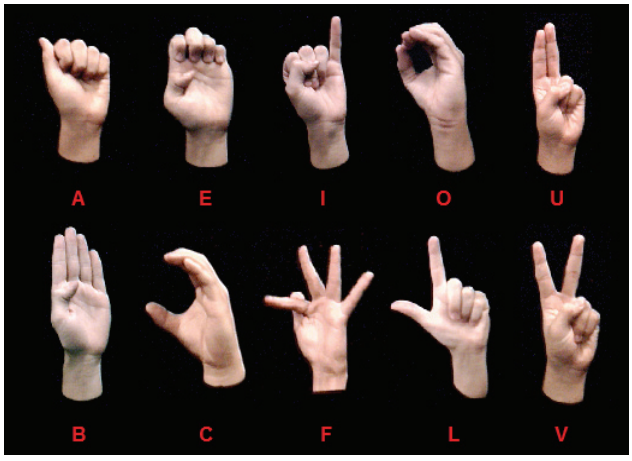

Fig. 7 – The static gestures of Libras alphabet used for training the ANNs.

We gathered a training set of 150 images and a testing set of 250 images of each static gesture in Figure 7. The testing set was never presented to the MLP during training process.

[7] http://www.educomp2011.com/

The datasets were created in order to represent possible small rotations and also scale variance that users may present at real world situations. This eases the generalization capacity of the MLP classifiers.

Furthermore, as we obtain images with variable scale using our segmentation algorithms we have to prepare all images to be used as input for the MLP classifier. To do so one should do the following procedure (illustrated in Figure 8):

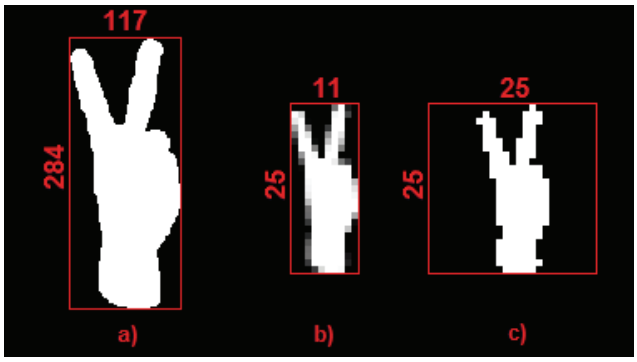

Fig. 8 – Sequence of operations to resize image: a) detected ROI; b) proportional resize; c) centered final result.

1. Calculate the aspect ratio of the ROI image and resize proportionally trying to achieve 25 pixels in at least one dimension.
2. The resized image will not be binary anymore due to approximations of the resizing algorithm, so it must be thresholded.
3. If the resized image has one of its dimensions different from 25, center it in a 25x25 image.

As explained earlier, during the test of the recognition we figured out that we got lower recognition performance if parts of the arm were visible in the image. An example for this problem can be seen in figure 9 in a) and c) where the arm is present and becomes part of the hand ROI. Due to 25x25 resizing we lose resolution in the hand representation that is the most important part for the MLP to distinguish between gesture classes.

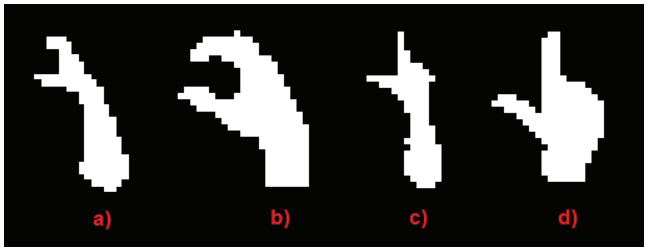

Fig. 9 – Variation of details without ARHCA in a) and b); With ARHCA in b) and d).

So we collected an additional testing and training set using the Aspect Ratio Hand Cropping Algorithm, presented in Section 2.2, to improve the detail level in the images as can be seen in b) and d) of Figure 9. In this way we can compare performances of recognition using both datasets.

To evaluate the execution time of the whole static gesture recognition system, we subdivided the steps of execution in three categories:

- **Segmenting:** involves capturing the depth map, detecting user position and the Virtual Wall algorithm.
- **Analyzing:** time to detect blobs; extract hands Region of Interest; and prepare images to feed the ANNs (resizing and centering).
- **Recognizing:** time for the trained MLP to process the input gesture and output a classified category.

The measurement of these categories times was performed using the POSIX Time Boost Library[8] with a microsecond (μsec) precision clock.

6.RESULTS

To evaluate the recognition performance of the classifier we had to isolate it from the whole system to perform a batch mode testing. This was done due to the characteristic of our real-time system that outputs the most probable gesture for every frame processed. As the MLP is a model trained to classify only the presented patterns in training mode, it is not able to take the best decision when an unknown gesture is presented due to unpredictable mappings inside the architecture of this statistical model.

In this context, when the user is changing the configuration of the hand posture the system may output some unreliable classifications until the user reaches the desired posture.

To avoid these problems in the evaluation of our recognition performance, we used the testing datasets that simulate user interaction by eliminating the intermediate transition postures. As mentioned in the earlier section, we gathered two sets, one without ARHCA algorithm and the other with.

Each set is formed by two groups of gestures: 1) A, E, I, O, U and 2) B, C, F, L, V. Each gesture is composed with 250 images with hand cropping and 250 images without hand cropping, totalizing 5000 samples.

We presented each set of images to the trained MLPs to evaluate the performance using or not the ARHCA algorithm. The results obtained are presented in Table II and Table III.

Table II – Recognition rates of A, E, I, O, U

Static Gesture	Recognition No ARHCA	Recognition ARHCA
A	65%	100%
E	60%	100%
I	74%	100%

[8] www.boost.org/doc/libs/1_50_0/doc/html/date_time/

O	70%	100%
U	68%	100%
Mean	**67,4%**	**100%**

Table III – Recognition rates of B, C, F, L, V

Static Gesture	Recognition No ARHCA	Recognition ARHCA
B	73%	100%
C	77%	100%
F	80%	100%
L	73%	100%
V	74%	100%
Mean	**75,4%**	**100%**

One can see from the results a major improvement when the ARHCA algorithm was used to collect sample data. Also we can see that the second MLP performed better, due to the visually perceptible discrepancy between patterns of B, C, F, L, and V that eased the discrimination of the gestures even with resolution loss.

In real-time testing, when performing the gestures sequentially we can perceive smooth classification of gestures with tolerance to small rotations, and bigger variances in scale, confirming that the training sets were enough to represent the variations within the gestures.

We also evaluated the execution time of the system according with the categories listed in the previous section. We gathered 5 streams of videos with 25 seconds each, in which a user was performing random gestures with both hands while moving towards and backwards in front of the Kinect. The computer used for testing was an Intel i7 processor with 6GB of RAM.

Table IV – Mean Times of execution along with the standard deviation of each category

Task	Mean	Standard Deviation
Segmenting	7295.17 µsec	3169.06 µsec
Analysing	2180.62 µsec	724.56 µsec
Recognizing	651.26 µsec	131.27 µsec

With the streamed videos we obtained 3750 measurements of frame processing. The measured mean times and its standard deviations are presented in Table IV. Furthermore, Figure 10 shows the frequency distribution of each category time of execution represented by a histogram.

One can notice the fluctuation of Segmenting execution due to variation in the scene observed by the Kinect while the user moves his hand and body, in contrast with the low variation of Analysing and Recognizing steps.

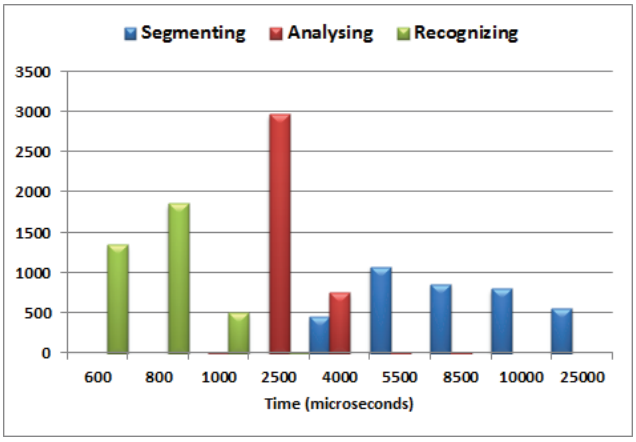

Fig. 10 – Histogram of times of execution per task

Summing the means of execution time we obtain 10127.05µsec that is the time necessary for the system to analyze one single frame, meaning that the system can process frames with a frequency of 98.81Hz. However, there is an overhead of about 6 milliseconds to update the interface and give feedback to the user. So the system is actually able to process frames at 62.5Hz.

Unfortunately, Kinect is only able to output 30 frames per second, consequently the system is limited by 30Hz.

7.CONCLUSION AND FUTURE WORK

In this paper we detailed our experiments with segmentation and recognition of static gestures in real-time using as use case the Libras alphabet. We obtained a recognition rate of 100% for A, E, I, O, U and B, C, L, F, V using the Aspect Ratio Hand Cropping algorithm along with the Virtual Wall algorithm. The overall execution frequency of the system was 30Hz limited by the Kinect device frame rate, however the system is able to reach 62.5Hz.

We also presented the software GestureUI that encapsulates all the approaches showed and a TCP/IP Protocol to ease integration with third party interfaces.

As future work we intend to add dynamic gesture recognition to GestureUI set of functionalities. This is already being implemented focusing on Libras words recognition.

ACKNOWLEDGMENTS

We would like to thank FAPESP and CNPQ for sponsoring our work.

REFERENCES

1. Bellarbi, A., Benbelkacem, S., Zenati-Henda, N., Belhocine, M. Hand gesture interaction using color-based method for tabletop interfaces. IEEE 7th

International Symposium on Intelligent Signal Processing (WISP), 2011.

2. Bretzner, L., Laptev, I., Lindeberg, T., Lenman, S., Sundblad, Y. A Prototype System for Computer Vision Based Human Computer Interaction. Technical Report, Department of Numerical Analysis and Computing Science KTH (Royal Institute of Technology), Stockholm, Sweden, 2001.
3. Chang, F., Chen, C., Lu, C. A linear-time component-labeling algorithm using contour tracing technique. Elsevier Computer Vision and Image Understanding, Volume 93, Issue 2, Pages 206–220, 2004.
4. Chang, Y., Chen, S., Huang, J. A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. Elsevier, Research in Developmental Disabilities, Volume 32, Issue 6, 2011.
5. Chen, Q., Georganas, N. D., Petriu, E. M. Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar. IEEE Transactions On Instrumentation And Measurement, Vol. 57, No. 8, 2008.
6. Ciaramella, A., Donalek, C., Staiano, A., Ambrosio, M., Aramo, C., Benvenuti, P., Longo, G., Milano, L., Raiconi, G., Tagliaferri, R., Volpicelli, A. Applications of neural networks in astronomy and astroparticle physics. Research Signpost, Recent Res. Devel. Astrophysics, 2005.
7. Cole, J. B., Grimes, D. B., Rao, R. P. N. Learning Full-Body Motions from Monocular Vision: Dynamic Imitation in a Humanoid Robot. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007.
8. Fang, Y., Wang, K., Cheng, J., Lu, H. A Real-Time Hand Gesture Recognition Method. IEEE International Conference on Multimedia and Expo, 2007.
9. Feuerstack, S., Anjo, M. S., and Pizzolato, E. Model-based Design, Generation and Evaluation of a Gesture-based User Interface Navigation Control. 5th Latin American Conference on Human-Computer Interaction (IHC 2011), Porto de Galinhas, Brazil, October 25-28, 2011.
10. Feuerstack, S., Anjo, M. S., Colnago, J., Pizzolato, E. Modeling of User Interfaces with State-Charts to Accelerate Test and Evaluation of different Gesture-based Multimodal Interactions. Accepted for Workshop "Modellbasierte Entwicklung von Benutzungsschnittstellen (MoBe2011)", Informatik 2011, 4-7. Berlin, Germany, October 2011.
11. Gardnera, M. W., Dorlinga, S. R. Artificial neuralnetworks (the multilayer perceptron) - a review of applications in the atmospheric sciences. Atmospheric Environment, Volume 32, Issues 14–15, Pages 2627–2636, August 1998.
12. Hayashi, Y., Setiono, R. Combining neural network predictions for medical diagnosis. Computers in Biology and Medicine, Volume 32, Issue 4, Pages 237-246, July 2002.
13. Karlik, B., Olgac, A. V. Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks. International Journal of Artificial Intelligence and Expert Systems, Volume 1, Issue 4, pages 111-122, 2010.
14. Mitra S., Acharya, T. Gesture Recognition: A Survey. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, Vol. 37, No. 3, May 2007.
15. Nguyen, D., Widrow, B. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. Proceedings of the International Joint Conference on Neural Networks, 3:21–26, 1990.
16. Phu, J. J., Tay, Y. H. Computer Vision Based Hand Gesture Recognition Using Artificial Neural Network, in Proc. Int'l Conf. on Artificial Intelligence in Engineering and Technology (ICAIET'06), Kota Kinabalu, November 2006.
17. Pizzolato, E. B., Anjo, M. S., and Pedroso, G. C. Automatic recognition of finger spelling for LIBRAS based on a two-layer architecture. In Proceedings of the 2010 ACM Symposium on Applied Computing, ACM, 969-973, 2010.
18. Ren, Z., Meng, J., Yuan, J., Zhang, Z. Robust hand gesture recognition with kinect sensor. Proceedings of the 19th ACM international conference on Multimedia, pages 759-760, 2011.
19. Rosenblatt, F. The Perceptron - a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory, 1957.
20. Rumelhart, D. E., Hinton, G. E., Williams, R. J. Learning Internal Representations by Error Propagation. Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations. MIT Press, 1986.
21. Santos, E. S., Lamounier, E. A., Cardoso, A. Interaction in Augmented Reality Environments Using Kinect. XIII Symposium on Virtual Reality (SVR), 2011.
22. Weissmann, J., Salomon, R. Gesture recognition for virtual reality applications using data gloves and neural networks. IEEE International Joint Conference on Neural Networks, 1999.
23. Wysoski, S. G., Lamar, M. V., Kuroyanagi, S., Iwata, A. A Rotation Invariant Approach On Static-gesture Recognition Using Boundary Histograms and Neural Networks. Proceedings of the 4th International Conference on Neural Information Processing , V. 4, 2002.

24. Yang, A. Y., Iyengar, S., Sastry, S., Bajcsy, R., Kuryloski, P., Jafari, R. Distributed Segmentation and Classification of Human Actions Using a Wearable Motion Sensor Network. IEEE Computer Vision and Pattern Recognition Workshops, 2008.

25. Yang, R., Pollefeys, M. Multi-resolution real-time stereo on commodity graphics hardware. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003.