# Parallel implementation of a gesture recognition algorithm

Author: Pavel Agrigoroaei

## 1. Gesture recognition algorithm

This hand gesture recognition algorithm is based on a chapter from the book by K. Kraiss[1] and uses the AForge.Net[2] framework for capturing frames from a webcam in real time.

Using a webcam frames are captured, processed for detection of the hand, features extracted and using the features the gestures are identified. There are 3 types of gestures that can be recognized: stop, right and left as seen in figure 1. The program was designed in C# with .Net Framework 4 and uses it’s threading support[3] to implement a parallel version of the algorithm.

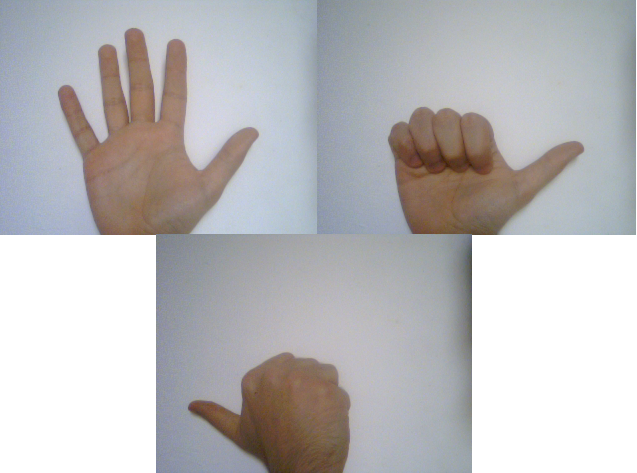


Figure – The three gestures: Stop, Right, Left

The algorithm steps are presented below and in figure 2:

* frame capture using AForge .Net Framework
* binarization of the image using a threshold method
* threshold filtering of the binary image using one upper and lower value
* region description (contour extraction)
* feature extraction
* gesture recognition

The two features that are used for identifying the three gestures are compactness and protrusion ration. Based on the values of these two parameters the hand gesture can be identified and the result shown to the user.

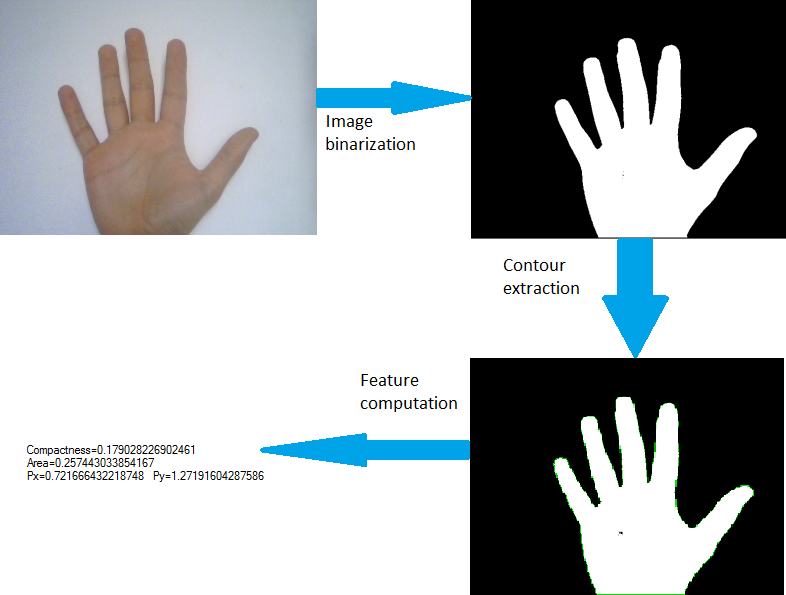


Figure – Gesture algorithm steps

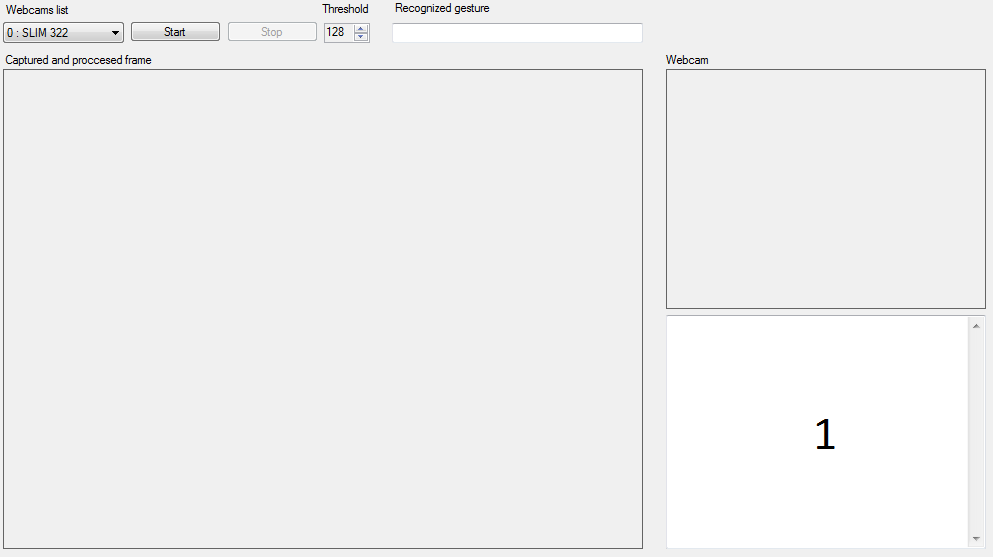


Figure – Graphical interface of the application

## 2. Interface

An overview of the graphical interface can be seen in figure 3 and contains the following sections:

* **Webcam list:** Displays the video capture devices detected on the system by AForge .Net.
* **Start and stop buttons:** Are used for starting and stopping the acquisition process.
* **Threshold:** The control is used for setting the threshold for the binarization process permitting the user to accommodate for different conditions.
* **Webcam:** The frames captured from the webcam, binarized and filtered are shown in that area
* **Captured and processed frame:** The area is used for displaying the current captured frame and the contour is highlighted. The frames are captured at a rate of one frame per second.
* **Recognized gesture:** The text box is used for displaying the current gesture that has been recognized. Values are: Stop, Right, Left, Not recognized.
* **Area 1:** The text box is used to display information about the current process and the features extracted.

**3. Parallel implementation**

Four steps of the algorithm have been chosen to be implemented in a parallel form to meet the systems real time constraint. Three of the steps are configured to detect the number of logical processors on the system and to start threads according to that number.

To implement a parallel version of the binarization process, which is an operation carried out on single pixels, the matrix containing the image was divided into lines and each thread processes one line (see figure 4). When a thread completes its task another is started and the process continues until all of the image has been converted. The same technique is used to copy the resulting image back to the original bitmap and displayed on screen.



Figure – Thread allocation for image binarization and back copy

For the task of filtering some of the noise present in the binary image a two threshold filter was used. A similar process to the one above was used to implement this step the only difference was that a window for the filter is used. For the result of a thread to not be influenced by another thread, two images were used, one for temporary storage of the result.



Figure – Thread allocation for binary image filtering

The step to use parallel implementation is the feature computation step. The features that are needed for the gesture recognition process: compactness and protrusion ratio. To compute these features another set of features must be calculated first: xMax (maximum coordinate on the X axis), xMin, yMax, yMin, area, length of the border, center of gravity position on the X axis and center of gravity position on the Y axis. To accelerate this task, the set was split into groups of equivalent complexity and a thread was allocated for each one (see figure 6).



Figure – Thread allocation for feature computation

## 4. Results

The application was written in C# using .Net Framework 4 and utilizes AForge .Net Framework for interfacing with the webcam. For the program to be compiled the compiler must be set to allow unsafe code to be run, this is needed to accelerate the image processing steps. The program requires the AForge references: AForge.Controls, AForge.Video and AForge.Video.DirectShow.

To run the algorithm, select a webcam from the list. If no webcam is present try another webcam, reinstalling webcam drivers or search the AForge website for possible solutions. The start button starts the acquisition process, the image binarized, filtered and displayed in the webcam area. Once per second a frame is captured, the region is identified, the features are computed and a hand gesture is identified, if no gesture can be found a message will be displayed. Stopping the acquisition process is done with the stop button.

Getting a good capture of the hand is heavily depended on the acquisition conditions. For optimal result the hand should be in front of a white background and sufficient illumination be present to obtain a good contrast of colors. The threshold parameter allows tweaking the binarization step so that only the hand is shown. The hand in the captured and processed area should have a green outline to indicate that the correct region was identified. If the hand doesn’t have the outline then try moving the webcam, changing the background or adjusting the threshold.

A couple of result of running the software can be seen in the following images.

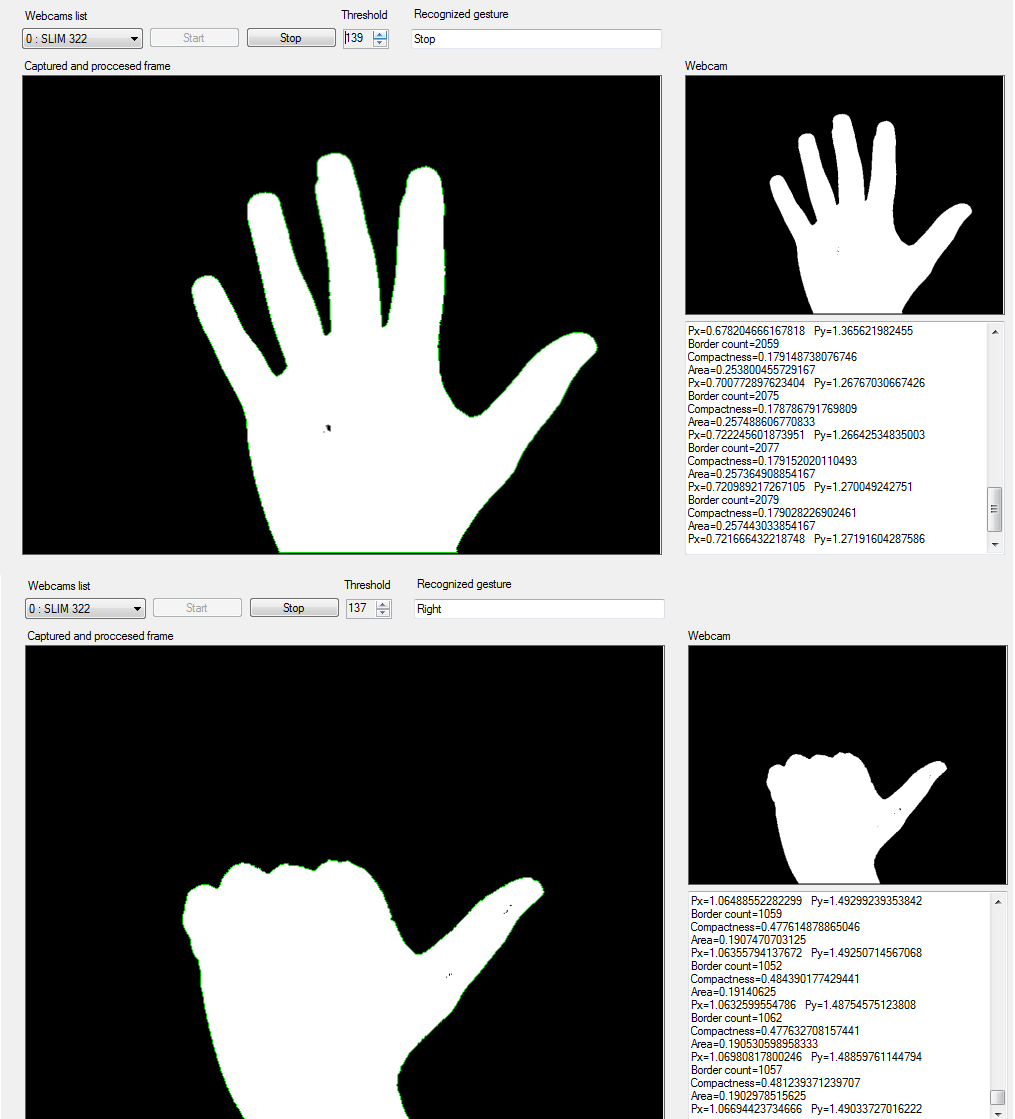


Figure – Gestures recognized by the software

# References

1. K. Kraiss, *Advanced Man-Machine Interaction Fundamentals and Implementation*, Srpinger, Berlin, 2006 available online at

http://www.springerlink.com/content/978-3-540-30618-4#section=458355&page=1&locus=0

2. <http://www.aforgenet.com>

3. Parallel Programming with Microsoft .NET http://msdn.microsoft.com/en-us/library/ff963553