# Rock, Paper & Scissors with Nao

Nimrod Raiman [0336696]
Silvia L. Pintea [6109960]

## Abstract

Throughout this paper we are going to explain our work on teaching a humanoid robot (*Nao*) to play *"Rock, paper & scissors"*. In order to accomplish this task we have used different theoretical methods which are described in the section 2. In Section 3 are presented our experimental results. Finally, we give an overall view of this paper and indicate the possible future work that could be done on this subject.

## 1 Introduction

*"Rock, Paper & Scissors"* is an easy and well known game. This is the reason for which it is interesting to learn a robot how to play it against human players. In order to do that the robot needs to be able to recognize the hands of its opponent and classify the gesture as: *"rock"*, *"paper"* or *"scissors"*.

In this paper we describe our approach to accomplish this in real time. Our solution is fairly robust to lightning condition and also the gestures of the player need not be restricted to certain angles or positions in the frame (given that the gesture is clear enough).

Our problem has been split into three main tasks: extracting the hands from the webcam stream, recognizing the gesture of the extracted hand and implementing motion and speech on *Nao*. Throughout our project we have tried different approaches in order to find the best method to solve the problem.

We have experimented with methods such as: *backprojection of probability histogram* for hand detection, *Gabor filters* and *PCA* for feature extraction. We will start by describing the methods that we have tried and then continue by giving an overview of the results and the conclusions.

## 2 Methods

For hand detection and recognition we have employed two different techniques: a naive approach and the backproject of the probability histogram of the pixels corresponding to skin.

For the gesture recognition we have experimented with different sets of data and we have extracted different features using methods such as: *PCA* and *Gabor filters*. We have also tried using two different types of classifiers: *SVM* (support vector machine) and *Knn* (K nearest neighbors).

We will continue by giving a more detailed description of the methods employed.

### 2.1 Hands extraction

#### 2.1.1 Naive approach

Our first approach for extracting the area from the webcam stream that corresponds to the hand was to define thresholds for the hue and the saturation values for the frames received from the webcam. The resulting binary image would contain all pixels corresponding to skin. Then using erosion and dilation noise was reduced. From the resulting binary image we then could easily find the area corresponding to the hand.

This naive approach was not reliable enough because it is very sensitive to objects with a color

that is close to skin color in the view of that camera. This approach is also very sensitive to lightning changes (artificial light is more blue than sunlight). Finally, the thresholds need to be defined very broad to make this work for the majority of skin colors, fact which might introduce even more false skin response.

### 2.1.2 Backprojection of skin pixels

After realizing that our first approach was too naive and instable we needed to find a more robust skin finder.

Inspired by [3] and *Gijs Molenaar* we implemented a skin *PDF* (probability distribution function) approach. In order to make such a *PDF* we needed to build a histogram (normalizing the histogram would give a *PDF*) of an area that we knew for sure it was representative for the skin color of the person standing in front of the camera. For this we chose to use the face area. Using a frontal face *Haar detector* we were able to find the face of the person standing in front of the camera and we constructed a histogram from the inside area of the rectangle returned by the face detector. We removed the borders in order to loose as much background and hair as possible and get an even more reliable skin *PDF*.

Once we had the histogram, we could backproject it on the image. The resulting image was a gray-scale one showing what areas are more or less probable to correspond to skin (see Figure 1 – left image). We first set the probability of the pixels in the face area to zero, because we did not want this to be interfering with finding the hand in the frame. Then we thresholded this image to get rid of areas with very low probability for the skin. Using rough erosion and dilation we got rid of noise and connected the areas that belonged together. This was done roughly because at this step, we just wanted to find the area of the hand and process it separately in a more sophisticated manner. From the resulting image we have selected the biggest blob of pixels that were connected and chose this as the most probable area to correspond to the hand.

Once we had the area that corresponded to the hand, we went back to the backprojection image. Here we selected the hand area again to do a more sophisticated thresholding, erosion and dilation on it. After dilation a part of the hand area would end up being white – corresponding probably to the skin, and another part being black – probably not skin. The black area was left unchanged while the white area was replaced by the original gray-scale image. The final step was to resize the hand image to $70 \times 70$ pixels to match the training images.

The resulting image was one in which the hand area was selected and most of the background was removed (see Figure 1 – right image).

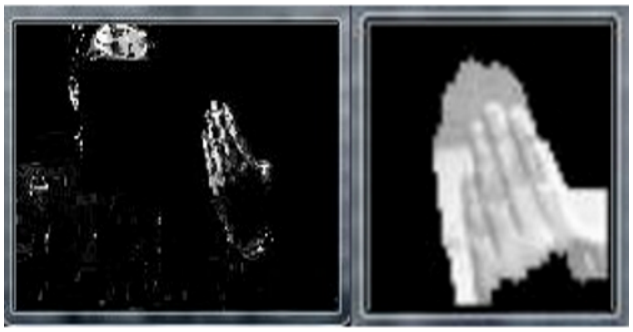

Figure 1: Hand detection – Example

## 2.2 Gesture recognition

For the gesture recognition task we have started by using a training set containing images of hands of $70 \times 70$px with different backgrounds. The problem proved to be too complex for our classifiers so we have decided to switch to a simpler one which would contain only centered hands and a black background.

Out of this dataset we have extracted the features to be used during the classification.

### 2.2.1 PCA

The first technique we have tried was *PCA*. We have computed the *eigen-hands* of our data and

then we have projected each set, separately, on this space.

There are three steps for generating the *eigen-hands*:

- Subtracting the mean of the data
- Computing the covariance of the data and the eigenvectors of the covariance matrix
- Projecting each training set (corresponding to each sign: *"rock"*, *"paper"* or *"scissors"*) on the space defined by the eigenvectors

For high-dimensional data a better approach than computing the eigenvectors of the matrix: $Data^T \times Data$ (which for a dataset of *[N,D]* with $D \gg N$ has the dimension *[D,D]*) would be to use an intermediate step and compute the eigenvectors $V \rightarrow eigh(Data \times Data^T)$ and then determine the final *eigen-space*: $U \rightarrow \frac{Data^T \times V}{norm(Data^T \times V)}$.

Unfortunately, given the fact that *PCA* is not background invariant, translation invariant or rotation invariant the results were not as good as we were expecting.

### 2.2.2 Gabor filters

A *Gabor wavelet* is created by defining the size of the wavelet to be generated and then by looping over the coordinates of each pixel ($x$ and $y$) and applying the following formula:

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\Pi \frac{x'}{\lambda} + \psi\right)$$

where $x' = x\cos\theta + y\sin\theta$

and $y' = -x\sin\theta + y\cos\theta$

The parameter $\lambda$ represents the frequency of the stripes in the wavelet, $\theta$ gives the angle of the stripes, $\psi$ is a translation parameter, $\sigma$ gives the size of the stripes while $\gamma$ indicates how elliptical they are.

Each image in the training set is convolved with the *Gabor wavelets* and then reshaped as a single row.

Figure 2: Gabor wavelet Example

In our project we have used 4 *Gabor wavelets* (with the angle $\theta \in \{0, \frac{\Pi}{4}, \frac{\Pi}{2}, \frac{3\Pi}{4}\}$). Each image was convolved with each one of these 4 wavelets and then reshaped as a row. All 4 results obtained in this manner were then concatenated and stored in single row for each image.

Besides using these 4 *Gabor wavelets* we have also tried concatenating to the result the original image also reshaped on a single row.

Unlike the *PCA*, this method gave satisfactory results, fact which indicates that it is capable of determining a set of representative features of the training set.

### 2.2.3 Classification

In order to be able to classify the gesture we have tried using both the **SVM** (support vector machine) and **Knn** (K nearest neighbor) classifiers.

Given the fact that the data is not perfectly aligned (hands have slightly different positions in the image and different angles) the problem seemed to be too hard for the **SVM**. It was not able to find a good boundary between the classes.

To ensure that our approach was correctly implemented we have created a more aligned training set (the hands had only small differences in the angles of their orientation $\in [-20, +20]$ degrees). For this task the **SVM** classifier was able to perfectly classify the data (it found a good boundary between the classes) and it gave an error of *0.0*.

Because we did not want to restrict the player to only a small amount of positions and ways in which a sign could be made we have decided not

to use **SVM**.

We have experimented also with the **Knn** classifier. This one was able to give accurate prediction even on the raw training images and thus we have decided to employ it for solving our task of gesture recognition.

### 2.3 Motion & Speech on Nao

*Nao* is a humanoid robot with a processor of 500MhZ and 3 fingers that move simultaneously.

Because of the limited computational power we had to run our software on a laptop that was connected to *Nao*. All decisions were made by the software that was running on the laptop and the resulting interaction was sent and executed by *Nao*.

Because the fingers can only be moved simultaneously we had to represent *"paper"* as hand open, *"rock"* as hand closed and *"scissors"* by closing and opening the hand a few times.

In order to make *Nao* move and talk we had to

Figure 3: Nao Aldebaran

generate behavior files using the *"Choregraphe"* simulator. These files were, then, transfered to *Nao*'s memory and then we had to remotely connect to *Nao* using its *IP* address and execute one of the predefined behaviors that was needed at that specific moment in the game.

The hand-detection and gesture recognition parts need to be run in parallel with *Nao*'s motion and speech behaviors such that it would be able vocalize the outcome of the game by comparing it's choice of move with the player's one.

## 3 Results

We have trained our models using images of hands with different positions and orientations. We have used:

- *1641* training examples for the *"rock"* sign
- *1522* training examples for the *"paper"* sign
- *1377* training examples for the *"scissors"* sign

The results are obtained using 50–fold cross validation.

In Table 1 are depicted the average errors we have obtained while testing our methods. We can notice from here that the *PCA* tends to perform worst than the other methods. It can also be observed that the *Gabor wavelets* have a high prediction accuracy and that the best results are obtained when concatenating the convolved images of *Gabor wavelets* with the original image for the size of $20 \times 20$px.

| Size | Method | Average Error |
|---|---|---|
| 70×70 | *PCA* | 0.475 |
| 20×20 | *PCA* | 0.470 |
| 20×20 | *Gabor* | 0.021 |
| 20×20 | *Gabor + PCA* | 0.510 |
| **20×20** | ***Gabor & Image*** | **0.012** |
| 20×20 | *(Gabor & Image) + PCA* | 0.447 |
| 70×70 | *Grayscale* | 0.016 |
| 20×20 | *Grayscale* | 0.014 |

Table 1: Average errors for different methods

# 4 Conclusions

Throughout this paper we have presented the methods we have employed in order to make a humanoid robot (*Nao*) play *"Rock, paper & scissors"* against a human opponent.

Our experiments indicate the fact that the **SVM** classifiers is not very well suited for solving this problem and that it is capable of finding a good boundary between the classes only in the case in which the orientation and the position of the hands does not vary too much. But on the other hand a more simple classifier like the **Knn** is capable of solving the problem and returning satisfactory results.

We have also seen that **PCA** was not able to give features strong enough to allow for a good classification performance. This might be due to the fact that **PCA** is not background invariant, rotation invariant or translation invariant.

Given the fact the the hand detection component of our project was not selecting hand as perfectly separated from the background as the ones used for training we had to enhance our training set with a number of examples given by the hand-detection component in order to ensure stability to our system and a more precise and robust prediction.

In the present our software runs at a rate of 4 frames per second. With some proper optimization techniques this can be improved to make *Nao* react even faster and more naturally.

Finally, we believe that other methods might give results as good as the ones presented here. Probably a good idea would be to try employing *SIFT* features in solving this problem.

# References


[1] Yen-Ting Chen, Kuo-Tsung Tsen, *Developing a Multiple-angle Hand Gesture Recognition System for Human Machine Interaction*; 2007

[2] M. R. Tuner, *Texture Discrimination by Gabor Functions*; 1986, Biol. Cybern. 55, p. 71-82

[3] Comaniciu, Dorin and Ramesh, Visvanathan and Meer, Peter, *Kernel-Based Object Tracking*; 2003, IEEE Trans. Pattern Anal. Mach. Intell., vol 25, p. 564–575