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 **??**. The next section presents 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.

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 pixel values for hand detection, *Gabor filters* and *PCA* for classifying signs. 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 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

Nimrod .....

2.1.2 Backprojection of skin pixels

Nimrod .....
Determine hue and saturation values for skin color
Threshold the image with hue and saturation values
Use erosion & dilation
Find ares corresponding to hands
vspace*10px Robust/sophisticated approach

Determine skin color histogram
- Detect face
- Build histogram of pixels corresponding to the face

Backproject skin color histogram on whole frame
Use erosion & dilation to reduce the noise and fill up gaps
Extract area of corresponding to the hand
Use more sophisticated erosion & dilation on hand area
- Retain the hand and remove the background
- Resize the area of interest to 70x70

2.2 Gesture recognition

For the gesture recognition task we have started by using a training set containing images of hands of *70*×*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 orientation invariant the results were not as good as we were expecting.

2.2.2 Gabor filters

$\boxed{xxxx}$

Sivlia .....

2.2.3 Classification

Silvia .....

Do we want subsub sections here?

Build reliable training set
Find useful features to train on: PCA, Gabor wavelets, grayscale images
Train a classifier: Knn /SVM
Create models & test in order to find the best one

2.3 Motion & Speech on Nao

How does Nao play Rock!Paper!Scissors!

Make Nao generate the moves for "rock", "paper" and "scissors"
I Make Nao keep the score of the game by recognizing the gestures of the other player

3 Results

Some nice results here...

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

Conclusion
What now?

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