Automated hand recognition as a human-computer interface

Sergii Shelpuk
SoftServe, Inc.
sergii.shelpuk@gmail.com

Abstract

This paper investigates applying Machine Learning to the problem of turning a regular webcam into the input device and human-computer interface by the hand recognition. The purpose of the project is to create an application for drawing and making notes on the presentation by moving the hand in front of the laptop camera. For the purpose of this task, there were created two separate datasets, applied color filtering and trained two neural network based classifiers. For the purpose of accuracy improvement, also there was introduced an algorithmic threshold based filer for the drawing.

1. Introduction

Human-computer interaction (HCI) focus is one of the most popular and promising ideas in software and hardware design of the last decade. Computer evolution and success stories (as well as reasons of failure) of numerous products were based on new types of interface and user experience that make the product convenient and easy to use for as many potential customers as possible. In the early days of personal computers, windows based interface and mouse made a revolution in PC world through making interaction with the programs more intuitive than the console. Today, we are witnessing another HCI-enabled revolution: smartphones and tablets. Removal of the mouse and keyboard and ability to access the software directly with the fingers make these devices such user friendly that even children who do not have the access to the education who and have never seen any computer before can use tablets to learn how to read and write by themselves[1]. The importance of this cannot be overestimated.

The key factor of success in HCI is making the experience of software product as close as possible to the experience of the real world interaction. Considering the examples above, visual interface and mouse are much more alike to the real world than the console. However, interacting with the software using our own fingers is even better since almost every interaction of a human being with the real world is of tactile (kinesthetic) nature.

This concept can be extended to the other types of interfaces as well.

Today, webcam is an integral part of almost any computing device. Laptops, tablets, smartphones, TV theaters, great number of IP phones, telepresence devices, ATM and even some cars have integrated webcam often with the purpose of capturing and transmitting someone's face during the conversation. However, from the HCI point of view, their potential seems to be greatly underestimated.

This project is devoted to the development of a new type of HCI allowing the presenter to make notes and draw on the presentation using just his/her fingers with no additional device by hand recognition with a webcam.

2. Data Set

Two separate datasets were created for the purpose of this project: hand image dataset and finger figure dataset.

Hand image dataset was obtained by capturing the picture from the camera and breaking it down into 100x100 pixel squares. After that, the pictures with the hand were separated manually, filtered and reduced to the 50x50 pixels size.

Finger figure dataset was obtained by breaking down 100x100 pictures with the hand into

[1] MIT Technology Review: Given Tablets but No Teachers, Ethiopian Children Teach Themselves

40x40 squares. After that, the pictures with finger figures were separated manually.

2.1 Hand image dataset

This dataset consists of 9404 images of 50x50 pixel size. The pictures are divided into two types: 2724 pictures with human hand (positive examples) and 6680 pictures without human hand (negative examples).

Filtering

Regular RGB color image contains information about pixel colors that is redundant for the purpose of this project – higher number of features requires more data to train the accurate classifier. Converting the image into grayscale does not fully address the problem: grey intensity of every pixel varies with the light condition. Additionally, the grayscale image still contains the background that also affects the classifier performance.

To address this problem the training set pictures were processed with the skin detection filtering.

The goal is to transform an RGB color image in the form that contains only evidences of the hand but not the background or any other redundant information. The transformation also should provide this result independent of light condition and white balance of the original picture.

The filter uses the fact that in standard grayscale image the pixel with intensity "0" corresponds to the black color, the pixel with intensity "255" corresponds to the white color and all the values between them corresponds to the shades. For the filtering, RGB value of the fingertips is used as a reference color. Each pixel of the image is transformed according to the formula[2]:

$$C_{filtered} = 255\left(1 - e^{-\frac{(C_{pixel} - C_{ref})^T (C_{pixel} - C_{ref})}{\sigma^2}}\right)$$

$C_{pixel} \in R^3$ is a vector of RGB values of the original pixel.

$C_{ref} \in R^3$ is a vector of reference color (the color of the fingertips in the current light condition).

$C_{filtered} \in R$ is a grayscale value of the pixel after transformation.

$\sigma^2 \in R$ is a variance constant.

Intuition behind this filtering is the following: after the transformation, grayscale picture has only the figures that are similar in color to the fingertips. Everything else is white.

(a) (b) (c)

Figure 1: Original image (a), greyscale image (b) without filtering (still contain a lot of redundant information) and filtered image (c) – positive example from the hand image dataset

(a) (b) (c)

Figure 2: Original image (a), greyscale image (b) without filtering and filtered image (c) – negative example from the hand image dataset

This allows to use the product with the different light conditions by changing value of the C_{ref} .

2.2 Finger figure dataset

This dataset consists of 7547 images of 40x40 pixel size. The pictures are divided into two types: 2273 pictures with finger figure (positive examples) and 5247 pictures without finger figure (negative example).

[2] According to the class restrictions, the filtering was written from scratch

No filtering was applied for this data set because grayscale image gives the easily recognized pattern.

(a) (b)

Figure 2: Positive (a) and negative (b) examples from the finger figure dataset

3. System Design

The system overall consists of two architectural components Octave scripts and Python program.

Octave scripts functionality:

1. importing pictures and transforming them into training set, cross validation set and test set
2. training classifiers
3. exporting weights in CSV format

Python program functionality:

1. loading weights from CSV files
2. capturing video from camera
3. performing skin detection filtering
4. detecting hand on the screen
5. detecting finger figure
6. transforming coordinates of the finger figure to the screen coordinates for the mouse position
7. performing threshold based filtering
8. drawing a figure based on the coordinates of the finger figure

OpenCV 2.1 library was used for the purpose of video capturing, resizing and transforming into grayscale[3]. Numpy 1.6.1 library was used for linear algebra calculations in Python.

3.1 Classifiers

The system needs to find the finger figure on the image taken from the webcam. For this purpose, Python program captures the image from the webcam, reduces its size from 640x480 to 320x240 for the reason of performance, performs filtering and breaks it into 50x50 squares and tests using two classifiers. First one is looking for a hand on the picture (hand image classifier).

If it returns "true" (the image contains hand) then the program transforms the coordinates of the positive square to the coordinates of the original 640x480 image, takes 100x100 square with the hand, transforms it to the greyscale and breaks into 40x40 squares.

Each of the 40x40 square is tested with the second classifier. If it returns "true" (the image contains finger figure) then the program transforms the coordinate of the square to the original coordinates and considers them as a drawing point.

For the purpose of this application, both classifiers should perform on two classes that are non-linearly separable. Two possible algorithms performing well on non-linear classification were considered for the purpose of this project: SVM with polynomial kernel and Neural Networks (NN).

The table below represents best achieved performance of two algorithms on the datasets:

	Hand image dataset	Finger figure dataset
SVM	88,01%	92,4%
NN	95,12%	97,15%

So, NN classifiers were selected for the application. Backpropagation was used as a training method.

Also, regularization was used to avoid overfitting. Regularization parameter was selected based on the cross-validation set performance.

3.2 Hand image classifier

Architecture of the hand image classifier and regularization parameter was selected based on the classifier performance on the hand image dataset.

The table below represents the accuracy of the three layer NN architecture given regularization parameter λ. NN architecture is represented as following: Input layer size/hidden layer

[3] According to the class restrictions, both neural network (NN) classifiers were written from scratch, OpenCV internal NN functionality was not used for the purpose of this project

size/output layer size. Some combinations of architecture/regularization parameter was not tested because they are clearly supposed to give the performance lower than already found maximum.

λ	2500/ 5/ 2	2500/ 30/ 2	2500/ 100/ 2	2500/ 200/ 2	2500/ 300/ 2
0	0,814	-	0,902	0,94	0,93
0,01	-	0,84	-	-	-
0,05	-	0,850	-	-	-
0,1	0,829	0,861	-	-	-
1	-	0,871	-	**0,9512**	0,950
2	0,817	-	-	-	-
5	0,84	0,888	0,939	0,949	-
10	0,850	0,890	0,946	-	0,949
15	0,850	-	-	-	-
20	0,829	0,871	0,913	0,942	0,946
30	0,829	0,84	0,891	-	-
40	-	0,813	-	0,936	0,94

Four layer NN architectures were also tested but they did not show any significant improvement of the performance. So three layer neural network with architecture 2500/200/2 and regularization parameter $\lambda = 1$ was chosen for the hand image classifier.

The performance of the classifier is specified in the table below:

Test set size	1800
Accuracy	0.951
True positive	536
False positive	31
True negative	1252
False negative	61
Precision	0.945
Recall	0.897
F_1 score	0.91

For the purpose of this project we really care about precision since false positives affects performance of the system overall. From this point of view, the performance of the classifier is acceptable.

3.3 Finger figure classifier

For the finger figure classifier, the same analysis was performed to define the better architecture and the better regularization parameter. The results of the analysis are presented in the table below.

λ	2500/ 30/ 2	2500/ 100/ 2	2500/ 200/ 2	2500/ 300/ 2
0	0,89	0,927	0,968	0.962
0,5	0,883	-	**0,9715**	0.962
1	-	0,932	0.97	0.964
2	-	-	-	0.965
5	0,83	-	0,961	0.962
7	-	0,912	-	0.98

So three layer neural network with architecture 2500/200/2 and regularization parameter $\lambda = 0.5$ was chosen for the finger figure image classifier.

The performance of the classifier is specified in the table below:

Test set size	1509
Accuracy	0.9715
True positive	421
False positive	12
True negative	1045
False negative	31
Precision	0.9722
Recall	0.9314
F_1 score	0.951

As well as for hand image classifier, precision is a key performance indicator so this classifier is acceptable as well.

3.4 Threshold based filtering

The main problem with the application is false positives: every false positive not only consumes CPU resources to evaluate the picture in details looking for finger figure, but also in case of false positive of the finger figure classifier, it produces a false drawing point. Despite pretty high level of accuracy, this happens and still makes the method inapplicable.

To address this issue, additional filtering method was introduced. Each drawing point is represented by two coordinates X and Y. We

make assumption that since the user moves the hand smoothly, so two sequential drawing points cannot be far away from each other. If the distance of the next drawing point found and previous drawing point exceeds some threshold d_{tr} then the new drawing point is likely to be a false positive and should be excluded from the drawing point lists for the current line.

After the filtering, the drawing point is connected with the previous drawing point by emulating mouse movement with the left button pressed.

The figure below represents the example of using the application to make some note on the PowerPoint presentation.

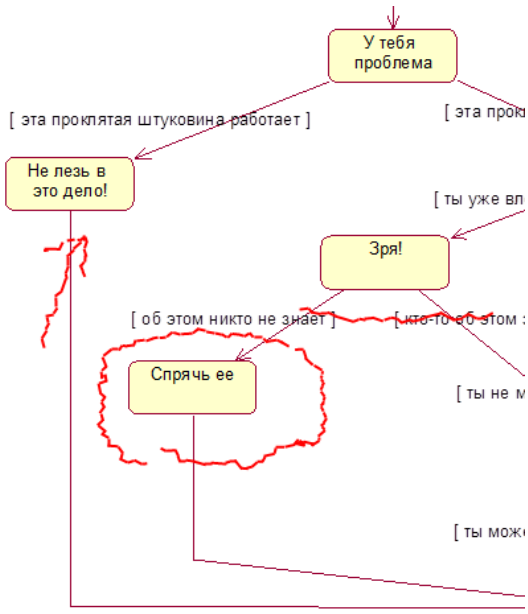

Figure 3. Drawing on the presentation with the developed application

4. Conclusion and future work

Clearly, the quality of the drawing on the current stage is not enough to call the application a completed product. However, it shows that Machine Learning can turn regular webcam into interface for human computer interaction.

The project has a lot of opportunities for improvement and future work:

- Deep learning techniques and Dropout method can be applied to improve the performance of the classifiers.
- Using the fact that if a 100x100 square contains a hand, then the shifted for =/- 10 pixels square should also contain the hand, the additional classification can be introduces to improve the accuracy of classification
- Currently, the application uses straight lines to connect drawing points. This results in quite ugly lines. Drawing points can be divided into groups of three and each group can be approximated by the second order polynomial. This will make the drawing smooth.
- Additional algorithm could be introduced to predict where the finger figure most likely will be in the next timeframe (next screen capture) based on the previous movements. This can reduce the CPU work and the amount of pictures to classify thus reducing the false positive chance.

References

Robert L. Harvey, Paul N. DiCaprio, and Karl G. Heinemann A Neural Network Architecture for General Image Recognition. The Lincoln Laboratory Journal Volume 4. Number 2. 1991 (189-207)

C.-C. Yang, S.O. Prasher, J.-A. Landry, H.S. Ramaswamy and A. Ditommaso. Application of artificial neural networks in image recognition and classification of crop and weeds. Canadian Agricultural Engineering Vol. 42, No. 3 July/August/September 2000 (147-152)

Christopher M. Bishop. Neural Networks for Pattern Recognition Oxford University Press, Nov. 23 1995