Dynamic Gesture Recognition for Gaming Interface

C Govindrajan
Student of Computer Engg.DY Patil College Of Engg.
Brahma Emerald County off Nibm Road, Pune

Deepak P U
Student of Computer Engg.DY Patil College Of Engg.
Lakshmi Palace, Akurdi, Pune

AkshayThosar
Student of Computer Engg.DY Patil College Of Engg.
Lakshmi Palace, Akurdi, Pune

ABSTRACT
This paper presents the seed idea of how we can replace the traditional inputs for the games with new gestures. For this purpose we integrated the idea of image processing and the neural network. We took the example of simple game of tetris to visualize the system. Neural Network will lead us to more efficient and rapid handling as compared to currently available aids. With the simple human gesture i.e. hand movements people will able to enjoy the game playing experience this concept can be simply adaptable to not only any games but also to many other application where the tradition input gesture can be replaced with the modern one.

General Terms
Algorithms, Performance, Experimentation, Verification.

Keywords
Neural Network, Dynamic Gesture Recognition, Game Playing, Learning methods..

1. INTRODUCTION
Advancement in the gaming industry has forced us to think out of the box to always come up with new and innovative ideas. There has been substantial development in the gaming consoles and has led us to a great revolution in this field. Keeping the same trend we want to introduce the new concept of game playing in which the tradition input i.e. the inputs from the keyboards or mouse will be replaced with the human gestures giving the game playing a new dimension.
Hence users are allowed for richer and user-friendlier man-machine interaction. This can lead to new interfaces that will allow the deployment of new commands that are not possible with the current input devices. Plenty of time will be saved as well.Recently, there has been a surge in interest in recognizing human hand gestures. Hand gesture recognition has various applications like computer games, machinery control (e.g. crane), and mouse replacement. Computer recognition of hand gestures may provide a more natural-computer interface [8].

Hand gestures can be classified in two categories: static and dynamic. A static gesture is a particular hand configuration and pose, represented by a single image. A dynamic gesture is a moving gesture, represented by a sequence of images. We will focus on the recognitionof dynamic images.

The application of this idea is not only limited to the field of game playing but also it can be made adaptive to any other application. There has been going lots of research work in this field even the likes of Intel, Microsoft are working with this new idea, however they are limited in their ideas so far. But they also believe that this is the new idea that will push the development in a new dimension. So far there is no new way of gesture recognition in the game that is played on the pc platform. That’s what we are going to present in this paper.

2. CURRENT APPROCHES AND SOLUTION
Research on hand gestures can be classified into three categories. The first category, glove based analysis, employs sensors (mechanical or optical) attached to a glove that transducers finger flexions into electrical signals for determining the hand posture. The relative position of the hand is determined by an additional sensor. This sensor isnormally a magnetic or an acoustic sensor attached to the glove. For some data glove applications, look-up table software toolkits are provided with the glove to be used forHand posture recognition [14].

The second category, vision based analysis, is based on the way human beings perceive information about their surroundings, yet it is probably the most difficult to implement in a satisfactory way. Several different approaches have been tested so far. One is to build a three-dimensional model of the human hand. These parameters are then used to perform gesture classification. A hand gesture analysis system based on a three-dimensional hand skeleton model with 27 degrees of freedom was developed by *Lee and Kunii*.

The third category, analysis of drawing gestures, usually involves the use of a stylus as an input device. Analysis of drawing gestures can also lead to recognition of written text.

We will focus on how the gesture recognition can be modularized using the skin detection algorithm and how neural network will help us to manage the environment for which gesture are produced. Environment is referred by us to any way application on which our concept is applicable say it be any game or any other application [11].

3. SYSTEM DESIGN
Now let’s focus on how we can achieve our goal with introduction to various modules of our system.

We will divide our system in three parts

1. Gaming Module
2. Gesture Recognition Module
3. Neural Network Module

3.1 Gaming Module
Here we refer to our module as gaming module but it can be any other application to which our concept is being implemented. We can take any game, but the thing to remember is that more complex the game is more kinds of gesture need to handle. We will take the simple game of tetris where we have to move the block of different sizes in a rectangular play area. If we are able to make a complete line with the help of these blocks we score points. As time passes the difficulty of the game increases i.e. the speed of the incoming blocks increases.

In these we have deal with four gestures

1. Move the block to left
2. Move the block to right
3. Move the block to downwards
4. Rotate the blocks.

We can have various other gestures depending of our games like if the take the example of complex games like Call Of Duty we will have gesture for shooting with the gun and rotating our viewpoints, movements etc.

3.2 Gesture Recognition Module

We now have to design a method that will be able to serve the mode for the gesture recognition of the player. For this purpose we will need some hardware support like in our case as we are only developing gesture for tetris games. For our case it is best suited that we use the some kind of detection algorithm but however this gesture recognition algorithm actually depends on our application.

Skin Detection Algorithm –
Calibration - Calibration of the input device is done by taking various samples of the different skin colors and locations. A more general approach (more skin colors) needs a more complex model that will be certainlymore computational demanding. Several studies already showed, that the majordifference in the skin color appearance lies, not in color itself but in the intensity. Departing from this argument we studied and tested a new scalingmethod for the testing pixels based on the calibration values that improved theoverall results. Using a simple normalization for the saturation value in [0.; 1.0](equation 1) range, and a scaling for the tint value obtained by eq. 2, based onthe saturation calibration minimal value, we obtained good results as shown insection 7, for white, black skin and descents, only using one skin chrominancemodel.

$$S = (S - MinS)/(MaxS - MinS) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$T = (T - MinS)/(MaxT - MinS) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Filtering - In any image with a complex background, the skin area is often smaller thanthe non skin pixels. Since we use a pixel evaluation procedure, to reduce thecomputation effort, we apply a initial filter that can remove all the pixels easilylabeled as non skin.This filter only aims to remove pixels from further processing.Skin segmentation- From previous work we can observe that the TSL chrominance spaceis very effective for skin segmentation when using a Gaussian model. This is also
true where illumination conditions vary. Using the following transformations theequivalent pixel representation is obtained on the ST color space [17].

Threshold - The threshold method, always select part of an image to the maximumvalue 1 and probably, if similar pixels are present, values very close to 1 willalso be present. This can be seen as a solution to maximize the difference amongcandidates or a problem if no skin is present, since at least a non skin pixel willmatch 100% as being skin, in result of the normalization. In our point of view,and also from direct experiment, images where skins do not exist do not have alarge group of pixels similar to skin, and normally they are isolated. In this waythe grouping method described in section 6, can manage this problem. Also, ifskin is present, is expected that the $\lambda 2i,j$will be smaller for skin pixels. Even ifskin is very different from the calibration data used, $\lambda 2i,j$is expected to be smallerthan those that are not skin, so the skin pixels will also tend to approximate thevalue 1[3].

Grouping - The resulting image from the previous section can be enhanced, also if we arelooking for skin regions, a grouping method should be used. After the imagethreshold we apply a median filter to smooth the image, and to reduce the smallamounts of isolated noise resulting from the method descried in section 4. Onlysmall areas are modified and almost no skin pixels are erased.Using a connect pixels analysis; we obtain groups **G** that are initially filteredby their size, if they present a small number of pixels in relation to the imagesize. We also connect from the initial universe **G**, two or more close groups toreduce some common errors (as example, two fingers separated). This was doneby finding the smaller distance between groups that must be proportional to thesize of one of the elements.Using the image size, we do not need a human interaction, and tests showedthe good results for images from size $320x200$ up to $1600x1200$. we observed that although separated, the skin elements aregrouped and are returned as a single one.This grouping method is useful for face detection and hand detection. Closeelements from distinct objects are also returned connected if the grouping evaluationis satisfied, so the grouping should always be performed with caution [4].

3.3 Neural Network Module

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements[19].

Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown in fig (1). There, the network is adjusted,based on a comparison of the output and the target, until the network output matches the target.

Typically many such input/target pairs are used, in this *supervised learning* (training method studied in more detail on following chapter), to train a network.

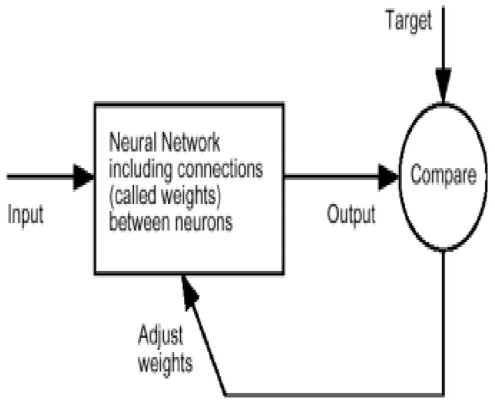
Target
Neural Network including connections (called weights) between neurons
Input
Output
Compare
Adjust weights

Fig 1: Neural Network Block Diagram

Neural networks have been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech, and vision andcontrol systems.

Today neural networks can be trained to solve problems that are difficult for conventionalComputers or human beings. The supervised training methods are commonly used, butother networks can be obtained from *unsupervised training* techniques or from direct *design* methods

Unsupervised networks can be used, for instance, to identify groups of data.

Now as we know something about neural network we can easily teach them and make it work as per the need of us.

In our case of game playing we can make it learn that there could be any of the following cases

1. If the skin detection algorithm shows that there is continuous increase in the x co ordinate then we have to perform MOVE1.

2. If there is continues decrease in x co ordinate value then we have perform MOVE2.

3. If there is continues decrease in y co ordinate value then we have to perform MOVE3.

4. If there is continues increase in y co ordinate value then we have to perform MOVE4.

5. All other case can be neglected.

MOVE1- move the block in right direction.

MOVE2- move the block in left direction.

MOVE3- move the block in downward direction.

MOVE4- rotates the block in anticlockwise direction.

For the general case we can implement the modified version of perception algorithm described as below. For the development of the error-correction learning algorithm for a single-layer perceptron, we will work with the signal-flow graph shown in fig (2). In this case the threshold $\theta(n)$ is treated as a synaptic weight connected to a fixed input equal to –1.

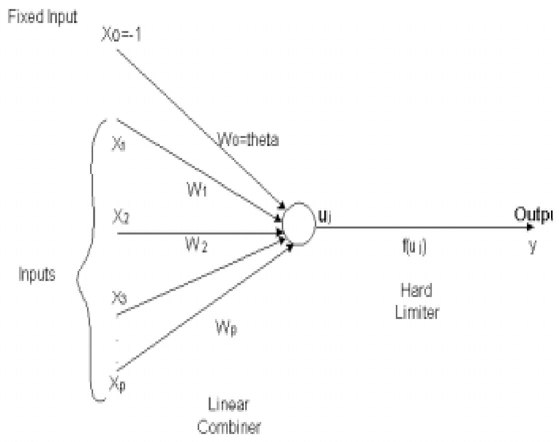

Fig 2: Perceptron Signal Flow Graph

We may then define the (p+1)-by-1 input vector[5]:
$x(n) = [-1, x_1(n), x_2(n), \ldots\ldots xp(n)]^T$
Correspondingly we define the (p+1)-by-1 weight vector:
$w(n) [\ (n), w_1(n), w_2(n), \ldots\ldots wp(n)]^T$
Below are some variable and parameters used in the convergence algorithm

$\boldsymbol{\theta}(n)$ =threshold
$y(n)$ = actual response
$d(n)$ = desired response
η = learning rate parameter, $0 < \eta < 1$
So let's see the 4-step algorithm in greater detail:
Step 1: Initialization
Set w (0) =**0.** Then perform the following computations for time $n=1, 2, \ldots\ldots$

Step 2: Activation
At time n, activate the perceptron by applying continuous-valued input vector $x(n)$ anddesired response $d(n)$.

Step 3: Computation of Actual Response
Compute the actual response of the perceptron:
$y(n) = \text{sgn}[wT(n)x(n)]$
The linear output is written in the form:
$u(n) = wT(n)x(n)$
where: $sgn(u) = +1$ *if* $u>0$
$sgn(u) = -1$ *if* $u<0$

Step 4: Adaptation of Weight Vector
$w(n+1) = w(n) + \eta[d(n) - y(n)]x(n)$
where$d(n) = +1$ *if* $x(n)$ *belongs to class* 1 *C*
where$d(n) = -1$ *if* $x(n)$ *belongs to class* 2 *C*
Step 5Increment time *n* by one unit and go back to step 2

This can adapted as per the requirement as the structure of neural network will also change as per the application requirement.

4. RESULT

I did perform a rough test on the prototype and the following result has been obtained for different level of frame rate –

Table 1.Percentage accuracy

Frame rate	Accuracy	Comments
10 fps	73 %	The main problem was inadequate inputs after filtering. User generally needs to input long gesture for recognition.
15 fps	91 %	A fairly acceptable accuracy figure given the amount of noise, input filtering does not filter much data because of the larger frame rate.
20 fps	81 %	At higher frame rates, more and more small sized vectors are created, meaning more filtering. Main source of error here is inadequate inputs after noise filtering.
25 fps	77 %	Again higher frame rates results in high input filtering, main source of error is again inadequate inputs resulting in no classification.

5. CONCLUSION

In this paper we have given the idea how the gesture recognition can be done using the neural network for games like Tetris and also we have given the basic introduction to neural network which can be implemented in various applications. We will try to implement our concept as mentioned in the paper.

Although a c/c++ implantation will be the fastest way to go but this will lead to a huge complexity in the design which is not best suited and would be preferred to implement in MATLAB. There are fields on which we want to work on which will make our concept more reliable and fast.

The initial test that has been run has shown an optimum performance of 91 %,it does come fairly close to the standard which is 97 % (market standard [1]) for complete user interface satisfaction. I would like to push this performance more to achieve the criteria.

6. ACKNOWLEDGMENT

The authors wish to thank KlimisSymeonidis for his valuable ideas during the prototype development .We also would like to thank the experts who have contributed towards development of the concept.

I also would like to thank my college which has helped us for this paper.

7. REFERENCES

[1] Araokar, S. “Visual character recognition using artificial neural networks”, pages 1-7.MGM’s College of Engineering and Technology, 2005

[2] William T. Freeman, Michael Roth, “Orientation Histograms for Hand Gesture Recognition” IEEE Intl. Wkshp. On Automatic Face and Gesture Recognition, Zürich, and June, 1995.

[3] Vladimir I. Pavlovic, Rajeev Sharma, Thomas S Huang, “Visual Interpretation of Hand Gestures for Human-Computer Interaction: A review” IEEE Transactions of pattern analysis and machine intelligence, Vol 19, NO 7, July 1997.

[4] SrinivasGutta, Ibraham F. Imam, Harry Wechsler, “ Hand gesture Recognition Using Ensembles of Radial Basis Functions (RBF) Networks and Decision Trees” International Journal of Pattern Recognition and Artificial Intelligence, Vol 11 No.6 1997.

[5] Simon Haykin, “Neural Networks, A comprehensive Foundation”, Prentice Hall

[6] Duane Hanselman, Bruce Littlefield, “Mastering MATLAB, A comprehensive tutorial and reference”, Prentice Hall.

[7] http://www.cs.cmu.edu/afs/cs/project/cil/ftp/html/vision.html

[8] http://www.merl.com/people/freeman/

[9] Improved Automatic Skin Detection in Color ,Images By Filipe Tomaz and Tiago Candeias and Hamid

[10] http://vismod.www.media.mit.edu/vismod/classe/mas622/projects/hands/games

[11] T.W. Yoo, and I. S. Oh, “A fast algorithm for trackinghuman faces based on chromatic histograms”, Pattern Recognition Letters 20(10): 967-978, 1999.

[12] Skin Detection using Neighborhood Information by Javier Ruiz-del-Solar and Rodrigo VerschaeMariaPetrou, PanagiotaBosdogianni,“Image Processing, The Fundamentals”, Wiley

[13] http://www.hav.com/

[14] http://www.neural-forecasting.com/

[15] http://www.tk.uni-linz.ac.at/~schaber/ogr.html

[16] Christopher M. Bishop, “Neural networks for Pattern Recognition” Oxford, 1995.

[17] http://aiintelligence.com/aii-info/techs/nn.htm#What

[18] http://www.cs.rug.nl/~peterkr/FACE/face.html

[19] http://www.dacs.dtic.mil/techs/neural/neural_ToC.html

[20] Walt Scacchi, *Research and Educational Innovations in Computer Game*s, pages 2-4California Institute for Telecommunications and Information Technology, 2002