

---

# Chessman Position Recognition Using Artificial Neural Networks

---

**Jun Hou**

JUNHOU@CAIP.RUTGERS.EDU

Center for Advanced Information Processing, Rutgers University, Piscataway, NJ 08854

## Abstract

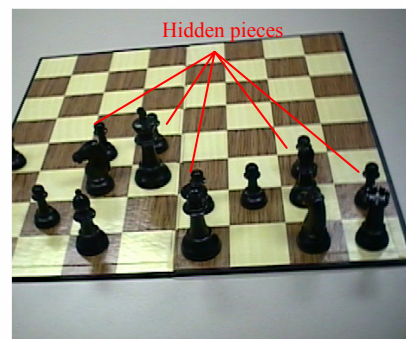
In the augmented reality chess game, a human user plays chess with a virtual user. To do real-time registration, the positions of the black chess pieces have to be found whenever the real world user makes a move. A Feed Forward Artificial Neural Network is used to recognize the chess piece positions. For training, images are acquired by synthesizing the chessboard and the pieces from different perspectives for different chess piece positions. To normalize the chessboard image with different perspectives, the chessboard corners are found and each chessboard square is divided into 8\*8 smaller squares. Several variations of Gradient Descent algorithms in Back Propagation are examined. Although there is still much work to be done, the ANN approach has revealed its convenience and performance in this task.

## 1. Introduction

In an augmented reality chess game, two remote users play chess over a standard IP network (Ganapathy, et al 2003). One is a novice who plays real black pieces in the real world, and the other is an expert who plays virtual white pieces on the computer screen. At the novice side, he only manipulates black pieces, but he can see the expert's virtual white chess pieces superimposed on his chessboard through the Head Mounted Display (HMD), while the expert can see the novice's chess pieces on his computer screen. There is an intelligent agent sitting between them providing help. In order for the agent and expert to know the novice's move, we need real time registration that recognizes the positions of all the chess pieces. For human eyes the problem is easy, but for computers it is another story. Through the HMD, sometimes the black pieces are gathered into several big clusters, making it difficult to see which piece is on which square, even for human eyes in certain cases. Please refer to Fig. 1 for an example.

Artificial Neural Networks, which simulate the analog processing of human brain cells, have been applied successfully in the field of machine vision (Littmann, et al 1992). Used in various environments, ANNs compliment the more traditional approaches in the case when the precise theory and algorithm of the problem is hard and

costly to formulate, e.g. pattern recognition of constantly changing elements that demand complicated algorithms. The CMU ALVINN (Baluja, 1996) is a successful example of ANN application in which a car is driven based on visual inputs.



*Fig.1* Illustration of clusters

Feature selection is important in neural network design, since the input data complexity determines the network structure. In doing this, 2D features, 3D features, and combinations of 2D and 3D features can be used (Alleazard, et al 2000). ICA (Independent Component Analysis) can provide a method for feature extraction. Hoyer & Hyvarinen (2000) apply the ICA method and decompose images into linear combinations of "basis" patches. When the input image data has large dimension, PCA (Principal Component Analysis) is usually used to reduce the input space. For example, Huang, et al (2001) use a polynomial neural network with input node features extracted using PCA when recognizing human faces.

In the project depicted above, the camera is attached to the human head that might move at any time, thus the 3D video image changes dramatically. Also, for a novice player who has little knowledge in chess, the black chess pieces can be moved anywhere, thus producing an unbounded number of patterns. In order to know the position of each chess piece, I used a Feed Forward Neural Network trained using traditional Gradient Descent, GD with a variable learning rate, and GD with momentum. The neural network is trained with synthesized clean images and tested both on the training set and the separate test set. Variations of the neural network training are also compared and analyzed.

The structure of the paper is organized as follows: Section 2 introduces the feature selection and data preprocessing

of this project, Section 3 is the neural network design, and Section 4 presents the experimental results. At last is the conclusion and future work.

## 2. Feature Selection and Data Preprocessing

In this project, since we only need to identify the real world user's real black pieces, RGB images are converted to intensity images whose features are enough for training. Normally, we begin playing chess by starting a new game, so the initial state of the chessboard is known. Also, only one piece is moved each time, except for castling. The number of chess pieces on the board is also known given the chessboard position. This reduces a great amount of complexity. As such, we don't need to know whether several pieces are moved simultaneously to different positions and which piece moved where. So the problem can be decomposed in this way: determine whether there is a chess piece on a certain square. Then this problem has a binary output for each square.

Traditional neural networks require a fixed number of inputs. But for a moving camera, the perspective is always changing, and the region of interest – the chessboard – is not a square in the 2D image. In order to get a fixed number of input nodes, the image of each chessboard square is divided into  $m*m$  smaller squares, and the average of each square is calculated. Because of the perspective of the chessboard, the black pieces that are closer to the camera occupy larger areas in 2D. Since the black pixel position and black ratio have effect on the result, thus each square is divided evenly in 3D and then mapped onto 2D. In order to do this, we first detect the corners of the chessboard in 2D – please refer to (Ganapathy 2003) as how to do this – then calculate the 3D corner positions using the camera projection matrix. Knowing the chessboard corners, the chessboard squares can be divided evenly in 3D, and then mapped to 2D.

For different camera viewing angles, the black portions of chess pieces are different. Even for the same angle the gray level will still change for different positions of the chessboard square. As training the neural network for each square based on all camera angles and positions is prohibitive (there are hundreds of them), the row and column of the square, the camera position and rotation are included as input nodes, to compensate for the black ratio difference for different perspectives. Besides, given a set of labeled chessboard images, the prior probability of the pieces on each square can be calculated. This is also provided as an input feature.

To reduce the number of input samples, if the average of gray scale is less than a certain threshold, then the data of that square will not be used for training. We are only interested in the square that contains at least a portion of black.

## 3. Neural Network Design

A feed forward neural network with one hidden layer is applied here. The input has four parts:  $m*m$  image data, the square position to be examined, the camera variables, and the prior probability that this square is occupied. Please refer to Fig.2 for an illustration.

The number of nodes in the hidden layers represents the characteristics of the problem. Since there are 6 types of chess pieces, each having a different shape, the number of nodes of the hidden layer could be 6 or even more. For the sigmoid unit,  $\tanh(x)$  function is used.

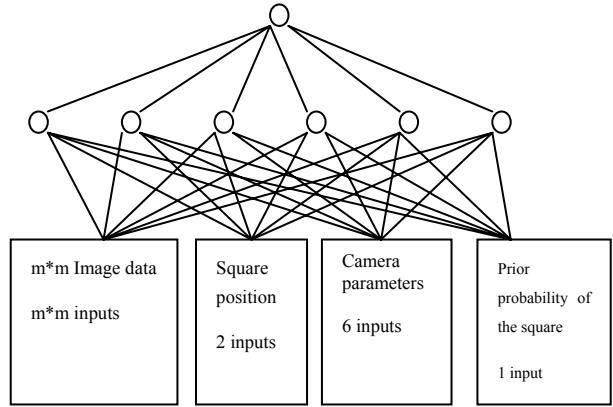


Fig.2 Neural Network Design

During training of the neural network weights, the traditional Gradient Descent (GD) algorithm, gradient descent with momentum, and a variable learning-rate algorithm are compared here. Using momentum, the update of the weights can be written as:

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1) \quad (1)$$

where  $\alpha$  is a constant that is called momentum.

GD with variable learning rate is denoted as

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} \quad (2)$$

where  $\eta$  is not a constant. It is calculated so that whenever the new error rate exceeds a predefined ratio, then the new weights and biases are discarded; otherwise, the learning rate increases by a certain amount.

For the performance metrics of this project, if the output is greater than 0.5, then it is considered as 1, otherwise as 0. The error rate is calculated as the ratio of wrongly classified squares vs. the total numbers of testing squares.

## 4. Experimental Results and Discussion

### 4.1 Data Generation

At the beginning of this stage, we only use synthesized data. Synthesized images are clean and can be labeled when generating them, thus reducing manual work. Also, we want to test the algorithm on clean images first. If the result is successful, we can move on to real video data.

Originally, the black pieces are at their beginning positions. Because the real world user is a novice player, he can make illegal moves. To simulate this, a random move is played on the current chessboard to get a new chessboard state. Then the camera position and rotation are changed to look at the chessboard from different angles. The range of the camera position is limited to one side of the chessboard where the user sits. The azimuth of his head is almost always in 90 degrees of range; the elevation is in always 10 to 20 degrees of range. The azimuths chosen are between  $-45^\circ$  and  $45^\circ$ , with step  $15^\circ$ . The elevations are  $30^\circ$  and  $40^\circ$  respectively. The camera is always looking at the center of the chessboard.

We use Blender 2.30 to model the pieces and write Python scripts to manipulate the pieces and generate the images automatically. The images are clean, with high contrast, little noise, and without lighting variations (constant ambient lighting from all directions). The image format is PNG type, which has very good compression ratio and yet with less distortion than JPEG images. The camera used is ideal with no distortion, and its field of view is  $37.85^\circ$  and focal length 68.54mm.

We generated 1500 images with  $640 \times 480$  resolution in this way.

#### 4.2 Data Filtering

Using the data filtering schemes described in the previous sections, each chessboard square is partitioned into  $8 \times 8$  smaller squares, thus producing 64 input units for each image. When filtering the data in the grayscale range of 0 and 1, a threshold of 0.13 is done to select the small black regions and eliminate the effect of yellow and brown squares. Another threshold of 0.1 is done to select the useful squares with some black portion on it. An example of the original data and the processed data is shown in Fig.3.

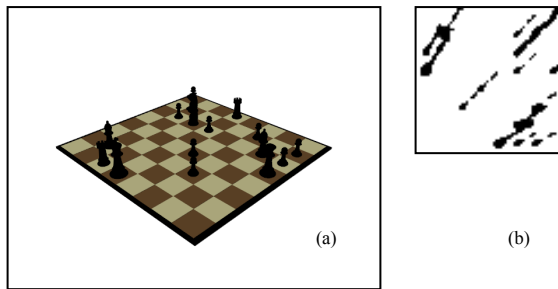


Fig.3 (a) Original synthesized  $640 \times 480$  3D image and (b) converted  $64 \times 64$  2D image. Only none white squares are used for training. Trying to restore the 3D image to 2D image. The chessboard is as if rotated  $45^\circ$ .

#### 4.3 Training

The training is done all in batch mode for traditional GD, GD with momentum and GD with variable learning rate.

1000 of the 1500 chessboard images are used for training, and the other 500 images are used for testing.

To know how many training iterations are necessary, we first examine the effect of training epochs, although this number may vary due to the different sizes of training samples. Fig.4 is a result based on GD with a variable learning-rate using 100 chessboard images for training. The reason this number is small can be that a single square is selected for one input sample. And a chessboard image contains normally more than 25 samples.

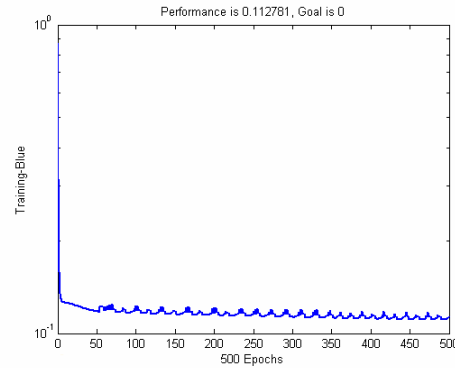


Fig.4 Effect of epochs of training. Based on 500 training chessboards

In Fig. 5, the performance difference between the training set testing and separate test set testing is shown, both trained on 1000 samples using GD with variable learning rate. From this figure, we know that the training set accuracy drops when the number of training samples gradually grows large; while the test set accuracy gradually grows, until at last they merge to about 72%, if there are enough training samples. This is because when the training set is small, the ANN weights are adjusted only to the small data, thus recognizing very well. And because it is not trained for other possible situations, the accuracy for the test set is low. As the size of training samples grows larger, the test set result becomes better. This is because the more training data, the more similar are the training and the test sets.

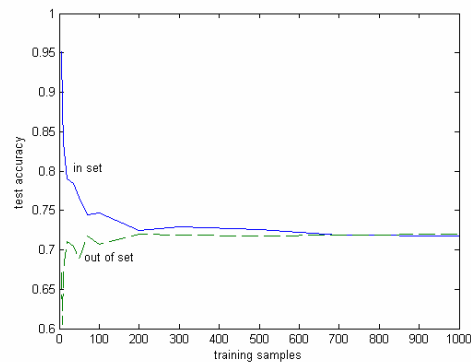


Fig.5 Training set test accuracy and test set accuracy

Fig. 6 presents the training result based on the same data set as in Fig.4, when using GD with momentum. Comparing Fig. 6 and Fig.4, we find that the performances are similar, although the learning curves are different. These two algorithms are quite similar in that they both try to adjust the difference in two successive loops. And GD with variable learning rate is more likely to oscillate.

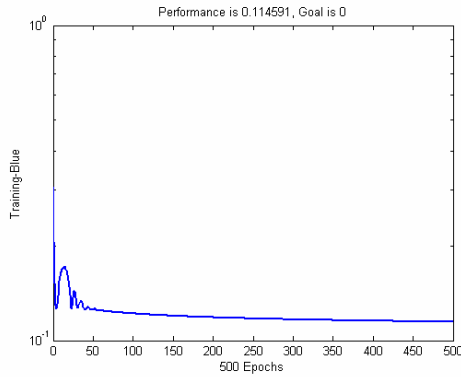


Fig.6 Training using GD with momentum

The test set performance comparison of GD, GD with variable learning rate, GD with momentum are similar, all achieving about 72% accuracy, with little difference. The speed of training time is shown in Table 1. GD with momentum is a little faster than the GD and GD-VLR.

Table 1. Training time (in seconds) differences between GD, GD-M, and GD-VLR. Training based on 500 iterations.

	GD	GD-M	GD-VLR
500 IMAGES	76.43	79.79	76.37
1000 IMAGES	138.3	145.5	139.3

## 5. Conclusion and Future Work

In this paper, feed forward neural network is applied to help recognize the positions of the chess pieces. The neural network is trained on synthesized image data. We devised a way to normalize the image to adapt to the requirements of the ANN. The variations of the gradient descent algorithms are also compared, and we found that both the GD with momentum and GD with variable learning rate are faster than the traditional GD.

Due to time limit, many ANN training parameters have not been adjusted yet, like the number of division in a square when normalizing the data, the effect of randomizing the training data, the increasing and decreasing values used in the variable learning rate algorithm, the choice of good momentum, etc. This could be done in future work.

Also, rules can be used in this task to help recognize the chess piece positions. For example, we can use the difference between the output score and desired output score, which are 0.9 and 0.1 respectively, as a confidence measure. If the confidence is too low, then we use rules to help, e.g. knowing the total number of pieces on the chessboard, we select the squares classified as “1” that have the highest score (highest confidence); knowing there is only one piece moved, we can look at the successive results and find the most likely piece move. Also, when the human even has difficulty to see the piece positions clearly, he will naturally turn his head and look at the chessboard from another angle. So multiple frames of the same chessboard can be used together to decide the piece positions jointly. Perhaps stochastic constraint satisfaction or belief propagation can be applied here also. These tasks will be done in the following weeks.

## References

- Allezard, N., Dhome, M., & Jurie. F. (2000). *Recognition of 3D Textured Objects by Mixing View-Based and Model-Based Representations*. International Conference on Pattern Recognition (ICPR'00)-Volume 1. September 03 - 08, 2000. Barcelona, Spain, pp. 1960-1963.
- Baluja, S. (1996). *Evolution of an artificial neural network based autonomous land vehicle controller*. IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 26, No. 3, June, 1996, pp. 450 - 463.
- Ganapathy, S. K., Morde, A. and Agudelo, A. *Tele-Collaboration in Parallel Worlds*. ACM SIGMM Workshop on Experiential Telepresence. Berkeley, CA, Nov. 7, 2003.
- Hoyer, P. O. & Hyvarinen, A. (2000). *Independent Component Analysis Applied to Feature Extraction from Color and Stereo Images*. Network: Computation in Neural Systems, vol. 11, no. 3, pp. 191--210, 2000.
- Huang, L., Shimizu, A, Hagihara, Y., & Kobatake, H. (2001). *Face Detection From Cluttered Images Using a Polynomial Neural Network*. In the Proceedings of IEEE International Conference on Image Processing, 2001.
- Littmann, E., Meyering, A., and Ritter, H. *Cascaded and Parallel Neural Network Architectures for Machine Vision – A Case Study*. In Proc. 14. DAGM\_Symposium 1992, Dresden, S. Fuchs(ed.), Springer, Heidelberg, Berlin, New York, 1992.