Augmented Reality using a Neural Network

Pye Phyo Maung
Department of Mathematics and Computer Science
Ripon College
Ripon, Wisconsin 54971
maungp@ripon.edu

March 9, 2012

Abstract

In this paper, we present a simple augmented reality system using a neural network inspired by the biological vision system. It uses a feature-based recognition system developed in the C# programming environment. It uses OpenCV for feature detection and extraction, a self-organizing map (SOM) for picture recognition, and Microsoft WPF 3D for rendering 3D objects on the screen. This paper describes the characteristics of augmented reality system. For image recognition, the global representation of images are used as image features. We explore different global features that can be used to improve the recognition and discuss how the self-organizing map (SOM) is trained using these global features using an unsupervised learning algorithm. The SOM is a two-layer neural network comprising the input layer and the competition layer that is a grid of neurons. Each neuron is represented by a vector that has the same dimension as an input feature vector. During the training, the SOM groups the similar pictures together in the competition layer and chooses a neuron for each group.

We describe the process of building an augmented reality system using the self-organizing map. Using a webcam as a visual sensor, the system is presented with a sequence of input images that are processed frame by frame. On receiving an input image, the system detects a rectangle of interest in the input image by analyzing contours from the edges extracted with an edge detection algorithm in OpenCV. From the picture within the rectangle of interest, a feature vector is extracted and later fed into the neural network for picture recognition. Picture recognition is done by choosing the best representative neuron that represents a group of pictures in the competition layer, and then choosing the most similar picture in the group. Once a picture is recognized, a 3D object corresponding to the picture is rendered on the screen using WPF 3D.

1 Introduction

Recent advances in computer processing capability, new 3D technologies, and efficient image processing technologies have opened the door for real-time systems with intensive computing such as the augmented reality system, which needs to process enormous amount of input data from a visual sensor and give responsive user interaction. Research in augmented reality systems has been focusing on improving the technologies in such application domains as medical operations, defense, education, robotics, manufacturing, maintainence, assisted driving, mobile assistance, and entertainment.

1.1 Augmented Reality

Augmented reality is generally the enhanced view of a real and physical environment in which virtual elements are blended with the real elements. According to [1], Augmented Reality is usually defined as the view of the real world that has the following three characteristics:

Augmented Reality

1. blends real and virtual elements, in the real environment,
2. is Real-time interactive, and
3. is Registered in 3-D, that is, the computer generated virtual objects are fused with the three-dimensional real world.

In terms of Milgram's taxonomy of mixed reality visual displays[2], the concept of augmented reality falls between the reality and the virtuality. It is closer to the reality since virtual elements are added to the real environment. In contrast, the augmented virtuality, which is also a type of mixed reality similar to augmented reality, is closer to the virtual reality since users completely immerse in the simulated environments. Milgram defines the relationship among the reality, the virtuality and the mixed reality as the Reality-Virtuality Continuum as shown in Fig. 1.

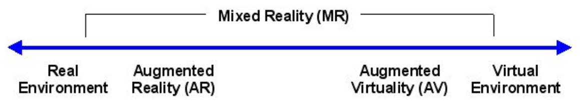

Figure 1: Milgram's Reality-Virtuality Continuum

1.2 Vision Recognition

At the heart of an augmented reality system is a vision recognition system and a 3D rendering engine since the ultimate goal of the augmented reality is to blend the virtual elements wth the real elements. In our project, the vision recognition system is inspired by the human vision recognition system. A great deal of research has been going on in computer vision in order to mimic how a human being's brain processes visual information and recognizes patterns. Some of the operations such as pattern recognition that our brain can do so well seem very difficult for a computer. Studies in psychology suggest that when the brain processes visual information, it divides a visual scene into sub-dimensions, such as color, movement, form and depth and delegates them to different areas of the brain. The holy grail of vision has been how the brain integrates these sub-dimensions into the perceived image[8]. To illustrate this, imagine seeing a scene that includes a bird. The visual information of that bird is processed to mark areas of interest and is divided into sub-dimensions such as color, depth and form, which are then processed in different areas of the brain. Then, for each sub-dimension, the neurons that are associated with corresponding memory will fire. Using the collection of these represented neurons, the brain tries to generate its own image of a bird, after which we perceive it as a bird. As we explore our environment with our eyes, enormous amount of information streams into different parts of the brain through our visual sensory system. The brain's visual centers are constantly reorganizing themselves to adapt to this new information. We demonstrate this idea using a feature-based vision recognition system that consists of a type of neural network known as "self-organizing map" [3] and a collection of global features such as color, texture and edges.

1.3 Overview of an Augmented Reality System

Inspired by the biological vision system, we build a simple augmented reality system using a neural network. Our system receives inputs from a visual sensor such as a web-cam and divides them into sub-dimensions such as color, texture and edges. The features of these sub-dimensions are fed into the neural networks. The neuron that fires in the output layer of the neural network determines which pattern is recalled from the training set.

Our system uses the combination of global features to compare image similarity. Before running the augmented reality system, the neural network is trained using a training set of images. The type of neural network we are using is called a self-organizing map (SOM). SOM is a two-layer neural network comprising the input layer and the competition layer, which is a grid of neurons. SOM is trained using an unsupervised learning algorithm, in which the training examples in the training set are unlabeled, that is, the input training examples are not given the corresponding outputs. In other words, unsupervised learning can be considered as finding patterns in the data with unstructured noise. Each neuron is represented by a vector that has the same dimension as an input feature vector. During the training, the SOM groups the similar pictures together in the competition layer and chooses a neuron for each group.

After the training, using a webcam as a visual sensor, the system is presented with a sequence of input images that are processed frame by frame. On receiving an input image, the system detects a rectangle of interest in the input image by analyzing contours created from the edges in the image. These edges are extracted by using edge detection algorithms in OpenCV. From the rectangle of interest, a feature vector is extracted, and later fed into the neural network for recognition. Pattern recognition is done by choosing the best rep-

resentative neuron that represents a group of pictures in the competition layer, and then choosing the most similar picture in the group. Once a picture is recognized, a 3D object corresponding to the picture is rendered on the screen using WPF 3D.
The process of our augmented system can be illustrated as a pipeline as shown in Fig. 2.

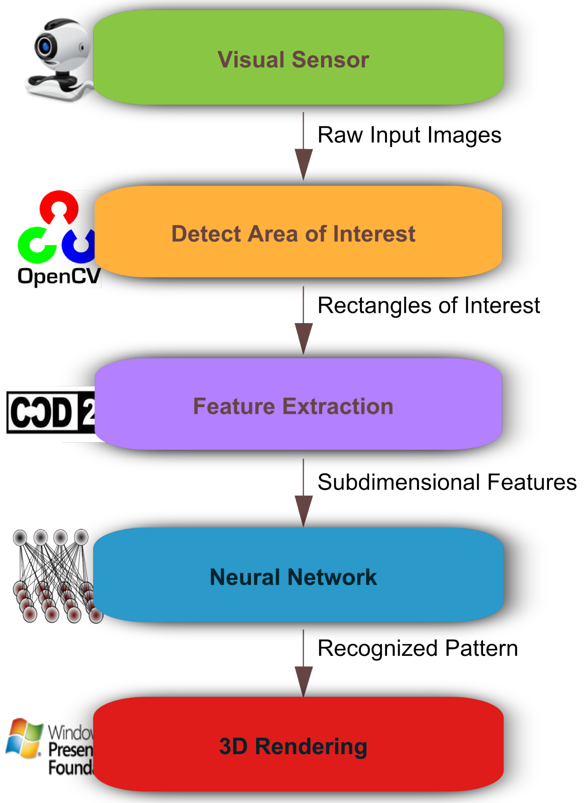

Figure 2: A Pipeline for Augmented Reality System

2 Feature Detection and Extraction

2.1 Rectangle of Interest

Since our project uses the feature-based technique for pattern recognition, features need to be detected and extracted from input images. In order to extract features and later feed them into the neural network, an area of interest in an image needs to be defined. Our system defines the area of interest as rectangles, which can be referred to as rectangles of interest. Our system detects rectangles of interest in the images by using Canny edge detector[9] and contour extraction algorithms using OpenCV. Canny edge detector is a edge detection technique which uses a multi-stage algorithm to detect a wide range of edges in images. It is used to extract contours in the images and examines the angles, sides, and vertices made by contours to detect the rectangles of interest.[7]. This detection process of the rectangle of interest is illustrated in Fig. 3

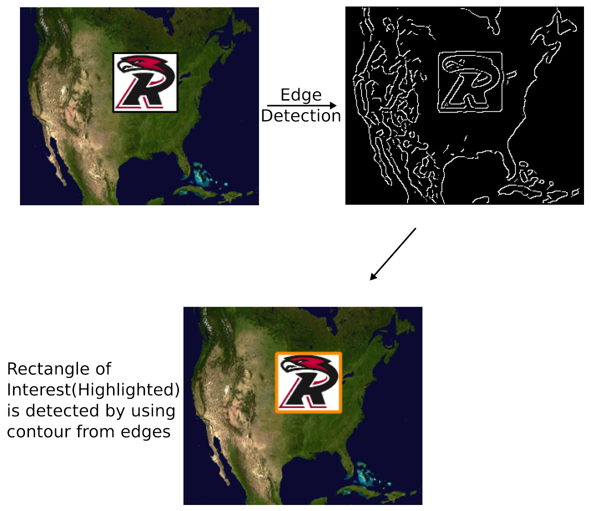

Figure 3: An illustration of edge detection and rectangle of interest detection

2.2 Feature Extraction

Choosing the right features is the most important part of the feature-based vision recognition system. However, for simplicity, we first experimented with simpler global features such as intensity histograms and color areas. These are usually widely used as global features in a traditional image retrieval system. Histograms are simply collected counts of the underlying data organized into a set of predefined bins. They can be populated by counts of features computed from the data, such as gradient magnitudes and directions, color, or just about any other characteristic. This gives a statistical picture of the underlying distribution of data.[7] An intensity histogram is a graph that shows the counts of pixels in a gray scale image at each different intensity value. Since different images have different intensity histograms, it can be used as a sub-dimension for pattern recognition. An example of a histogram is illustrated in the Fig. 4

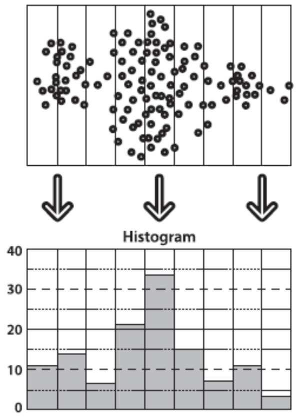

Figure 4: An Example of Histogram [7]

Since intensity histograms work on gray scale images, the input RGB images are temporarily converted to gray scale images. The intensity histograms for each image are obtained using built-in functions in OpenCV. The data type for each intensity histogram is defined as a float array of size 256, in which the values are distributed so that the intensity of pixels increases with the index of the array, i.e., index 0 has the lowest intensity and index 255 has the highest intensity. The resulting float array is to be fed into the neural network for pattern recognition. Similarity between intensity histograms of two images are measured by the Euclidean distance.

After experimenting with simpler features, we use a more complex feature called the Joint Composite Descriptor, which is mentioned in [15], which combines the Color and Edge Directivity Descriptor(CEDD)[11] and the Fuzzy Color and Texture Histogram(FCTH)[10], proposed by Chatzichristofi and Boutalis. We find that the Joint Composite Descriptor works significantly better than the intensity histogram and color areas. We briefly summarize the extraction of these features.

2.2.1 Color and Edge Directivity Descriptor (CEDD)

The Color and Edge Directivity Descriptor (CEDD) is extracted as defined in [11]. CEDD is the extraction of a low level feature that combines color and texture directivity into one histogram. In CEDD feature extraction, the image is separated in a preset number of blocks. Then, the color information based on HSV color space is extracted using a set of fuzzy rules. This generates a 10-bin quantized histogram, each of which corresponds to a preset color. The number of blocks assigned to each bin is stored in a feature vector. Then, an extra set of fuzzy rules is applied to a two input fuzzy system, in order to change the 10-bins histogram into a 24-bins histogram, thus importing information related to the hue of each color that is presented. Next, 5 digital filters are also used for exporting the information related to the texture of the image by classifying each image block in one or more of the 6 texture regions that has been fixed, thus shaping the 144-bins histogram. Then, this histogram is quantized so that the value of each bin falls within the range of 0 and 7. The final result of CEDD feature extraction is a 144-dimentional vector that gives information about color and edge directivity of an image.

2.2.2 Fuzzy Color and Texture Histogram (FCTH)

The Fuzzy Color and Texture Histogram (FCTH) is extracted as defined in [10]. The extraction of the FCTH feature is very similar to that of CEDD. FCTH is also the extraction of a low level feature that combine color and texture. Like CEDD, a 24-bins histogram is calculated by using a set of fuzzy rules to the image in HSP color space, which follows the same part of feature extraction in CEDD. Then, each image block is transformed with Harr Wavelet transform and a set of texture elements are exported. Using these elements, the 24-bins histogram is converted into a 192-bins histogram, importing texture information in the feature. Similar to CEDD feature, this 192-bins histogram is quantized so that the value of each bin falls within the range of 0 and 7. Hence, the final result of FCTH feature extraction is a 192-dimensional vector that gives information about color and texture of an image. Results of image retrieval using these features works very well for natural color images, and the result can be found in [15].

2.2.3 Joint Composite Descriptor (JCD)

The two descriptors CEDD and FCTH come from similar methods and so they can be combined to become a single histogram. The combination of these two descriptors is the Joint Composite Descriptor (JCD). The combination is done using the method described in [15] by combining the texture information carried by each descriptor, giving a 168-dimentional vector. The extraction of the Joint Composite Descriptor is illustrated in Fig. 5. This feature is used as a pattern and later is sent to the neural network for training and recognition.

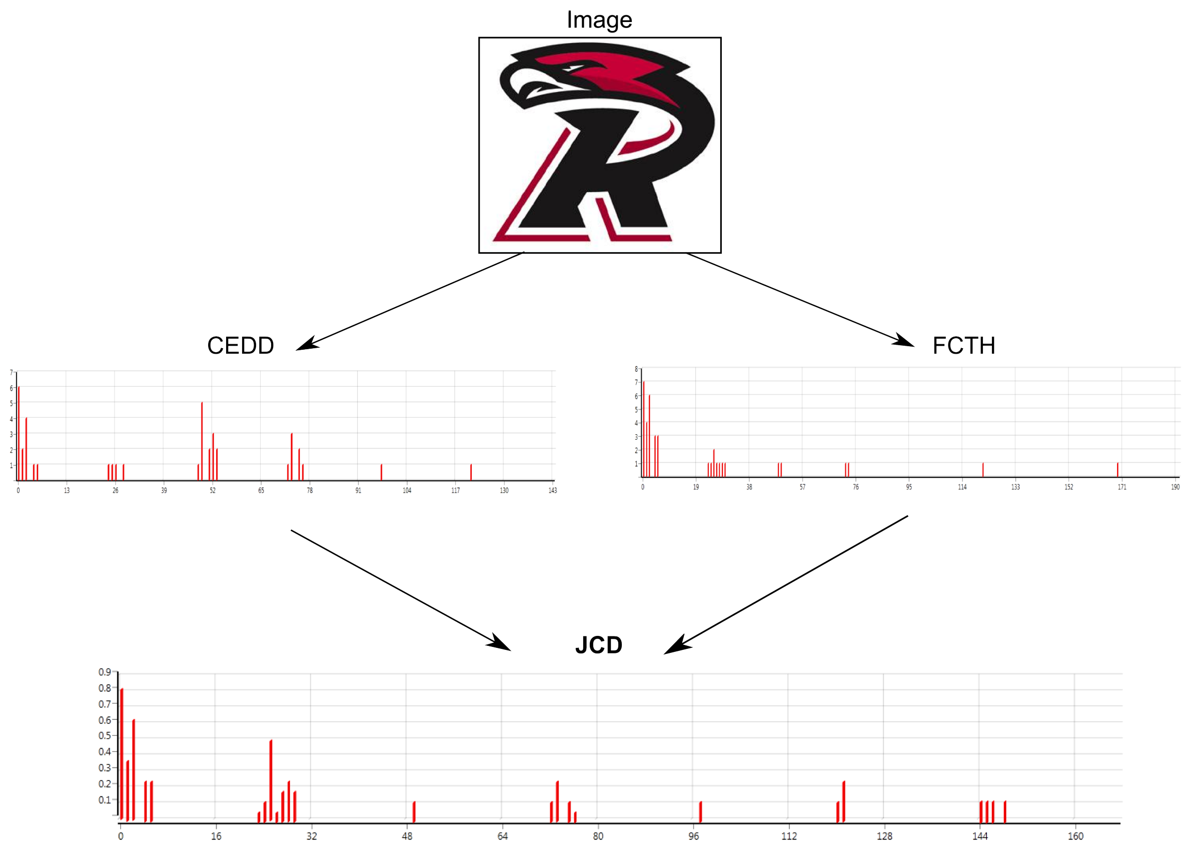

Figure 5: An illustration of feature extraction using Joint Composite Descriptor

2.2.4 Similarity Metric for Joint Composite Descriptor

The similarity between the images is calculated using the non-binary Tanimoto Coefficient [15]:

$$T(A, B) = \frac{A \cdot B}{\|A\|^2 + \|B\|^2 - A \cdot B} \tag{1}$$

where A and B are the feature vectors and $A \cdot B$ is the dot product of A and B. The value of the Tanimoto Coefficient is equal to 1 for two identical images while this value is zero in the maximum deviation.

3 Neural Network (Self-organizing map)

A self-organizing map (SOM), also known as Kohonen Neural Network, is a type of neural network which was invented by Teuvo Kohonen[3], orginally used for data visualization and classification. It reduces higher dimensional data into lower dimensions, usually one or two dimensions. And it stores information in such a way that the topological relationship between the training examples is preserved. This process of reducing the dimensionality of vectors is called ”vector quantization”, which is one of the techniques used for data compression[5]. For our project, we use the self-organizing map to recognize images.

A self-organizing map can be considered as a two-layered neural network in which the first layer is the input layer and the second layer is the competition layer in which neurons are competing with one another to become the best matching unit or ”winning neuron” for each of the similar input neurons. These ”winning” neurons represent a group in the training set that is presented to the SOM. One interesting characteristic of the self-organizing map is the use of an unsupervised learning algorithm, in which learning is done without specifying the correct outputs that corresponds to the input training examples, unlike supervised learning algorithms such as backpropagation where the training set consists of pairs an input and a target or the correct answer. The architecture of our SOM is shown in Fig. 6

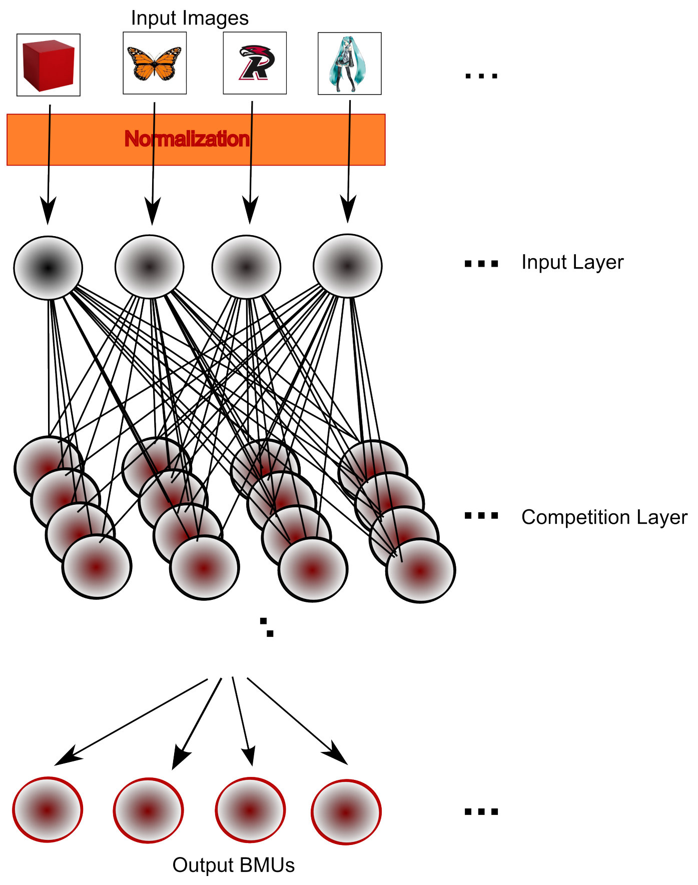

Figure 6: Architecture of SOM

3.1 Training the Neural Network (Unsupervised Learning)

Upon creation, our SOM creates a grid of neurons in the competition layer whose width and height are determined by the number of training examples in the training set. Each neuron in the input layer is connected to every other neuron in the competition layer, but no two of the neurons in competition are connected to each other. The weights of each neuron in the competition layer are represented by a vector whose dimension is the same as the input vector or weights. The SOM is trained using the algorthm described in [5]:

1. The training examples in the training set are normalized so that they are between 0 and 1.

2. The weights of the neurons in the competition layer are initialized with random values ranging between 0 and 1.

3. In each iteration, a random training example is selected.

4. For the selected training example, the best matching unit (BMU) is calculated by examining all the neurons in the competition layer for the neuron with its weight that is

the most similar to the feature of the selected training example. The similarity comparison is done by using a metric function, Tanimoto Coefficient for Joint Composite Descriptor.

5. The radius of neighbourhood of BMU, as illustrated in Fig. 7, is calculated. At the beginning of the training, the radius starts as a large value but diminishes over time steps.

6. The amount of influence on the neighbourhood by BMU is calculated.

7. Given a time step, the learning rate of the neural network is calculated.

8. Using the radius of neighbourhood, the amount of influence and the learning rate, the weights of neurons within the neighbourhood are updated so that they are more like the input training example. The closer a neuron is to the BMU, the more its weights get modified.

9. Step 2 to 7 is iterated over several time steps or iterations.

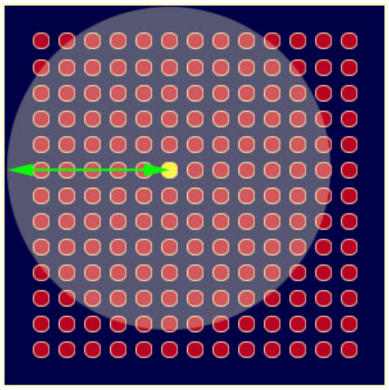

Figure 7: BMU's Neighbourhood [5]

3.1.1 Normalization of Training Set

In the input layer, the raw input vectors are normalized so that the weights are within the ranges of 0 and 1. In our study, we learn that normalization of the training examples greatly improves the precision in pattern recognition.

3.1.2 Initialization of Weights

Before training the SOM, the weights of neurons in the competition layer must be initialized. The network initializes the weights in each of the neurons of the layer by random values uniformly distributed between 0 and 1.

3.1.3 Finding Best Matching Unit

Since SOM uses unsupervised learning, it does not need a target output to be specified. Instead, the network learns each training example by selecting the best matching unit for it in the competition layer. To find the BMU for each training example, our system use the Tanimoto Coefficient between the training example and the neurons in the competition layer.

Then, the selected area around BMU is optimized to more closely resemble the training example by updating the weights of the BMU and its effective surrounding neurons. This learning is done to all the training examples over many iterations making the SOM to settle eventually into a map of stable zones.

3.1.4 Finding Radius of Neighbourhood

After the BMU has been determined in each iteration, BMU's local neighbourhood is calculated by using the exponential decay function:

$$\sigma(t) = \sigma_0 \ \exp(-\frac{t}{\lambda}) \tag{2}$$

where $\sigma(t)$ is the radius of neighbourhood given time t, σ_0 is the initial radius, usually the size of the neuron lattice and λ is a time constant calculated by the following equation:

$$\lambda = \frac{\text{Current time step}}{\log(\text{Neuron lattice size})} \tag{3}$$

During the training, this radius of neighbourhood shrinks, as illustrated in Fig. 8, over time to the size of just one neuron, which will be the final BMU. After the radius of neighbourhood has been determined the weights of neurons in that radius are updated so that these neurons will react more strongly to the input the next time it is encountered. As different neurons win for different patterns, their ability to recognize particular pattern will increase.

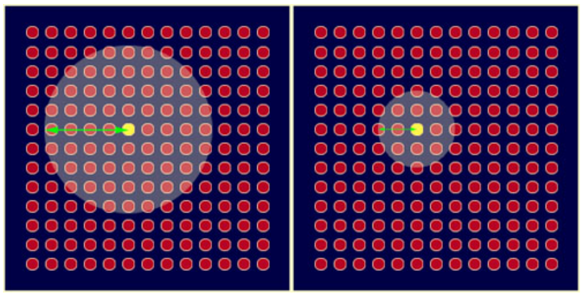

Figure 8: BMU's Shrinking Neighbourhood [5]

3.1.5 Calculating the Amount of Influence

After finding the radius of neighbourhood, the amount of influence is calculated to be used in updating the weights of the neurons within the radius of neighbourhood. The amount of influence on the neighbourhood by BMU can be given by the following equation:

$$\Theta(t) = \exp(-\frac{d^2}{2\sigma^2(t)}) \tag{4}$$

where $\Theta(t)$ is the amount of influence at time t, d is the distance between BMU and a neuron in the neighborhood, and $\sigma(t)$ is the radius of neighborhood at time t.

3.1.6 Learning Rate

The learning rate of the SOM is initially set with a constant between 0 and 1. It then gradually decreases over time so that during the last few iterations it is close to zero. The decrease of the learning rate is given by the following exponential decay function:

$$L(t) = L_0 \exp(-\frac{t}{\lambda}) \tag{5}$$

where $L(t)$ is the learning rate at given time t, L_0 is the initial learning rate and λ is the time constant.

3.1.7 Updating the Neurons

Using the radius of neighborhood and the influence, BMU and every neuron in BMU's neighborhood get their weights updated using the following equation:

$$W(t+1) = W(t) + L(t)\,\Theta(t)\,(V(t) - W(t)) \tag{6}$$

where W is the weight of a neuron, L is the learning rate, Θ is the amount of influence, and V is the input matrix.

3.1.8 Pattern Recognition

After training the SOM over several time steps when the error rate is acceptable, SOM is ready to be used for pattern recognition. This is done by simply finding the BMU for the input image in the competition layer using the same corresponding similarity metric used in training.

4 3D Rendering

The goal of an augmented reality system is to render 3D model objects in the real environment. The 3D rendering is done using Microsoft Windows Presentaton Foundation (WPF)

3D engine. One of the challenges in using WPF for rendering 3D objects on 2D screen is the conversion of three-dimensional coordinates to two-dimensional coordinates. WPF defines 3D models in three-dimensional space using three-dimensional coordinates and the perspective view of the camera, we cannot render these 3D objects without converting the three-dimensional coordinate to two-dimensional coordinate in order for them to overlay these models on the video from the camera.

In addition, 3D models rendered on the screen is determined by the result given by the recognition process discussed in the previous session. To accomplish this, the system keeps a database of 3D models, each of which is labeled with the desired patterns. Therefore, the system will render the 3D models corresponding to the result of the recognition process.

5 Conclusion and Future Work

In this study, we build a biologically inspired augmented reality system using neural network (self-organizing map). Since the system uses a set of global features for images, the computation is cheap and fast. Our system could maintain the average frame rate of about 20 frames per second during the rendering process. We tested our system with a set of test images and found that it worked well under certain conditions in which the environment allows a clear and bright view of the images without much exposure and color distortion. We find that the precision of recognizing picture suffers from some noise in the variables of the environment such as lighting and the angle of view. One promising improvement would be cooperating local features such as Speeded-Up Robust Features (SURF)[16] into the recognition process. The local features are able to select interest poins at distinctive locations in the image, such as corners, blobs and T-junctions. They allow object-detection in the image, which is not available in global features.

Acknowledge

I would like to thank all the professors from Department of Mathematics and Computer Science at Ripon College for supporting me with this project. Special thanks to Professor Kristine Peters for support and mentoring.

References

[1] Schmalstieg, David. ”Introduction to Augmented Reality.” Web. November 30, 2011. `www.iswc.ethz.ch/events/tutorials/slides_schmalstieg.pdf`

[2] Milgram, P. and F. Kishino. ”A Taxonomy of Mixed Reality Visual Displays.” IEICE Transactions on Information Systems E77-D (12): 1321-1329.

[3] Kohonen, T., 1990. ”The Self-organizing Map.” Proceedings of the IEEE 78, 1464-1480.

[4] Heaton, Jeff. ”Introduction to Neural Networks for C#.” Second Edition. Heaton Research, 2008. Print.

[5] Buckland, Mat. AI-Junkie.com. ”Self Orgainizing Maps - Part One”. Web. 22 Nov 2011. ⟨ `http://www.ai-junkie.com/ann/som/som1.html` ⟩

[6] Chesnut, Casey. ”Self-organizing Map AI for Pictures.” Web. 22 Nov 2011. ⟨ `http://www.generation5.org/content/2004/aisompic.asp` ⟩

[7] Bradski, Gary, Adrian Kaehler. ” Learning OpenCV.” O'Reilly Media Inc., 2008. Print.

[8] Myers, David G. ”Psychology.” Ninth Edition. New York: Worth Publishers, 2010. Print.

[9] Canny, J. ”A Computational Approach To Edge Detection.” IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679-698 1986.

[10] S. A. Chatzichristofis and Y. S. Boutalis, FCTH: FUZZY COLOR AND TEXTURE HISTOGRAM- A LOW LEVEL FEATURE FOR ACCURATE IMAGE RETRIEVAL, 9th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), IEEE Computer Society, pp.191-196 , May 7 to 9, 2008, Klagenfurt, Austria.

[11] S. A. Chatzichristofis and Y. S. Boutalis, CEDD: COLOR AND EDGE DIRECTIVITY DESCRIPTOR A COMPACT DESCRIPTOR FOR IMAGE INDEXING AND RETRIEVAL., 6th International Conference in advanced research on Computer Vision Systems (ICVS), Lecture Notes in Computer Science (LNCS), pp.312-322, May 12 to 15, 2008, Santorini, Greece.

[12] Chi, Z., Yan, H. and Pham, T. ” Fuzzy Algorithms: With Applications to image processing and pattern recognition, Advane in Fuzzy Systems” Applications of theory, Volume 10, World Scientific, 1996.

[13] A. Chatzichristofis, Y. S. Boutalis and M. Lux, SELECTION OF THE PROPER COMPACT COMPOSIDE DESCRIPTOR FOR IMPROVING CONTENT BASED IMAGE RETRIEVAL., The Sixth IASTED International Conference on Signal Processing, Pattern Recognition and Applications (SPPRA), ACTA PRESS, pp.134-140, February 17 to 19, 2009, Innsbruck, Austria.

[14] S. A. Chatzichristofis, K. Zagoris, Y. S. Boutalis and N. Papamarkos, ACCURATE IMAGE RETRIEVAL BASED ON COMPACT COMPOSITE DESCRIPTORS AND RELEVANCE FEEDBACK INFORMATION, International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), Volume 24, Number 2 / February, 2010, pp. 207-244, World Scientific.

[15] S. A. Chatzichristofis. ”Compact Composite Descriptors.” 2011. Web. February 18, 2012
⟨ `http://chatzichristofis.info/?page_id=15` ⟩

[16] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool ”SURF: Speeded Up Robust Features.” Computer Vision and Image Understanding (CUIU), 110(3):346-359, 2008. ⟨ `http://www.vision.ee.ethz.ch/~surf/` ⟩