3D City Model Generator: The Application of Neuro-Fuzzy Systems in CAD

Yoshihiro Kobayashi

Arizona State University

Abstract

This paper introduces a computer-aided design (CAD) system in which a neuro-fuzzy system is integrated as a main engine for learning. Specifically, a computer system that generates 3D city models from satellite images is formulated, implemented, and tested. Techniques from neural networks, fuzzy systems, image processing, pattern recognition, and machine learning constitute the methodological foundation of the system. The usability and flexibility of the system are evaluated in case studies.

1 Introduction

Computers have become standard and indispensable tools for producing practical works in architecture and urban planning. Especially, it has been important for designers and planners to create 3D computer city models in order to visualize, simulate, and estimate their design plans. In general they use commercial software to create city objects such as buildings, streets, and houses. It is labor- and time-intensive work because users need to create the city objects one by one with manual operations. Therefore, it is expected to develop a system that will generate 3D city models more easily and user-friendly.

The purpose of this research is to develop a system that can automatically generate the 3D models from satellite images. The system is expected to have ability to extract the regions of buildings/houses from the satellite images, and the 3D models are generated by assigning the height of buildings to the extracted regions.

There are mainly three problems in developing the system. First, the satellite images used for creating city models in architecture and urban planning are not standardized. Some designers use detailed pictures, and some use very simple maps. Therefore, the system should support different types of images as the input. The second problem is that precision of 3D models required by designers varies from case to case. Some cases need detailed models, and some need simple ones. Therefore, the system should allow the user to define the precision of models flexibly. The last problem is that it is not easy for any computer system to detect all objects in an image correctly. Making some mistakes is still inevitable even with the most current technologies. Therefore, the system should have the process to modify mistakes when it extracts objects incorrectly.

A fuzzy multiple layers perceptron (MLP), which is one of neuro-fuzzy models, is applied to realize the system (Pal and Mitra 1992). The fuzzy MLP can store patterns of complex relations between inputs and outputs as neural networks and represent them as a set of fuzzy <If, then> rules in linguistic forms. Using the fuzzy MLP, the system is capable of adapting flexibly to what the user desires. In the system, the user trains the patterns by giving a set of sample inputs, which are the features of pixels/regions in satellite images, and the corresponding outputs, which are the functions applied to the pixels/regions. Then the system generates 3D city models in the image at a time by using the patters. This ability makes it possible to support different types of images and precisions of city models because the user can decide what to train. Also, when the system is not trained well, the user can update the pattern by adding more samples so that the mistakes that the system made can be modified.

The goal of developing this system is to automatically generate the 3D city models once the users give sample data to create the patterns. This reduces time and labor of designers and planners dramatically. Also, the approach proposed and tested in this research may ultimately make it possible to develop a CAD system that will manipulate design knowledge in architecture and urban planning as patterns.

2 Background

2.1 Problems

Computer-aided design (CAD) systems have been required to solve various problems promptly because many practical decisions have to be made during the course of architecture and urban planning. It has often been believed that the best CAD system is realized by adding as many functions and rules as possible. Expert systems and object-oriented approaches are the typical examples. By using these technologies, it is possible to apply mathematical functions (mathematical knowledge) and logic rules (encyclopedic knowledge) for decision-making. However, there are still some limitations in the CAD systems that can be summarized as:

- Systems have more complex graphical user interfaces (GUI) as more complex tasks are implemented, making it difficult for users to learn system commands.
- Users must repeat routine processes again and again because systems cannot apply the prior strategies in similar cases.
- Systems cannot adapt to new or different types of problems flexibly because users are not allowed to change system logic rules.

Another problem is that humanistic knowledge, which is to recognize objects roughly and understand complex situations ambiguously, has been ignored in computer systems. In general, practical design processes are accomplished by combining mathematical/encyclopedic knowledge that all of us can share as a common knowledge and humanistic knowledge based on user's subjective favors and ideas. Therefore, the research to develop computer systems that are capable of manipulating humanistic knowledge has become important, and a neuro-fuzzy system is considered as an effective model to store, represent, and reuse/update the humanistic knowledge (Kobayashi 2001).

2.2 Related Area

This research is related to two research fields. One is the study of image understanding (IU) in image processing, especially the study of monocular building extraction, which is an application of IU. The other field is the study of soft computing in AI, especially the study of how to make computer systems that gain humanistic knowledge (Yager and Zadeh 1994). This research can then be defined as the integration of two technologies in order to generate 3D computer city models for architecture and urban planning.

Image processing using knowledge-based databases is a combination of traditional image processing technologies (i.e. edge detection, labeling, pattern classification, and segmentation) and expert systems with a set of <If ~, then ~> rules. The original study of monocular building extraction began in the 1980s. The initial approaches used region growing techniques and the simple models of imaging geometry (Conners et la. 1984; Harwood et al. 1987; Herman and Kanade 1986; McKeown and Denlinger 1984). Later, shadow analysis and line-corner analysis to derive building structure were also considered (Huertas and Nevatia 1988; Liow and Pavlidis 1990; Mckeown 1990). Modern approaches have used elaborate data-driven methods with multiple modes of analysis to infer 2D and 3D building structures (Jaynes et al. 1994; Shufelt 1993).

Soft computing is a field of study in artificial intelligence (AI) that implements humanistic knowledge by using the techniques of neural networks, fuzzy systems and genetic algorithms. The fusion of neural networks and fuzzy systems has been well developed in research because it is possible to translate the

output values of neurons in neural networks into the membership values in fuzzy systems. This makes it possible to create systems that can learn complex relations between inputs and outputs and can represent the relations with a set of <If ~, then ~> rules in linguistic forms at the same time.

The integrated model of neural networks and fuzzy systems is called a neuro-fuzzy system. Many different neuro-fuzzy systems/models have been introduced since the 1990s. According to S. Pal (Pal and Mitra, 1996), the methodologies can be classified as follows. 1) Incorporating fuzziness into the neural net framework (Pal and Mitra 1992), 2) Designing neural networks guided by fuzzy logic formalism (Bezdek et al. 1992), 3) Changing the basic characteristics of the neurons (Carpenter et al. 1991), 4) Using measures of fuzziness as the error or instability of a network (Ghosh et al. 1993), and 5) Making the individual neurons fuzzy (Lee and Lee 1975).

3 Methodology

In this research, the fuzzy multiple layers perceptron (MLP) (Pal and Mitra 1992) is used as the system learning engine. As it is beyond the scope of this research to explain the algorithms of fuzzy MLP, only the outline is introduced here.

3.1 Fuzzy MLP

Fuzzy MLP is one of the neuro-fuzzy systems. It is the model incorporating fuzziness into the neural net framework, and it has the ability to represent the complex relations that are learned by the weight matrices of neural networks. It can represent the relations in linguistic forms with fuzziness such as “IF feature-1 is very low and feature-2 is high, then the result is very likely Case-2”. The structure of neural networks belongs to the feedforward type model, which consists of three types of layers: one input layer, one output layer, and either one or more hidden layers. Backpropagation with a momentum factor is used as a learning algorithm in this research.

3.2 Input Data

The input data is represented as a set of feature values. Each feature is represented in terms of a combination of membership values in the linguistic property sets such as low (L), medium (M), and high (H). For example, when the *j*th input feature F_j is low, the feature is represented as, $F_j = \{\mu_{low}(F_j), \mu_{medium}(F_j), \mu_{high}(F_j)\} = \{0.95, 0.5, 0.05\}$. The input is represented as a set of features, $\mathbf{F} = [F_1, F_2, ..., F_n]$, where n is the number of features. The number of input neurons must be 3n, because each feature F_j needs 3 input neurons in fuzzy MLP.

3.3 Output Data

The output data in fuzzy MLP is represented as a set of membership values (μ), lying in the range [0,1], of each class. The output value of each neuron represents the membership value for the input data. For example, if the *k*th output neuron has the value 0.8 for the given input data $\mathbf{F}$, $\mu_{Ck}(\mathbf{F})$ is represented as 0.8, where C_k represents Class *k*, and $\mu_{Ck}(\mathbf{F})$ is the membership value of Class-k for $\mathbf{F}$. In short, the input data $\mathbf{F}$ belongs to Class-*k* with high probability. The output data is represented as a vector $[\mu_{C1}(\mathbf{F}), \mu_{C2}(\mathbf{F}), ..., \mu_{Cj}(\mathbf{F})]$, which is a set of the membership values described above, where *j* is the number of output neurons as shown in Figure 3-1. For the output data, the factor to estimate the belief of the results, bel_j^H, indicates how salient the result is compared to the other choices. It is defined as $bel_j^H = y_j^H - \sum_{i \neq j} y_i^H$. This value is used to choose linguistic forms among

very-likely ($0.8 \le bel_j^H \le 1$), likely ($0.6 \le bel_j^H < 0.8$), more-or-less ($0.4 \le bel_j^H < 0.6$), not-unlikely ($0.1 \le bel_j^H < 0.4$), and unable-to- recognize ($bel_j^H < 0.1$). The chosen forms are applied to modify <then> clauses in the generated fuzzy rules.

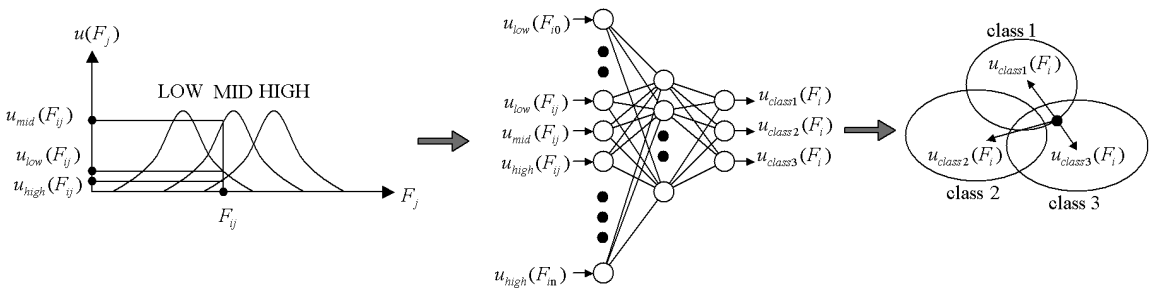


Figure 3-1. Inputs and Outputs of Fuzzy MLP

3.4 Generating Rules

This section introduces the algorithm of how fuzzy MLP generates and infers rules from its networks. It is based on the study of S. Mitra (Mitra 1995). The algorithm finds the paths from the neurons in the output layer to the neurons in the input layer. The connected neurons in the input layer represent the factors of the <if> clause, and the connected neurons in the output layer represent the factors of <then> clause. For example, when the system finds the path from Class-3 neuron in the output layer and $F1_{low}$ and $F2_{high}$ neurons in the input layer as shown in Figure 3-2, it generates a fuzzy rule: "If F1 is medium AND F2 is high, then Class-3." The certainty of Class-3 depends on the output value from the Class-3 neuron described in Section 3.3.

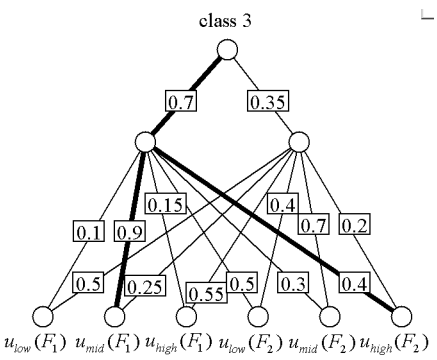


Figure 3-2: Example Rule Generation Scheme

4 System

4.1 System Outline

In order to generate 3D computer city models, the system requires two stages, segmentation and interpretation. The segmentation stage involves the assignment of pixels in images to specific labeled classes and the generation of regions by connecting the similar neighboring pixels. The interpretation stage determines what the labeled regions are after the segmentation stage. In other words, it is the stage where we apply each region to a predefined function in order to generate 3D objects.

Inside the stages described above there are two processes, the training and the refining process. The

training process is the process that supervises networks manually when there is no previous experience. On the other hand, the refining process is used to rearrange the networks by observing the generated fuzzy rules and output images. The details of these processes are shown in Figure 4-1 and 4-2.

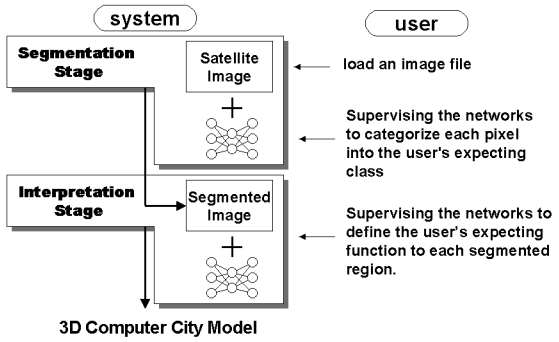


Figure 4-1: Segmentation and Interpretation Stages

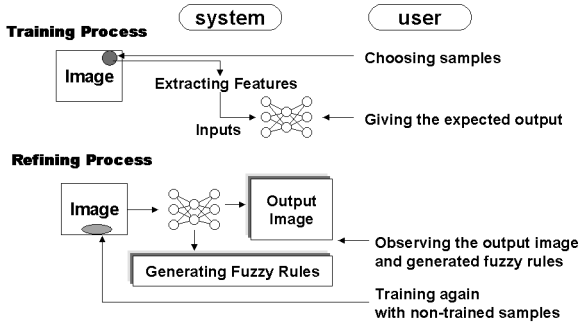


Figure 4-2: Training and Refineing Processes

The system is programmed in Java, an object-oriented programming language. It supports two image file formats, JPG and GIF, and the resultant 3D city models that are generated by the program are saved as DXF files.

4.2 Interface

The interface of application program in this system has mainly three components: a training screen, a diagram screen, and an information screen. In the training screen, the users can load a satellite image, train the images into several categories, and see the visual output images. In the diagram screen, the users can observe the weights of the neural networks, change the parameters for neural networks, and obtain the results of self-organizing map information. In the information screen, the user can see the information of samples in the training process and the fuzzy rules in linguistic forms once the relations of inputs and outputs have been calculated. These interfaces are shown in Figure 4-3.

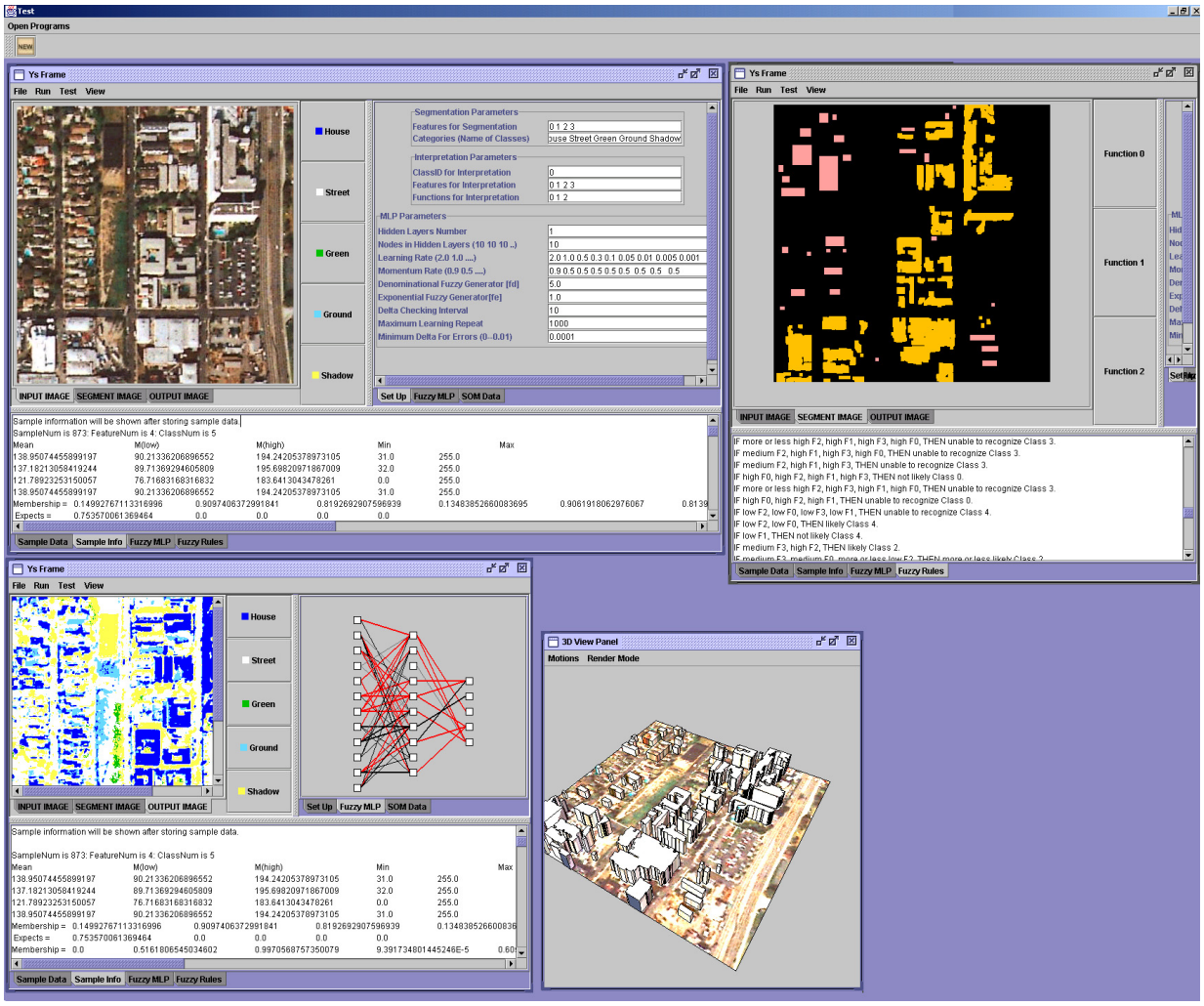


Figure 4-3: Interface

4.3 Scenario

The first stage in this system is segmentation stage. For a start, the user loads a satellite image, which is then set automatically in training screen. Next, the user picks sample pixels one by one and selects one category corresponding to each picked pixel manually. The user can also observe the selected pixel information in the information screen. After 1% or more pixels are categorized, the user calculates the neural networks. In the first calculation, the weights of neural networks are generated randomly and are supervised with the picked sample data. Next, the categorized output image is seen in the training screen and the generated fuzzy rules in the information screen. The diagram screen provides the condition of neural networks. If the output image is close to what the user expects, the training has finished. Otherwise, the user observes the generated fuzzy rules in order to find the information that is necessary but lacking. The user then categorizes more pixels by evaluating the fuzzy rules in order to adapt the system to what is required.

The next stage is interpretation stage. In this stage, the output image produced in the segmentation stage is loaded, and the user determines several geometric functions to create objects. First, the user picks sample regions one by one and select one function corresponding to the picked regions manually. After 1% or more regions are categorized, the user calculates the neural networks in the same manner

as the previous stage. If the user is satisfied with the output, the training has finished. Otherwise, the training processes are repeated.

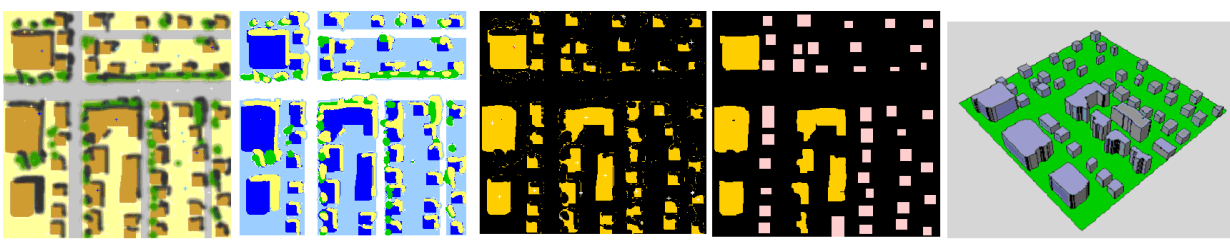
After the two processes above, the neural networks should be well trained for both segmentation and interpretation. Using these well-trained neural networks, the user can now load different satellite images and generate 3D objects automatically without any training.

5 Case Studies

In all of the case studies in this research, the same values for system setting are used in order to check the system's ability to adapt to different types of input images. In the segmentation stage, the four inputs, 1) red value, 2) green value, 3) blue value, and 4) intensity (gray value) are applied for each pixel, and the five categories (house, street, green, ground, and shadow) are defined as the outputs. In the interpretation stage, the four inputs are the maximum width, the maximum height, the area, and the ratio of the width and height; these are used to each region. For the output data there are three predefined functions: to delete the region, to create a rectangle, and to generate a polygon from the region. The neural networks that have one hidden layer with ten neurons are used in the both stages.

5.1 Drawing Case

In a simple drawing case, the input image representing the plan of a city was drawn manually using commercial photo-retouch software as shown in Figure 5-1-a). The size of the image was 300x300 pixels. The sample number used in training networks was 30 for the segmentation stage, and 22 for the interpretation stage. The percentage of accuracy correctly categorized by the system among the given samples was 100% in the segmentation stage and 95.45% in the interpretation stage. The number of generated fuzzy <if ~, then ~> rules was 16 in the segmentation stage, and 14 in the interpretation stage. The generated rules are described in Table 5-1 and Table 5-2, and the image generated in each steps is shown in Figure 5-1.



a) b) c) d) e)

Figure 5-1: Results of Drawing Image
a) Original Image, b) Categorized Image, c) Extracted Image, d) Interpreted Image, e) 3D City Model

Table 5-1: Generated Fuzzy Rules in Segmentation

If (RGB)=(med, med, low) and Gray=more or less high, -> likely House
If (RGB)=(med, more or less low, low) and Gray=med, -> likely House
If (RGB)=(med, med, low) and Gray=med, -> likely House
If (RGB)=(med, med, low) and Gray =med -> very likely House
If (RGB)=(high, ?, more or less high) and Gray=med -> likely Street
If (RGB)=(med, med, ?) and Gray=high -> not likely Street
If (RGB)=(low, med, ?) and Gray=low, ->very likely Green
If (RGB)=(low, med, low) and Gray=low, -> more or less likely Green
if (RGB)=(low, more or less, ?) and Gray=low, -> more or less likely Green
If (RGB)=(low, med, low) and Gray=low, -> likely Green
If (RGB)=(?, ?, high) and Gray=high, ->very likely Ground
If (RGB)=(med, ?, med) and Gray=med, -> not likely Ground
If (RGB)=(?, ?, more or less low) and Gray=?, likely Shadow
If (RGB)=(?, low, low) and Gray=?, ->likely Shadow
If (RGB)=(?, ?, low) and Gray=?, -> likely Shadow
If (RGB)=(?, low, more or less low) and Gray=?, -> likely Shadow

Table 5-2: Generated Fuzzy Rules in Interpretation

If (Area, Width, Height, Ratio)=(?, high, high, low), -> likely "create Polygon "
If (Area, Width, Height, Ratio)=(?, more or less high, high, low), -> likely "create Polygon "
If (Area, Width, Height, Ratio)=(high, ?, high, med), -> likely "create Polygon "
If (Area, Width, Height, Ratio)=(?, med, high, ?), -> likely "create Polygon"
If (Area, Width, Height, Ratio)=(?, more or less high, med, low), -> more or less "create Polygon "
If (Area, Width, Height, Ratio)=(low, med, med, low), -> likely "create Rectangle
If (Area, Width, Height, Ratio)=(low, med, med, more or less low), -> likely "create Rectangle
If (Area, Width, Height, Ratio)=(low, med, med, low), -> more or less "create Rectangle
If (Area, Width, Height, Ratio)=(low, med, low, med), -> likely "create Rectangle
If (Area, Width, Height, Ratio)=(low, low, low, ?), -> likely "delete the Region"
If (Area, Width, Height, Ratio)=(low, low, low, ?), -> very likely "delete the Region"
If (Area, Width, Height, Ratio)=(low, low, med, ?), -> likely "delete the Region"
If (Area, Width, Height, Ratio)=(?, med, med, low), -> ???
If (Area, Width, Height, Ratio)=(low, low, low, ?), -> very likely "delete the Region"

5.2 Practical Satellite Image Case

This case uses the satellite image of Los Angeles from "Los Angeles CityRom" (Small Blue Planet Atlas Co. 1999) as the input. The size of the image was 400x400 pixels. The sample number used in training networks was 1000 for the segmentation stage, and 100 for the interpretation stage. The percentage of accuracy correctly categorized by the system among given samples was 81.05% in the segmentation stage and 76.04% in the interpretation stage. The number of generated fuzzy <if ~, then ~> rules was 86 in the segmentation stage, and 43 in the interpretation stage. Due to the limitation of space, the generated rules are not shown here. The image generated in each steps is shown in Figure 5-2.

5.3 Similar Image Case

Similar satellite images were used in two case studies. In each case, the image of a different location from Case 5.2 was used with the neural network weights trained in Case 5.2. In the first case shown in Figure 5-3, the system could generate a 3D city model successfully without any additional training. Alternatively, in the second case many of the pixels were not categorized correctly, particularly the houses with red roofs. As the image used in Case 5.2 did not have any red roof houses, the system recognized the red pixels as a ground. Therefore, 20 more samples were required to train the system to recognize a red roof as a house. These were added to the previous samples and retrained. The results are shown in Figure 5-4.

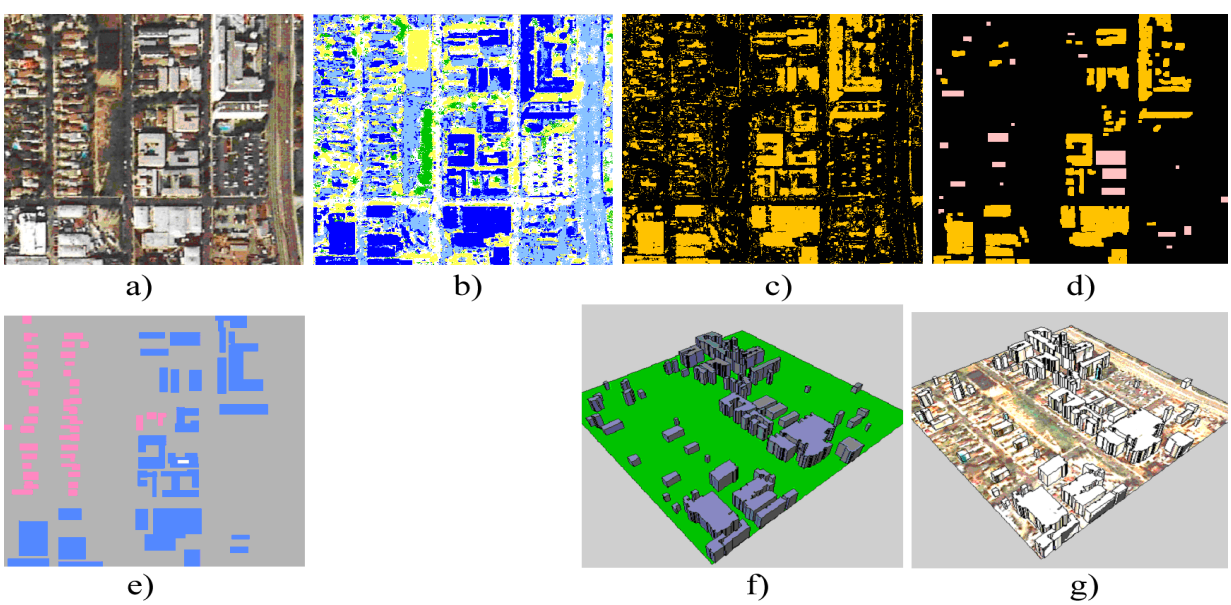


Figure 5-2: Results of Satellite Image**:** a) Original Input Satellite Image; b) Categorized Image; c) Extracted Image; d) Interpreted Image; e) Expected Result Image; f) 3D City Model; g) 3D City Model with Texture Mapping

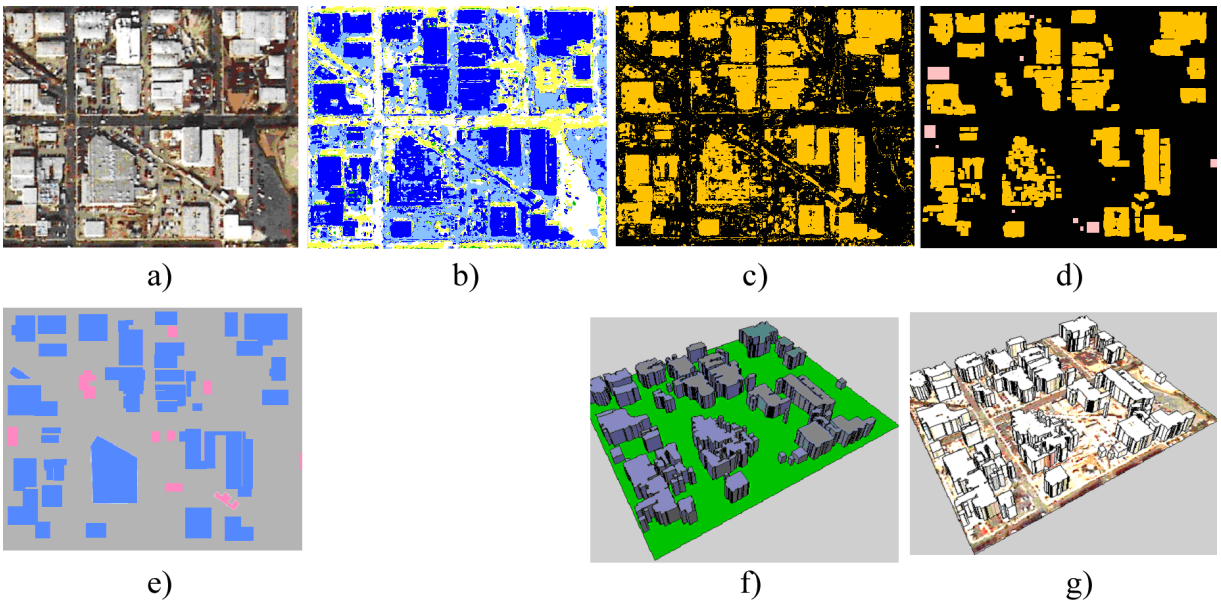


Figure 5-3: Results of Similar Satellite Image without Training
a) Original Input Satellite Image, b) Categorized Image, c) Extracted Image, d) Interpreted Image, e) Expected Result Image, f) 3D City Model, g) 3D City Model with Texture Mapping

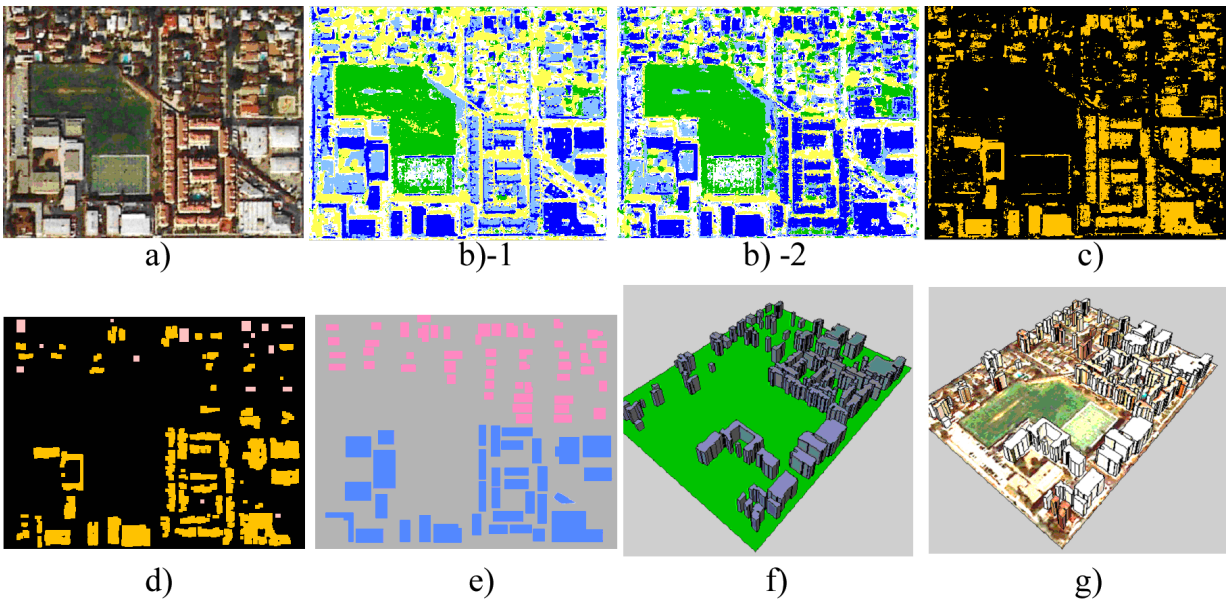


Figure 5-4: Results of Similar Satellite Image with Re-training
a) Input Satellite Image, b)-1 Categorized Image without Training, b)-2 Categorized Image after Training, c) Extracted Image, d) Interpreted Image, e) Expected Result Image, f) 3D City Model, g) 3D City Model with Texture Mapping

6 Consideration

6.1 Results in Segmentation

The results of simple drawing case show that the system can categorize pixels and regions well enough to satisfy the user requirements. On the other hand, in the three cases using practical satellite images, the system cannot recognize some objects well, especially the small houses. The main reason is the resolution of image is not high enough to distinguish the small houses from the color information. Although the user can detect houses in the images from both color information and from the relations among the surrounding objects (streets, trees, shadows, etc.), the system does not have the ability to check the relations between different type objects.

The evaluation of system effectiveness is made by comparing the system outputs and the results expected by the user. It is expected that the number of objects that the system detects should match user's manual count. In the evaluation, two types of objects are defined. These are analyzed separately. One type is "Clear Object", which is the object visibly detectable as a building or house. The other is "Ambiguous Object", which is unclear but recognizable as a building or house.

Figure 5-2-d), 5-3-d), and 5-4-d) show the system outputs, and Figure 5-2-e), 5-3-e), and 5-4-e) show the results expected by the user. In the expected images, the blue areas are Clear Objects, and the pink areas are Ambiguous Objects. These decisions are based on the user's subjective requirements, and the expected images are drawn manually. The results of this comparison are described in Table 6. For Clear Objects, the system can perform 94.2% correctly. For Ambiguous Objects, it can perform 39.6% correctly. Therefore, it is concluded that the system can detect Clear Objects well, but there is a room for improvement to detect Ambiguous Objects.

Table 6: Results of Comparison.

Case	Clear Objects	Ambiguous Objects	Objects Mistaken by System
Figure 5-2	23/25 (92.0%)	6/37 (16.2 %)	14
Figure 5-3	28/19 (96.6 %)	4/9 (44.4 %)	8
Figure 5-4	31/33 (93.3 %)	28/48 (58.3 %)	12

6.2 Fuzzy MLP

Since the purpose of this research is to develop a system to adapt to several types of images rather than find the optimum setting for neural networks, the results of experiments to find the best setting for the neural networks are not described in this paper.

The parameters for fuzzy MLP in the systems such as the number of hidden layers, the number of neurons in the hidden layers, the number of samples, etc. are decided from the results of 1) the number of epochs in training samples, 2) the percentage of samples the system can detect correctly, 3) the self-organizing map for samples, and 4) the output images measured by the user.

6.3 Generated Fuzzy Rules

In this research, the generated fuzzy rules are used to observe what information the fuzzy MLP has learned and to assist users in training and refining the weights of neural networks. However, the usefulness of the generated fuzzy rules is not proved because it is easier to understand what types of data are missing from the images than from the generated rules. Moreover, the generated fuzzy rules are too complicated for users to recognize the information exactly. The generated rules are too many, and some of them overlap. For example, the two rules, "if feature-1 is high, then likely Class-2" and "if feature-1 is high, then very likely Class-2" were recognized as two different rules in the system.

6.4 3D Model

Although this system demonstrates that 3D city models can be created from satellite images, the generated models are not as accurate as the real ones. However, the generated 3D models can be useful in visualizing cities in computers. For example, there may be many cases in which only simple 3D volume objects are needed.

7 Conclusion

A computer system that can generate 3D computer city models from satellite images is implemented. A fuzzy multiple layers perceptron (MLP) is applied as a learning engine of the system. The approach to implement the technologies of neuro-fuzzy in computer-aided design (CAD) systems is relatively new for the time being.

The system in this research generates 3D city models automatically after the user trains the patterns by giving a set of the sample inputs and the outputs corresponding to the inputs. This system has flexibility of adapting to the different kinds of inputs such as paintings and real images. This flexibility does not exist in traditional CAD systems.

In addition, the system allows the user to apply prior patterns to another case that is similar to previous ones. If the user trains the neural network weights well, the system can generate 3D models automatically. On the other hand, if user training is not enough, and the results are far from the expected ones, the user has to observe the generated fuzzy rules and refine the system by giving additional samples. In short, the system in this research generates the expected 3D city models semi-automatically. This ability still helps to heighten the usability of CAD system because the user does not have to create 3D models one by one in the system.

There is still room for improvement in this research as described below.

- It was not examined whether the predefined functions/pixel-features for segmentation and interpretation were effective enough.
- The function to simplify the generated fuzzy rules enough for the users to understand the relations was not implemented.
- The height of building was assigned by using predefined values because the function to calculate the height of buildings from its shadows was not implemented.

Furthermore, neural networks themselves have some inherent problems in terms of their initial configuration. For example, it is difficult to find the optimum number of hidden layers and nodes in each hidden layer.

The future works will be to find the solutions for three problems above and minimize the inherent problems of neural networks in this research, so that a more practical and effective system that generates 3D city models from satellite images automatically can be developed.

References

Bezdek, J.C., E.C. Tsao and N. R. Pal. (1992). Fuzzy Kohonen clustering networks. *IEEE International Conf. on Fuzzy Systems*: 1035-1043. San Diego.

Conners, R. W., M. M. Trivedi, and C. A. Harlow. (1984). Segmentation of a High-Resolution Urban Scene Using Texture Operators. *Computer Vision, Graphics, and Image Processing* 25: 273-310.

Carpenter, G.A., S. Grossberg, and D. B. Rosen. (1991). Fuzzy ART-Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks* 4:759-771.

Ghosh, A., N. R. Pal, and S. K. Pal. (1993). Self-organization for object extraction using multiplayer neural network and fuzziness measure. *IEEE Trans. of on Fuzzy System* 1:54-68.

Harwood, D., S. Chang, and L. Davis. (1987). Interpreting Aerial Photographs by Segmentation and Search. *Proceedings of the DARPA Image Understanding Workshop* 1987-2: 475-482.

Herman, M. and T. Kanade. (1986). Incremental Reconstruction of 3D scenes from Multiple, Complex Images. *Artificial Intelligence* 30: 289-341.

Huertas, A. and R. Nevatia. (1988). Detecting Buildings in Aerial Images. *Computer Vision, Graphics, and Image Processing* 41, 131-152.

Jaynes, C., R. Stolle, and R. Collins. (1994). Task Driven Perceptual Organization for Extraction of Rooftop Polygons, *ARPA Image Understanding Workshop* 1994:359-365.

Kobayashi, Y. (2001). *3D City Modeler with Fuzzy Multiple Layer Perceptron.* Dissertation: University of California, Los Angeles.

Lee, S.C. and E. T. Lee. (1975). Fuzzy neural networks. *Mathematical Bioscience* 23: 151-177.

Liow, Y.T. and T. Pavlidis. (1990). Use of Shadow for Extracting Buildings in Aerial Images, *Computer Vision, Graphics, and Image Processing* 49: 242-277.

McKeown, D. M., and J. L. Denlinger. (1984). Map-guided feature extraction from aerial imagery, *Proceeding of Workshop on Computer Vision-Representation and Control.* 205-213, Annapolis, MD.

McKeown, D. M. (1990). Toward Automatic Cartographic Feature Extraction from Aerial Imagery. *Mapping and Spatial Modeling for Navigation* F65: 149-180.

Pal, S.K. and S. Mitra. (1999) *Neuro-Fuzzy Pattern Recognition*, John Wiley & Sons Inc., New York.

Pal, S.K. and S. Mitra. (1992). Fuzzy Multi-layer perceptron. *IEEE Trans. of Neural Networks* 3: 683-697

Shufelt, J.A. (1996). Projective Geometry and Photometry for Object Detection and Delineation Techniques, *APRA Image Understanding Workshop* 1996.2:1113-1132.

Small Blue Planet Atlas Company. (1999). Los Angeles CityRom (CD-ROM),

Yager, R.R. and L.A. Zadeh. (1994). *Fuzzy Sets, Neural Networks, and Soft Computing*. Van Nostrand Reinhold, New York.