CHAPTER II

THEORETICAL BACKGROUND AND RELATED WORKS

2.1 Three Dimension to Two Dimension Transformation Process

This chapter discusses the theoretical background in four different subjects (1) 3-D to 2-D transformation process, (2) 2-D to 3-D transformation process, (3) wavelet transform theory, and (4) neural network theory. The subject in 3-D to 2-D transformation process during image acquisition using sensor arrays, a simple image formation model, image sampling and quantization and representing the digital images. The details of 2-D to 3-D transformation process is several kinds of algorithm relating to the depth recovery information from the images. The overview of wavelet transform is discussed in the third part and the last part is the neural network theory.

2.1.1 Image Acquisition Using Sensor Arrays

The images that we are interested are generated by the combination of an *illumination* source and a reflection or absorption of energy from that source by elements in the *scene* being imaged [1]. Light intensity can be translated into an electrical signal by using the photosensitive cells. These devices can be used to make an image on the camera by representing levels of light intensity for each *dot* on the the image.

Figure 2.1 shows the principal sensor arrangement used to transform illumination energy into a digital image. The process is as simple as follows: The incoming energy is transformed into a voltage by a combination of input electrical power and sensor material responsive to the particular type of energy being detected. The output voltage waveform is the response of the sensor(s), and a digital quantity is obtained from each sensor by digitizing its response. The sensor array shown in Figure 2.2(c) is a two dimensional array.

The image capturing process with the array sensors is shown in Figure 2.2. This figure shows the 3D object to 2D image transformation starting at the energy from an illumination source(a) being reflected from an object(b). The light energy is transmitted

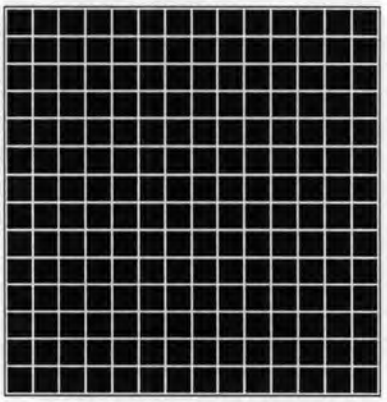


Figure 2.1: Array sensor.

through the imaging system(c). The imaging system in Figure 2.2(c) is to collect the incoming energy and focus it onto an image plane(d). A simple imaging system includes a lens and array sensor. The light energy on the array sensor are transformed from the light energy to the electrical signal in this step. The light energy to electrical signal transformation is the sampling and quantization. The output of quantization process can be seen in the digital image on the digital device such as image plane on the monitor(e).

2.1.2 A Simple Image Formation Model

We can described the images by using a two-dimensional functions of the form $f(x, y)$. The value or amplitude of f at spatial coordinates(x, y) is a positive scalar quantity whose physical meaning is determined by the source of the image. The source images in this dissertation are single monocular gray-scale images. Then the amplitude of the f will have the range number between 0 to 255.

$$0 < f(x, y) < 255 \tag{2.1}$$

The function $f(x, y)$ may be characterized by two components: (1) the amount of source illumination incident $i(x, y)$ on the scene, and (2) the amount of illumination reflected $r(x, y)$by the objects in the scene. The two functions are combined as a product to form $f(x, y)$:

$$f(x, y) = i(x, y)r(x, y) \tag{2.2}$$

where

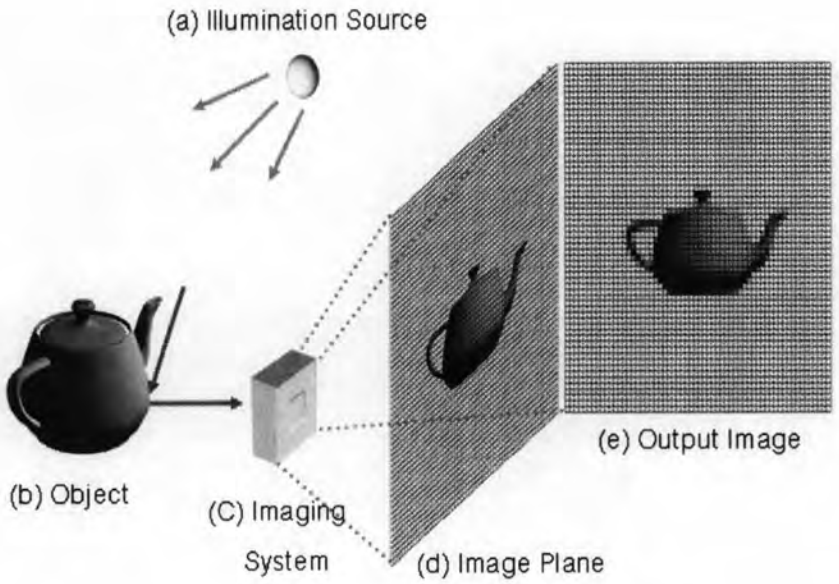


Figure 2.2: An example of the digital image acquisition process. (a) Energy (*illumination*) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

$$0 < i(x, y) < 255 \tag{2.3}$$

and

$$0 < r(x, y) < 1 \tag{2.4}$$

The nature of $i(x, y)$ is determined by the illumination source, and $r(x, y)$ is determined by the characteristics of the objects.

2.1.3 Image Sampling and Quantization

From the discussion in the previous section, before the product from the sensor arrays can be seen in the digital images format, we need to convert the continuous signal to the digital number by using the two processes: *sampling* and *quantization*.

Basic Concepts in Sampling and Quantization

The basic concept of sampling and quantization is illustrated in Figure 2.3. Figure 2.3(a) shows a continuous image, $f(x, y)$, the $x - axis$ represent the time line of continuous signal, and $y - axis$ represent the amplitude. To convert it to a digital form, we have to sample the function in both coordinates and in of the signal. Digitizing in time-line of signal values is called *sampling*. Digitizing the amplitude values is called *quantization*. The one-dimensional function shown in Fig. 2.3(b) is a plot of amplitude (gray level) values of the continuous image along the line segment A *to* B in Fig. 2.3(a). To sample this function, we take equally spaced samples along line $A - B$, as shown in Fig. 2.3(c). The location of each sample is given by a vertical tick mark in the bottom part of the figure. The samples are shown as small red circle superimposed on the function. The set of these discrete locations gives the sampled function. However, the values of the samples still span (vertically) a continuous range of gray-level values. In order to form a digital function, the gray-level values also must be converted (*quantized*) into discrete quantities.The right side of Figure 2.3(c) shows the gray-level scale divided into eight discrete levels, ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight gray levels. The continuous gray levels are quantized simply by assigning one of the eight discrete gray levels to each sample. The digital samples resulting from both sampling and quantization are shown in Figure 2.3(d). This sampling and quantization processes are based on the strip sensor

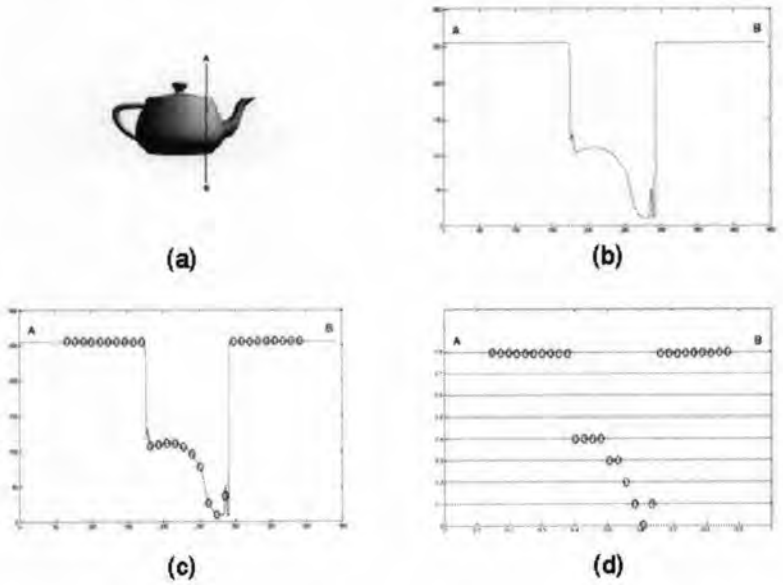


Figure 2.3: Generating a digital image. (a) Continuous image. (b) A scan line from A to B in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling signal. (d) Quantization of the scan line from A to B.

type. For the conventional digital image, these processes are based on the array sensor

type. The product after the quantization process is a digital image.
Figure 2.4 illustrates this concept. Figure 2.4(a) shows a continuous image projected

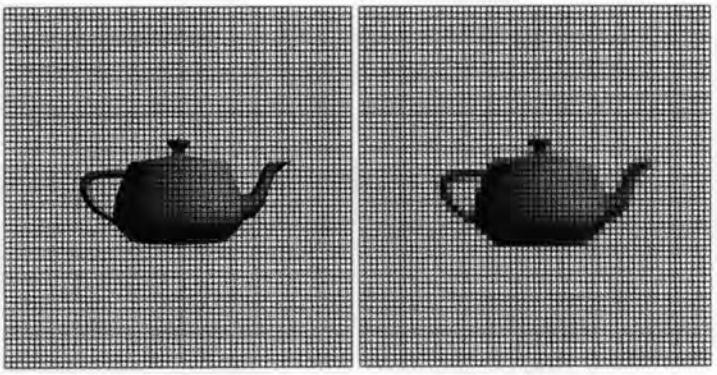


Figure 2.4: Overview of sampling and quantization processes. (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

onto the plane of an array sensor. Figure 2.4(b) shows the image after sampling and quantization.

2.1.4 Representing Digital Images

Assume that an image $f(x, y)$ is sampled so that the resulting digital image has M rows and N columns. The values of the coordinates (x, y) is the discrete numbers. We used the integer values for these discrete coordinates notational. The values of the coordinates at the origin are (x,y)=(0, 0). The next coordinate values along the first row of the image are represented as $(x, y) = (0, 1)$. It is important to keep in mind that the notation $(0, 1)$ is used to signify the second sample along the first row. It does not mean that these are the actual values of physical coordinates when the image was sampled. Figure 2.5 shows the coordinate convention used throughout this dissertation.

For the $M * N$ digital image can be shown in the matrix form as follows:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(0,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.5}$$

Each element of this matrix array is called an *image element* or *pixel*. The digitization process requires decisions about values of pixel. The numbers in the pixels must be

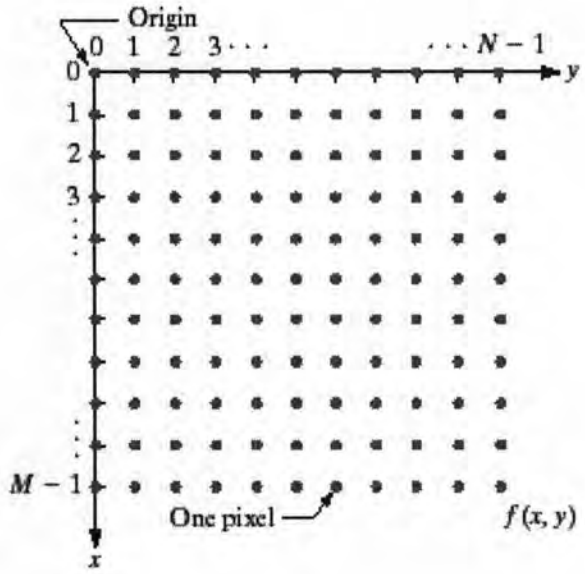


Figure 2.5: Coordinate convention used in this dissertation to represent digital images.

positive integers, in this dissertation, we specifying the image type to the gray-scale image. Thus, the number of gray levels typically is an integer power of 2:

$$L = 2^k \tag{2.6}$$

where L is the number of gray value and k is the number of gray levels. The number of gray levels corresponding to each value of k is shown in image.

2.2 2D to 3D Transformation Process

In images, three-dimension scenes are projected on a two-dimension image plane. Thus the depth information is lost and special imaging techniques are required to retrieve the surfaces images or volumetric images. Depending on the number of input images, we can categorize the existing conversion algorithms into two groups: algorithms based on two or more images and algorithms based on a single still image. In the first case, the two or more input images could be taken either by multiple fixed cameras located at different viewing angles or by a single camera with moving objects in the scenes.

Our work is using focus and defocus information in an image to be a features and the source of image is a single monocular image, then, I will literature the method that using focus and defocus features in both more than one image and only one image

Table 2.1: The depth cues used in 2D to 3D conversion algorithms.

The Number of Input Images	Depth Cues
Two or More Images (binocular or multi-ocular)	Binocular disparity Motion Defocus Focus Silhouette
One Single Image (monocular)	Defocus Linear perspective Atmosphere Scattering Shading Patterned texture (Incorporates relative size) Symmetric patterns Occlusion - Curvature - Single Transform

methods.

2.2.1 Defocus using more than two images

Depth-from-defocus methods generate a depth map from the degree of blurring present in the images. In a thin lens system, when the objects stay in the focal length, the image will focused image. On the other hand, if the objects stay out-of-focus, the image appearing blurred. Figure 2.6 shows a thin lens model of an out-of-focus real world point P projected onto the image plane. Its corresponding projection is a circular blur patch with constant brightness, centered at P" with a blur radius of σ. The blur is caused by the convolution of the ideal projected image and the camera point spread function $(PSF)g(x,y,\sigma(x,y))$ where (x,y) are the coordinates of the image point P". It is usually assumed that $\sigma(x,y) = \sigma$ where σ is a constant for a given window to simplify the system and Gaussian function is used to simulate the PSF:$g(x,y) = \frac{1}{\pi\sigma^2} e^{\frac{x^2+y^2}{\sigma^2}}$. In order to estimate the depth u, we need the following two equations. The fundamental equation of thin len describes the relation between u, v and f as:

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f} \tag{2.7}$$

Pentland [2] has derived a relationship between the distance u (Figure ??) and the blur σ in equation(2.8):

$$u = \begin{cases} \frac{fs}{s-f-kf\sigma} if\, u > v \\ \frac{fs}{s-f+kf\sigma} if\, u < v \end{cases} \tag{2.8}$$

where u is the depth, v is the distance between the lens and the position of a perfect focus, s is the distance between the lens and the image plane, f is the focal length of the lens, and k is a constant determined by the lens system. All of s, f and k are camera parameters that need the calibration process. With Eq.(2.8), the problem of computing depth u is converted into a task of estimating camera parameters and the blur parameter σ. When camera parameters can be obtained from camera calibration, the depth u can be computed from Eq.(2.8) once the blur parameter σ is known. The depth-from-focus algorithms focus thus on the blur radius estimation techniques.

With two unknowns u and f, Eq.(2.8) is under-constrained. In this case,

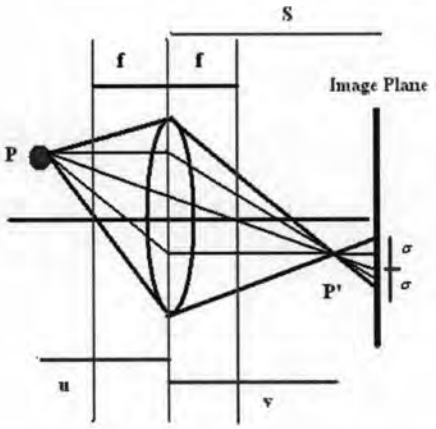


Figure 2.6: Thin lens model.

the output signal can be a projection of an out-of-focus step edge, an in-focus smooth transition (e.g. a smooth texture) or infinite situations in between these two extremes [3]. To solve the problem, most of the depth-from-defocus algorithms rely on two or more images of the same scene taken from the same position with different camera focal settings to determine the blur radius. When the blur radius is estimated and camera parameters are obtained from camera calibration, then the depth can be computed by Eq.(2.8). The example of the blur radius estimation techniques are inverse filtering. The blur difference is retrieved by solving a set of equations, derived from the observation that the coefficients of the Hermite polynomial estimated from the more blurred image, the less blurred image and the blur difference.

2.2.2 Focus

The depth-from-focus approach is closely related to a family of algorithms using depth from defocus. The main difference is that the depth-from-focus requires a series of images of the scene with different focus levels by varying and registering the distance between the camera and the scene, while depth-from-defocus only needs two or more images with fixed object and camera positions and use different camera focal settings. Figure 2.7 illustrates the principle of a depth-from-focus approach [4]. An object with an arbitrary surface is placed at the translational stage, which moves toward the camera (optics) starting from the reference plane. The focused plane is defined by the optics. It is located at the position where all points on it are focused on the camera sensor plane. Let s be a surface point on the object, when moving the stage towards the focused plane, the images of s become more and more focus and will obtain its maximum sharpness when s reaches the focused plane. After this, moving s furthermore makes its image defocus again. During this process, the displacements of the translational stage are registered. If we assume that the displacement is $defoused$ when s is maximally focused and the distance between the *focused plane* and the reference plane is d_f, then the depth value of s relative to the stage will be determined as $d_s = d_f - d_{focused}$. Applying this same procedure for all surface elements and interpolating the focus measures, a dense depth map can be constructed.

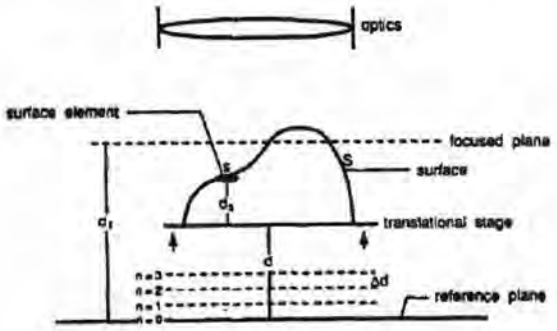


Figure 2.7: Depth from focus.

2.2.3 Defocus using a single image

In section 2.2.3(Defocus using more than two images), depth-from-defocus algorithms based on two or more images are introduced. The reason for using more images is to eliminate the ambiguity in blur radius estimation when the focal setting of the camera is unknown. Wong and Ernst [3] have proposed a blur estimation technique using a single image based on the second derivative of a Gaussian filter [5]. When filtering an edge of blur radius σ with a second derivative of a Gaussian filter of certain variance s

, the response has a positive and a negative peak. The distance between the peaks as d , which can be measured directly from the filtered image. The blur radius is computed according to the formula $\sigma^2 = (\frac{d}{2})^2 - s^2$ (see Figure 2.8). When the camera parameters are unknown, we can still estimate the relative depth level of each pixel based on its estimated blur radius by mapping a large blur value into a higher depth level and a smaller blur value to a lower depth level.

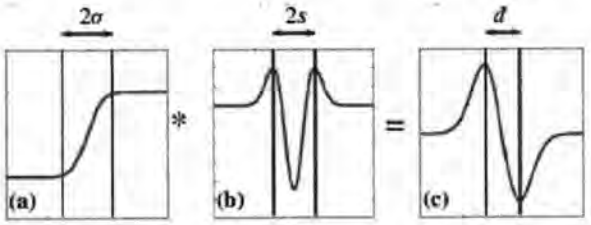


Figure 2.8: Radius estimation: (a) Blurred edge; (b) Second derivative of Gaussian filter with standard derivation s; (c) Filter response, the distance d between peaks can be measured from the filtered image. Figure source: reference[7].

2.3 Wavelet transform

2.3.1 Introduction to Wavelet

A signal is usually defined as an oscillation function of time and space. Fourier analysis is a signal analysis tool that expanded the signal in the term of periodic sinusoidal signal. For the non-periodic signal, Fourier analysis is computationally hard to detected. Gabor was developed windowed Fourier transform [6] in 1940 which is know as the sliding-window Fourier transform. The idea is to study the frequencies of a signal segment by segment. The size of the segment is defined by window, which is a fixed size.

Fourier transform compares a signal in infinite sines and cosines of different frequency in order to how much of each frequency of the signal contains. Windowed Fourier transform compares a segment of the signal to bits of oscillating curves. For a small window, the high signal frequencies was appeared but the low signal frequencies are loss in the small window size. On the other hand, thee wider window will contains the low signal frequencies and loss the high signal frequencies. The wavelet transform was developed to solve this problem. Wavelet is a function that can be translations and dilates itself in order to fit the original signal in the window. From this properties, we can obtains both high and low signal frequencies that appeared in the window.

The basis function of the wavelet transform is generated from a basic wavelet function $\psi(x)$ together with translations, b, and dilations, a. The scaled and translated

wavelet can be written as:

$$\psi_{a,b}(x) = f\left(\frac{x-b}{a}\right) \tag{2.9}$$

where $b \in \Re^+, a \in \Re$. $f(x)$ is translated by $f(x-b)$. The function $f(x)$ is translated by b units and dilated by a scaling factor, a, that stretches or compress the wavelet. Figure 2.9 shows wavelet functions with different scaling factor, a, and translation factor, b. The signal function $f(x)$ can be described by a linear combination of wavelet function

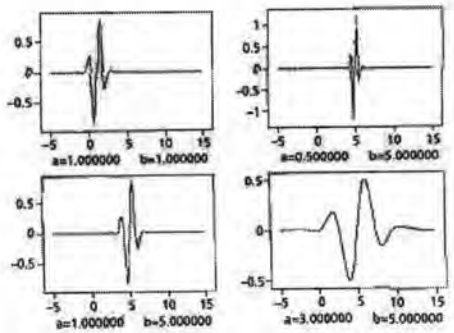


Figure 2.9: Wavelet Functions.

$\psi_{j,k}(x)$ as

$$f(x) = \sum_{j=0}^{a}\sum_{k=0}^{b} c_{jk}\psi_{jk}(x) \tag{2.10}$$

where j and k are integer indices and c_{jk} is wavelet coefficients.

2.3.2 Continuous Wavelet Transform

The continuous wavelet transform decomposes a signal $f(x)$ into a set of basis functions $W_f(a,b)$, which is called wavelets:

$$W_f(a,b) = \int_0^1 f(x)\psi_{a,b}(x)dx \tag{2.11}$$

The wavelets are generated from a mother wavelet $\psi(x)$ by scaling factor, a, and translation factor, b:

$$\psi_{a,b} = \psi\left(\frac{x-b}{a}\right) \tag{2.12}$$

where $a \in \Re^+, b \in \Re$. The signal can be reconstructed by

$$f(x) = \iint W_f(a,b)\psi_{a,b}(x)d(x) \tag{2.13}$$

2.3.3 Discrete Wavelet Transform(DWT)

To reducing the time-bandwidth of continuous wavelet transform output products, we used the discrete wavelet transform(DWT) to solve the problem of the large time-bandwidth output product of the wavelet transform. In addition to computing the wavelet transform in digital computer, the DWT decomposes the signal into a set of basis functions.

$$f(x) = \sum_{j=0}^{a}\sum_{k=0}^{b} c_{jk}\psi_{jk}(x) \tag{2.14}$$

where $\psi_{jk}(x)$ are wavelet basis functions and c_{jk} are wavelet coefficients that can be calculated by the inner product.

$$c_{jk} = \langle f(x), \psi_{jk}(x)\rangle = \int f(x)\psi_{jk}(x)dx \tag{2.15}$$

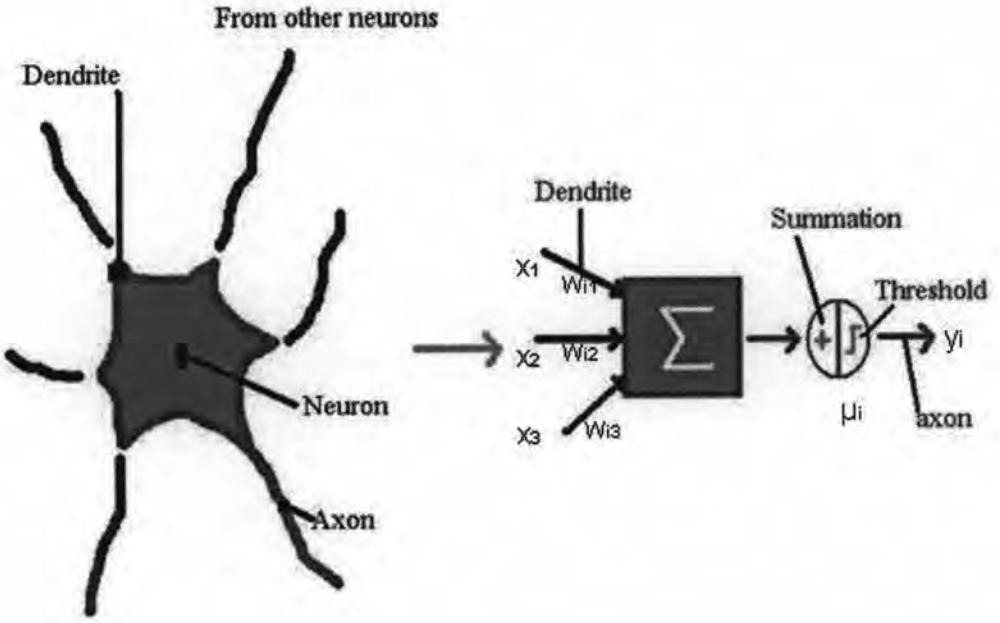
2.4 Artificial Neural Networks (ANNs)

Background Theory of Artificial Neural Network

In some hard classification problems, we can used the computer to help the human to reduces the classification time and sometime, we want to have an automatical results from the generating process of computer. The problems are inspired the scientist to discover the computer technique for simulating the human brain that we called *Artificial Neural.* The structures of human neurons and artificial neural network models are shows in Fig. 2.10.

Figure 2.10(b) shows the structure of a single node of neural. The neural in figure, designated the j^{th} neuron that is this neuron accepts inputs from other neurons and its outputs to other neurons. Any neuron in network are interconnected together.

The generalized neuron gets its inputs from the interconnections from the output of other neurons. As same the biological of neuron brain, the interconnection part between neuron cell are also known as *synapses.*In the neural network model, the synapse connections are weighted. When the i^{th} neuron sends a signal to j^{th} neuron, the signal is multiplied by the weighting on the i, j synapse. This weighting can be symbolized as w_{ij}. If the output of the i^{th} neuron is designated as x_i, then the input to the j^{th} neuron from the i^{th} neuron is $x_i w_{ij}$. Summing the weighted inputs to the j^{th} neuron can be



(a) (b)

Figure 2.10: (a)Component of Human Neurons and (b)The diagram of generalized node in an artificial neural network. The i^{th} node receives inputs (x1,x2,x3) from other nodes. Each of these multiplied by the corresponding synapse weight, (w_{ij}), and the resulting products are summed within the j^{th} node to produce the activation, μ_i. The activation is transformed to produce the node's output signal, y_i.

shown in Eq. 2.16.

$$\mu_i = \sum_i x_i w_{ij} - \theta_i \tag{2.16}$$

where θ_j is a bias term.

The sum values that we obtain is called the activation of the neuron. This activation can be zero, positive or negative, because the weightings and inputs can be either positive or negative such as the family of sigmoid function, hard limit function or liner transfer function.

2.4.1 Multilayer Perceptron Neural Networks (MLP Networks)

The structure of the neural network can be shows as Fig.2.11. This type of architecture is a part of the feedforward neural networks. The feedforward neural network share a common feature that all neurons in a layer are connected to all neurons in the adjacent

layers through unidirectional branches. Feedforward networks do not allow connections between neurons within any layer of the architecture. The outputs of nodes in one layer are transmitted to nodes in another layer through links. The link between nodes indicates flow of information during recall. During learning, information is also propagated back through the network and used to update connection weights between nodes.

Let o_j be the output of the previous layer and w_{ij} the connection weight between the i^{th} node in one layer and j^{th} node in the previous layer. The total input to the i^{th} node of a layer is:

$$net_i = \sum_j w_{ij} o_j$$

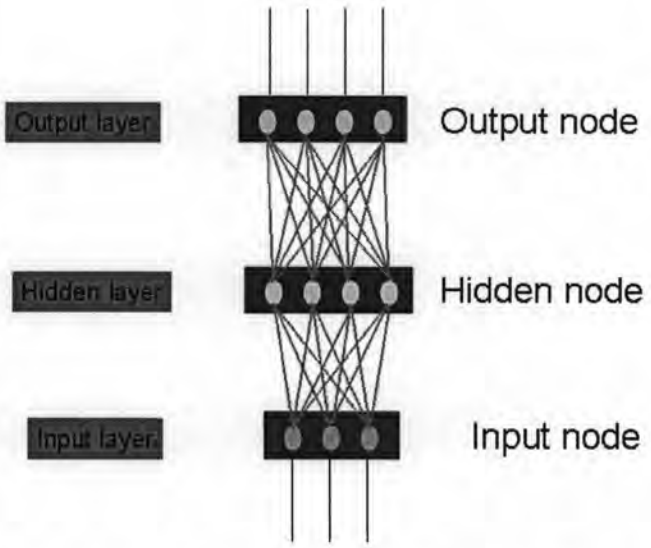


Figure 2.11: A simple feed forward three-layers perceptron neural network.

2.4.2 Learning Algorithm

This section is described the backpropagation learning algorithm. A simple feed forward three-layers perceptron neural network is shown in Fig.2.11. The input unit in the first layer serve only to the next layer, they perform no input summation. Each neurons in the first and the second hidden layers produces *net* and *out* signals as shown in Eq.2.17 and Eq.2.18.

$$net = \sum_{i=1}^{n} O_i w_i \tag{2.17}$$

$$out = F(net)out = \frac{e^{net_i} - e^{-net_i}}{e^{net_i} + e^{-net_i}} \tag{2.18}$$

where F is the activation function, which has a hyperbolic tangent activation function. The backpropagation learning algorithm can be shown below [7].

Step 1: Initialize the weighed. The weights between input layer, the hidden layers are initialized to small random values.

Step 2: Present an input vector $x = (x_1, x_2, x_3, ..., x_n)^T$ to the input layer and obtain the output vector $y = (y_1, y_2, y_3, ..., y_n)^T$ at the output layer. In order to obtain the obtain the output vector y, calculation is done layer by layer by starting at the hidden layer. The net value of each neuron in the hidden layer is calculated as the weighted sum of its input. The net input is then passed through the activation function F to produce *out* values for each neuron in the hidden layer. The outputs of the hidden neurons serves as inputs to neurons in the output layer. The process is repeated to obtain the output y at the the output layer.

Step 3: Weights Adjustment. To adjusting the weight, the output y is compared with the desired output vector or the target vector d, and the error between the two vectors is obtained. The error is then propagate backward to obtain the change in weight Δw_{ij} that is used to update the weights. Δw_{ij} for weights between the output layer and the hidden layer is given by

$$\Delta w_{ij} = \alpha O_j \delta_i \tag{2.19}$$

where α is a learning rate, O_j is the output of neuron j in the hidden layer, and δ_i is given by

$$\delta_i = [\partial F(net_i)/\partial net_i](d_i - O_i)\delta_i = O_i(1 - O_i)(d_i - O_i) \tag{2.20}$$

where O_i represents the actual output of neuron i in the output layer and d_i represent the desired output at neuron i in output layer. For the hidden layer is has no target vector, the backpropagation algorithm in this layer is done by propagating the output error back through layer by layer, adjusting weights at each layer. The weights adjustment between the hidden layer can be obtained as

$$\Delta w_{ij} = \beta O_j \delta_{Hi} \tag{2.21}$$

where β is a learning rate coefficients for the hidden layer, O_j is the output of neuron j in the input layer, and

$$\delta_{Hi} = O_i(1 - O_i) \sum\nolimits_k \delta_k w_{ik} \tag{2.22}$$

where O_i is the output neuron i in the hidden layer, and the summation term represents the weighted sum of all δ values corresponding to neurons in output layer that are

obtained by using Eq.2.20.
Step 4: Update the weights:

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij} \tag{2.23}$$

where $w_{ij}(n+1)$ is the value of the weight at iteration $n+1$, and $w_{ij}(n)$ is the value of the weight at iteration n.
Step 5: Obtain the error ε for neurons in the output layer.

$$\varepsilon = \sum_i (O_i - d_i)^2 \tag{2.24}$$

If the error ε is greater than the minimum error, then repeat step 2 through 4, otherwise stop the training algorithm.

2.5 Related Work

2.5.1 Machine Learning Approach

Our works is based on the machine learning of the degree of focused and blurred of texture information that distributed in the foreground, middleground and background regions in an image. When the type of image is the depth-of-field image, the differencing of three regions are affected to the degree of blurring of the regions. We used the wavelet transform to transform the degree of blurring into the spatial frequencies space and use the neural network to learn the relationship between the logical depth and the degree of blurring of all three regions. Then the related to our works are the approach of the machine learning.Then When the number or the dimension of the input data is large, machine learning techniques can be an effective way to solve the problems. In recent years, as a tool to estimate depth maps, machine learning has been receiving increasing interest. Especially supervised learning making use of training data with the ground truth appears highly advantageous to the field of 2-D to 3-D conversion. As well as a set of representative and sufficient training data, good features and suitable classifiers are all essential ingredients for satisfactory results.

With statistical pattern recognition techniques, it is even possible to estimate the absolute average depth given a single image, even thought absolute depth estimation was claimed to be the plus point of multi-ocular cues such as binocular disparity or motion. Torralba and Oliva [8] are interesting the Fourier domain of images and used the global spectral signature of an image to finding the probability of the absolute depth in an image. They applied the EM algorithm (Estimation-Maximization) to find the conditional probability function (PDF) of the mean depth, given the measured image features. The mean depth is then estimated by using the resulting PDF into a mixture model of linear regressions.

If the mean depth of an image can be estimated by machine learning, then following the same way of thinking, it must be possible to estimate the depth value per pixel. A recent work of Saxena et al. applied this idea [9]. This approach is based on capturing the depth and relationship between depths using an Markov Random Fields(MRFs). Firstly, they using a 3-D distance scanner to collect training data, then, the MRF is discriminatively train to predict depth. Thirdly, the model parameters are learned using the maximum likelihood estimator and linear regression based on the training data. Finally, the estimated depth $\hat{d}$ is obtained as the maximum a posteriori estimate (MAP) of the depths $d's$, shown in equation(2.25):

$$\hat{d} = \arg\max_{d}\{P(d|features,\ model\,parameters)\} \tag{2.25}$$

They train by using MRF model for depth estimation from monocular images at multiple spatial scales and tested the model on the real world test set images of forests, campus regions and indoor places. In experimental results, 75% of images/depthmaps were used for training, and remaining 25% for testing, the average error is about 0.132 orders of magnitude.