AUTONOMOUS AGENT NAVIGATION BASED ON TEXTURAL ANALYSIS

BY

RAND C. CHANDLER

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2003

ACKNOWLEDGMENTS

First and foremost I would like to thank Katherine Meiszer, my fiance, for her support, encouragement, and love. I would also like to thank my parents for their advice and guidance throughout my life. Special thanks go to Dr. A. Antonio Arroyo for introducing me to the field of robotics and inspiring me to continue with my graduate education. Thanks go to Michael C. Nechyba for sharing his knowledge and time. Thanks go to Dr. Eric M. Schwartz for the use of his yard for the collection of lawn image data. Thanks also go to Dr. Carl D. Crane III for providing me with a research assistantship, thus allowing this research to reach fruition. Thanks also go to Donald MacArthur and Erica Zawodny for their assistance with the Eliminator mobile robot platform. Finally, thanks go to all the members of the Machine Intelligence Laboratory for all their friendship and support.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

AUTONOMOUS AGENT NAVIGATION BASED ON
TEXTURAL ANALYSIS

By

Rand C. Chandler

May 2003

Chairman: Dr. A. Antonio Arroyo
Cochairman: Dr. Michael C. Nechyba
Major Department: Electrical and Computer Engineering

We present a method for navigating an autonomous agent based on the textures present in an environment. Specifically, the autonomous agent in question is that of a robotic lawn mower. If we can successfully differentiate the textures of the cut and uncut lawn surfaces, then we can track the boundary between them and mow in a pattern as a human would. This dissertation covers the problem of detecting different textures present in an image and then using that information to guide an autonomous robot. The system uses the wavelet transform as the basis to perform texture analysis. The wavelet transform extracts meaningful features from the input images by breaking the image into different frequency subbands. Different subbands will isolate different features in the input image. In this way, we can generate a frequency signature of the image. After performing the wavelet transform, we perform a post-processing stage on these resulting features in an attempt to make them more acceptable to our classifier. These processed features are then grouped into vectors and then classified. The result is a clustered image based on texture.

Once we have the image segmented based on the textures present in the image, we then determine the boundary between them by use of a boundary detection algorithm. In this way we can give a robotic lawn mower the ability to track this boundary and mow as a human would mow. While we avoid the actual implementation of this algorithm on a real platform due to the hazardous nature of lawn mowing in general, we do show how this algorithm can be easily adapted to the task of sidewalk tracking. In this alternate task, the robot tracks the boundary on both sides of the sidewalk, giving the robot the ability to follow the sidewalk. In doing so we not only show the adaptability of our algorithm to another task but also show its implemention on a mobile robot platform.

CHAPTER 1
INTRODUCTION

1.1 Dissertation Statement

Navigating a robot through its environment can be a daunting task depending on the degree of functionality we desire. For instance, we may want to design a robot that simply avoids (while moving randomly) bumping into obstacles in a closed (confined) indoor environment. On the other hand, we may desire a robot that operates outdoors and needs the ability to know exactly where it is in its environment. Both of these situations require that the autonomous agent have the ability to sense its environment for successful navigation tasks. This is typically accomplished through the judicious processing of sensory driven data.

The types of sensors used on an autonomous agent depends on the application. For our indoor robot case, simple infrared emitter and detector sensors may be all that is necessary to avoid bumping into obstacles. However, for a robot that needs to accurately ascertain its location outdoors, some type of positioning system is required. This may require the use of a Global Positioning System (GPS) or a Local Positioning System (LPS). A GPS based system consists of a GPS receiver that provides an approximate location based on timing information received from a constellation of orbiting satellites. A LPS can be realized through the use of beacons placed throughout the operating environment and the use of triangulation to determine position.

Both GPS and LPS systems require the use of externally placed hardware. These systems have some limitations. In a GPS system, the receiver must be able to lock onto a minimum number of satellites in order to calculate its position. This may be a problem if the environment is replete with trees which interfere with the GPS signals. For LPS systems, the beacons need to be powered

by some means--either through batteries which need to be replaced when they get weak or wired directly to a power source such as an AC transformer.

The reliance on an external positioning system could be lessened (or possibly eliminated) if the robot was given the ability to perceive its environment through the use of computer vision. If a robot could recognize objects in its environment, theoretically it should be able to navigate through it. Such a system could be used for a variety of applications. For example, an autonomous lawn mower could determine the difference between cut and uncut grass in a typical lawn and use this information to track the boundary between these two regions.

1.2 Philosophy of Approach

In order to show that a robot can navigate through its environment using computer vision, we will develop a system which allows a robotic lawn mower to mow in a pattern similar to a human performing the same task. A human operator will typically mow a lawn using a plow-like (going back and forth) pattern or by starting at the outside perimeter of the mowing area and then mowing inward toward the center of the mowing area. Even though robotic lawn mowers do exist, they mainly rely on the principle of randomness to mow an area defined by a radio pet fence or other boundary defining system. The autonomous agent has no way of perceiving if it is mowing over a previously cut area or if it is mowing in an area that has not been cut.While this process does work, it is very inefficient in terms of time and energy consumption.

The system that we developed has the ability to recognize textures in an input image of a lawn allowing the mower to recognize the difference between the cut and uncut lawn surface. This gives the agent the potential to track the boundary between these two regions resulting in a net time savings and lower energy consumption.

While the main emphasis of this dissertation focuses on the development of an agent capable of autonomous lawn mowing, we also show how the vision capable system can be adapted to other tasks. One such task is that of navigating a sidewalk. Because of the hazardous nature of lawn

mowing, our algorithm was applied to a sequence of captured image files. We then discuss how this can be extended to a real-time vision equipped agent. This system is also implemented on a non-mower mobile robot platform for the task of sidewalk tracking.

1.3 Thesis Contents

Chapter 2 gives a general overview and motivation for developing a vision-based autonomous lawn mowing agent. In chapter 3, we give a thorough introduction to texture analysis. Chapter 4 develops the algorithm used in our texture analysis system. In Chapter 5, we show how our texture analysis system is applied to the task of autonomous lawn mowing. Experimentally verified results demonstrate the efficacy of the system. In Chapter 6, we demonstrate the adaptability of the texture analysis based system by performing sidewalk tracking on a autonomous mobile robot. This is followed by a discussion on the conclusions and relevant contributions of this research and possible avenues of related future work.

CHAPTER 2
MOTIVATION FOR AUTONOMOUS LAWN MOWING

2.1 Benefits of Autonomous Lawn Mowing

Mowing a lawn can be a tedious and sometimes dangerous task. Any task that is hazardous is well suited to the field of robotics. Robots operate without fatigue, lapses in judgement, or distractions. They also allow humans to have more free time. In recent years, robotic lawn mowers have become a reality. However, their approach to accomplishing the task of mowing is radically different from ours. Most human lawn mower operators mow in a plow-type fashion (going back and forth) or start at the outside perimeter of the mowing area and then work their way inward toward the center of the mowing area in a spiral pattern.

A major problem with current robotic lawn mowers is that they do not have any internal representation of the mowing area. They are usually equipped with sensors (sonar or InfraRed) which are used to avoid obstacles in the mowing area [30]. Autonomous agents usually mow randomly within a defined perimeter for a predetermined period of time not knowing which areas of the lawn have or have not been cut. The expectation is that given a sufficient time interval the lawn will be mostly cut. While this method may be adequate for small lawns with a minimum number of obstacles, this tends to deteriorate as we increase either the size of the lawn or the number of obstacles in a given area. The primary limitations are those of elapsed time and energy consumption. We seek a methodology that would empower an autonomous mower to operate in a manner similar to the typical human performing the same task.

In order to perform the mowing task autonomously, the perimeter of the mowing area needs to be defined by some mechanism that can be sensed by the agent. One way to accomplish this is to use a radio pet fence. This device consists of two parts: the radio fence (a continuous loop of

wire that is buried along the perimeter of the mowing area which is connected to a radio transmitter) and the receiver module (mounted on the mower) [7, 30]. When the autonomous mower approaches the fence, its on-board receiver picks up the RF signal being generated by the fence and turns away, keeping the mower within the defined boundary. This system has built-in disadvantages. The first (obvious) difficulty is the placement of the wire. The wire has to be relocated if the landscape of the mowing area changes. The second disadvantage is the delivery of power to the transmitter module and its relocation in the event of alterations to the boundary. Finally, several areas contained within the mowing area such as island flower beds, ponds, sidewalks, etc. may need to be avoided making this a difficult system to implement under these conditions.

Alternate means are available to define the mowing area boundary. For example, one could deploy sonar beacons in the mowing area, each with its own unique identification number. This forms the basis of a "local" positioning system. Alternatively, the user could employ the use of a Global Positioning System module(s). The homeowner would be tasked with deploying another type of hardware in order to define the mowing boundary. Power supply issues must also be dealt with in order to deploy either system.

Evidently, an autonomous lawn mower robot capable of mowing in a manner similar to its human counterpart requiring no special perimeter-defining hardware would be highly desirable. In order to accomplish this, the mower must have some means by which it can sense the mowing area and determine the difference between cut and uncut grass. Humans sense the mowing area using vision. They accomplish the task efficiently using visual cues to differentiate between cut and uncut regions and adapting their trajectory accordingly.

A digital video camera can be attached to a robotic lawn mower in order give it the ability to "see" the mowing area. The capturing of an image(s) does not provide in and of itself the ability to recognize objects. Object recognition is one of the goals of the field of computer vision and image processing.

Some metric (i.e., color, grey level intensity, etc.) must be employed in order to analyze the acquired image input into the system. Color would seem at first to be a good metric to use. However, an object's color depends greatly on the current lighting conditions. This is especially true for outdoor settings where lighting conditions vary depending on weather conditions (e.g., cloudy vs. sunny) and the position of the sun given the time of day. For example, consider the images shown in Figure 2-1. As can be seen, the colors in identical scenes are quite different based on the time of exposure. Ideally one would like to choose a metric that is less susceptible to changes in lighting conditions. One such metric is texture. The 2nd edition *American Heritage Dictionary* defines texture as "the representation of the structure of a surface as distinct from color of form." Practically all objects (natural or man-made) exhibit some form of texture at the macroscopic level. For instance, the texture of a concrete sidewalk is different from that of grass. This makes texture a potentially useful measure for analyzing the contents of an image.

(a) (b)

Figure 2-1: Two pictures of the same outdoor area under two different lighting conditions. Picture (a) was taken in partly cloudy conditions, and picture (b) was taken under sunny conditions.

2.2 Relevant Previous Work

Several approaches have been proposed for designing an autonomous lawn mower robot. Most of these approaches rely on external hardware to give the mower information about its surroundings. For instance, some systems use radio fences to keep the mower confined within a

defined perimeter, while others use some type of navigation system. Still, others may use a combination of these two methods.

Various methods are also used for analyzing texture [1, 4, 10, 13, 23, 28, 39]. Methods used in the field of texture analysis can be grouped into two main categories: (1) methods that are based on statistical approaches or (2) methods that are based on spatial/frequency approaches [32]. Some statistical methods are co-occurrence matrices, local-linear transforms, and second order statistics [32]. Methods based on spatial/frequency approaches include the use of Gabor filters and wavelet transforms [4, 12, 16, 32].

2.3 Approaches for Autonomous Robot Lawn Mowing

Several approaches have been studied to perform autonomous lawn mowing [7, 21, 22]. These include navigation based systems [7], random based systems [30], and computer vision based systems [21]. Each type will be discussed further in the following sections.

2.3.1 Navigation Based Approaches for Autonomous Mowing

To facilitate autonomous lawn mowing, Palmer [22] and Hakala [7] use external hardware to allow the mower to know where it is in the mowing environment. Specifically, Palmer [22] used an RF positioning system (consisting of a set of stationary beacons) to provide real-time positioning information. The mower used in the experiment did not contain any sensors to aid in obstacle avoidance. In fact, the mower simply followed a pre-programmed path.

In his master's thesis, Hakala [7] employed the use of an active beacon navigation system (sonar based) so that the mower could calculate its position in the mowing area and thus attempt to mow in a plow type fashion. The distances between the beacons were known a priori. To locate its position, the mower calculates its distance from the different beacons based on the time of flight of the sonar pulse. This method is known as trilateration. One main problem was that Hakala's algorithm assumed the mowing area to be planar. Thus, the mower would not have been able to operate

in hilly terrain. This mower did have on-board sensors (infrared emitters and detectors) so that it could detect obstacles in the mowing environment and act accordingly.

2.3.2 Mowing Randomly

Instead of relying on a positioning system to keep track of the areas of the yard that have or have not been cut, some autonomous lawn mowers mow an area randomly while being contained within the perimeter of the mowing area [7, 30]. Hakala [7], in conjunction with his beacon navigation system, also used the idea of an RF (radio frequency) fence to keep the mower contained within the perimeter of the mowing area. The radio fence used was actually an off-the-shelf electric "pet fence" designed for use with cats or dogs. The receiver (normally worn on the collar of the animal) was placed on the mower thus giving it the ability to stay within the perimeter defined by the fence.

The commercially available Robomow [30] uses an RF fence to keep itself contained within the mowing area. In this case, the wire is not buried but secured to the ground using plastic stakes. When the mower starts out, it locates and then follows the RF wire mowing the perimeter of the yard; it then mows randomly inside the area defined by the RF fence. According to the manual, it mows an area of approximately 2500 sq. ft. (about the size of a tennis court) in about 2.5 hours [30]. The mower does contain bump sensors for obstacle avoidance.

As stated earlier, the main problem with mowing randomly is that the mower has no internal knowledge of what it has or has not cut. This is the least efficient method in terms of the elapsed time and energy consumption. However, this is an easy algorithm to implement.

2.3.3 Mowing with the Use of Computer Vision

Ollis [21] used computer vision to guide an automated harvester robot. The crop that was chosen to harvest was alfalfa. Alfalfa is a tall grass, which when cut yields a quite noticeable boundary between the cut and uncut boundaries. Ollis used a color based discriminant function in order to locate the boundary between the cut and uncut areas of alfalfa. The discriminant function

(he ultimately concluded this worked the best overall) consisted of fitting a best fit step function along each given scan line in the input image. Because the conditions can vary between different fields of alfalfa he implemented a dynamic means of tuning the boundary algorithm to the local conditions to improve the robustness of the algorithm. His system produced impressive results. As an aside, he also showed how his algorithm performed in discriminating between plowed vs. unplowed snow and compacted vs. uncompacted trash.

2.4 Methods Used for Texture Analysis

Several methods have been used in the analysis of texture. These methods can be broken down into two main categories: (1) statistical based methods and (2) spatial/frequency based methods. Statistical based approaches include the use of Markov Random Fields [13, 23, 24, 31], co-occurrence matrices [25, 32], region competition [39], circular-mellin features [28], and second order grey level statistics [25]. Spatial/frequency based approaches include the use of Gabor filters [4, 9, 16] and Wavelet transforms [10, 11, 12].

2.4.1 Statistical Based Approaches

Unser [32] reports that most statistical based methods are best suited for the analysis of micro textures. This is due to the fact that they are restricted to "the analysis of spatial interactions over relatively small neighborhoods." In recent years, the spatial/frequency approach to texture analysis has grown in popularity. This is due in part to research that suggests the existence of an internal spatial/frequency representation in the human visualization system [1, 12].

2.4.2 Spatial/Frequency Based Approaches

These approaches include Gabor filters [4, 16] and wavelet transforms [10, 33]. Gabor functions consist of sinusoids of a particular frequency that are modulated by a Gaussian envelope. They (like their wavelet counterparts) provide localized space-frequency information of a signal [5]. Two dimensional Gabor functions consist of a sinusoidal plane of some orientation and frequency modulated by a Gaussian envelope. However, Gabor filters are computationally very

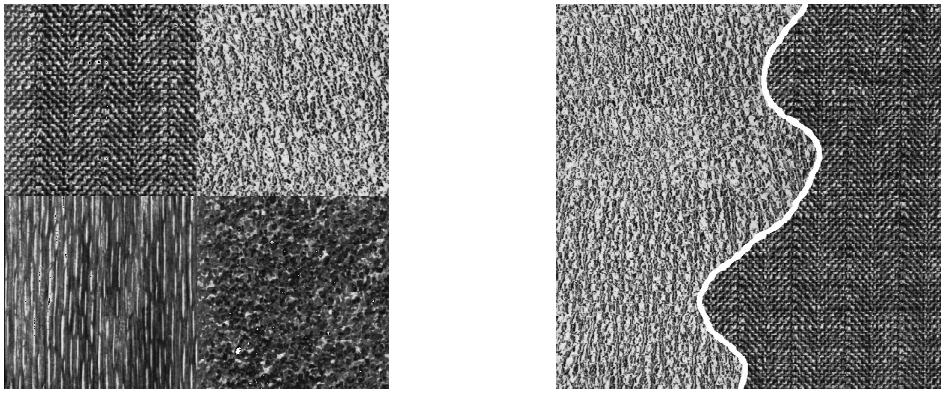
inefficient [10]. Also, the outputs of Gabor filter banks are not mutually orthogonal which could result in correlation between texture features [32].

Wavelet transforms can be thought of as a multi-resolution decomposition (multiple scale signal view) of a signal into a set of independent spatially oriented frequency channels [1, 35]. In its simplest form, the wavelet can be thought of as a bandpass filter. To begin a wavelet analysis, one starts with a prototype (or mother) wavelet. High frequency (contracted) versions of the prototype wavelet are used for fine temporal analysis while low frequency (dilated) versions are used for frequency analysis [6]. Unlike the Gabor functions described earlier, fast algorithms exist for wavelet decompositions. If one chooses wavelets that are orthonormal, then correlation between different wavelet scales is avoided. Furthermore, there is experimental evidence that suggests that the human vision system supports the notion of a spatial/frequency multi-scale analysis, making wavelet transforms an ideal choice for texture analysis [12, 32].

CHAPTER 3
RELATED TOOLS FOR TEXTURE ANALYSIS

3.1 Overview

Texture in an important quality when analyzing the contents of an image. Practically every object (natural or man-made) exhibits some form of texture at the macroscopic level. Figure 3-1(a) shows some examples of different textures taken from the Brodatz texture database. Thus, the use of texture information would be a practical means to segmenting the objects in an image. Texture segmentation is the process by which differently textured regions in the input image are accurately partitioned based on the borders between the different textures. Figure 3-1(b) shows an example of texture segmentation.



(a) (b)

Figure 3-1: Some examples of texture and texture segmentation. (a) Four sample Brodatz textures. From upper left going clockwise, D17: herringbone weave, D24: pressed calf leather, D68: wood grain, D29: beach sand. (b) An example of texture segmentation.

Image segmentation could also be carried out by segmenting the objects in an image based on their color. However, an object's color is highly dependent on the type of light the object is illuminated with. For example, an object's color may appear slightly different when viewed outdoors in sunlight versus indoors under fluorescent light. Given that this research involves designing an

autonomous lawn mower that will work outdoors where lighting conditions are subject to change (due to clouds, position of the sun given the time of day, etc.) color is not a suitable metric. Color could be an appropriate metric to use in image segmentation if it were carried out in a controlled lighting environment.

To perform the texture analysis portion of this research, the wavelet transform will be used. As mentioned earlier, this is a spatial/frequency based approach as opposed to a statistical based methodology. This approach has been chosen for several reasons: (1) fast algorithms exist for wavelet decompositions, (2) they provide a multiresolution (multiple-scale) view of a signal, (3) wavelets that are orthonormal posses orientation selectivity, and (4) there is experimental evidence that suggests that the human vision system supports the notion of a spatial/frequency multi-scale analysis [12, 32]. In the following section, we present an in-depth discussion of the wavelet transform.

3.2 The Wavelet Transform

Wavelet transforms can be thought of as a multi-resolution decomposition (multiple scale signal view) of a signal into a set of independently spatially oriented frequency channels [1, 35]. Alternatively, this amounts to looking at a signal at different "scales" and then analyzing it with various "resolutions." The wavelet itself can be simply thought of as a bandpass filter. To begin a wavelet analysis, one starts out with a prototype wavelet, the "mother" wavelet from which all other wavelets are constructed. For instance, high frequency (contracted) versions of the prototype wavelet are used for fine temporal analysis while low frequency (dilated) versions are used for fine frequency analysis [32, 33].

To illustrate this point, it is useful to contrast wavelet transforms with Fourier transforms. Fourier transforms use as their basis functions sines and cosines. These functions have infinite extent, which means that any time-local information (such as an abrupt change in the signal) is

spread over the entire frequency axis [35]. Therefore, Fourier transforms are not localized in space, whereas wavelets transforms are. Figure 3-2 helps to illustrate this point.

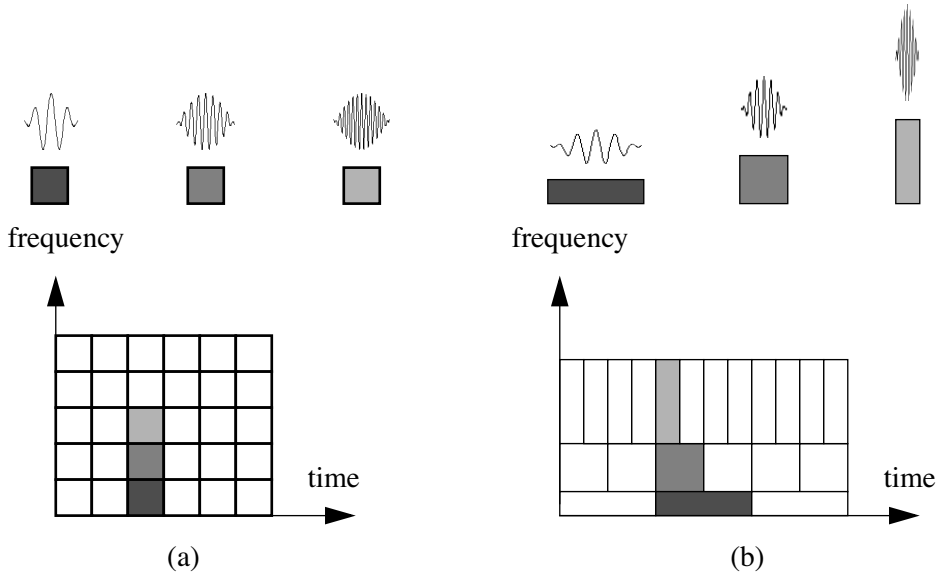


Figure 3-2: Contrast between the Fourier transform and wavelet transform. (a) Basis functions and time-frequency resolution of the Fourier transform. (b) Basis functions and time-frequency resolution of the wavelet transform.

Figure 3-2(a) shows the basis functions and time-frequency resolution of the Fourier transform. This is accomplished by using a square wave as a window to truncate the Fourier basis functions. As can be seen, the resolution of analysis is the same for all locations in the time-frequency plane. Unfortunately, this is a severe limitation in the Fourier transform.

On the other hand, Figure 3-2(b) shows the basis functions and time-scale resolution of the wavelet transform. It is important to point out that the basis functions are not limited to just two as in the case of the Fourier transform [6]. Many basis functions exist for the wavelet transform (e.g., Mexican hat wavelet, Daubechies, Lemarie-Battle, Haar, etc.). As can be seen, the size of the window is allowed to vary in a wavelet transform. This allows one to trade resolution in time for resolution in frequency [35]. If one wants to perform a fine temporal analysis (e.g., looking for an abrupt change in a signal) one would use high frequency (short) basis functions. Conversely, if one

wants to perform a detailed frequency analysis, low frequency (dilated) basis functions would be used. These short and dilated basis functions are obtained from the prototype wavelet.

3.2.1 The Continuous Wavelet Transform

The continuous wavelet transform (CWT) of a function *f(t)* with a wavelet $\psi(t)$ is defined as

$$W(a, b) = \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{a}}\psi^*\left(\frac{t-b}{a}\right)dt \tag{3-1}$$

where *a* and *b* are real variables [6].The variable *a* is referred to as the dilation variable. Depending on its value, it dilates or contracts the function $\psi(t)$. On the other hand, the variable *b* represents a time shift. Equation 3-1 can be written more compactly if one defines $\psi_{a,\,b}(t)$ as

$$\psi_{a,\,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{3-2}$$

Combining Equations 3-1 and 3-2, the CWT can be written more compactly as

$$W(a, b) = \int_{-\infty}^{\infty} f(t)\psi^*_{a,\,b}dt \tag{3-3}$$

Because the CWT is generated through the use of translations and dilations (or contractions) of the wavelet $\psi(t)$, $\psi(t)$ is generally referred to as the prototype or *mother* wavelet [15, 34]. It is the wavelet from which all resulting wavelets are generated. As stated in the previous section, unlike the Fourier transform, the wavelet transform offers both time and frequency selection.

If the variables *a* and *b* are allowed to be continuous, then the wavelet transforms that are generated are highly redundant [35]. The same can also be said in regards to the Fourier transform. To alleviate this problem, the CWT is usually evaluated on a discrete grid on the time-scale plane (see Figure 3-2). This yields a discrete set of continuous basis functions. A set of basis functions that have no redundancy form a orthonormal basis. However, an orthonormal basis does not guarantee that there will be no redundancy. It is possible to design a prototype wavelet in which translated and scaled versions of this wavelet form an orthonormal basis.

3.2.2 The Discrete Wavelet Transform

When dealing with the discrete wavelet transform, the notion of scale and resolution is still applicable. However, instead of modifying parameters to control the scale and resolution of the prototype wavelet, either a high or low pass filter is used to change the resolution of the signal while subsampling of the signal is used to change its scale [29]. Equation 3-4 illustrates this point. Given an input sequence $x(n)$, a lower resolution is derived by convolving it with a discrete-time low pass filter (LPF) having an input response $g(n)$. Following Nyquist's rule, one can then subsample by a factor of two, thus doubling the scale. This produces a signal, $y(n)$, having half the resolution and twice the scale of the original.

$$y(n) = \sum_{k=-\infty}^{\infty} g(k)x(2n-k) \tag{3-4}$$

This process can then be repeated on the output $y(n)$ to produce what is known as a multi-resolution signal analysis (MRA) [29]. What has not been discussed thus far is the corresponding high pass filter $h(n)$. Filtering the input signal $x(n)$ with $h(n)$ (followed by a downsampling by two) yields the high frequency "detail" information in the signal.

3.2.3 Computing the One-dimensional DWT

As alluded to in the previous section, the computation of the DWT involves convolution with discrete time low and high pass filters. Figure 3-3 shows the first step in computing the one-dimensional DWT. As can be seen, the first step involves convolving a discrete-time, low pass filter (LPF) at intervals of two with an input vector x. Shifting the filter in increments of two has the effect of downsampling the input vector by a factor of two. Alternatively, one could first perform the convolution by shifting filter by increments of one and once this is complete, then

downsampling the result by a factor of two. The results of these convolutions are placed in the left half of the *w* vector.

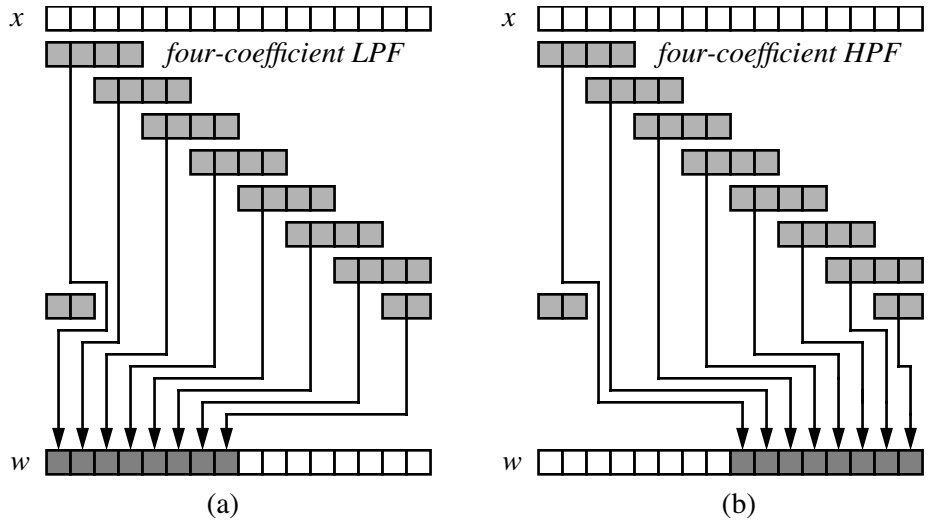


Figure 3-3: Computing the one-dimensional wavelet transform. (a) Step 1: filtering using a four coefficient LPF (in increments of two) on an input vector *x*. (b) Step 2: filtering using a four-coefficient HPF (in increments of two) an input vector *x*. The result of these operations are placed in the *w* vector.

The second part involved in computing the one-dimensional DWT involves convolving the input vector *x* with a discrete-time high pass filter (HPF) in increments of two. As before, the effect of shifting the filter by a factor of two has the result of downsampling the result by a factor of two. The results of these convolutions are then placed in the right half of the *w* vector.

The result of these two operations is the following: the left portion of the *w* vector represents the coarse approximation (the resolution has been halved) of the input vector *x* and the right hand portion localizes the high frequency components (added detail) of the input vector *x* [17, 29]. The result so far is known as a first-level DWT on *x*. However, one need not stop here. The same procedure can be reapplied to the left hand (low pass filtered) portion of the resultant *w* vector. This result would produce a second-level DWT. This process can then be re-applied again and again, each time working with the low pass portion of the previous generated vector as input to the next level. Figure 3-4 illustrates this process.

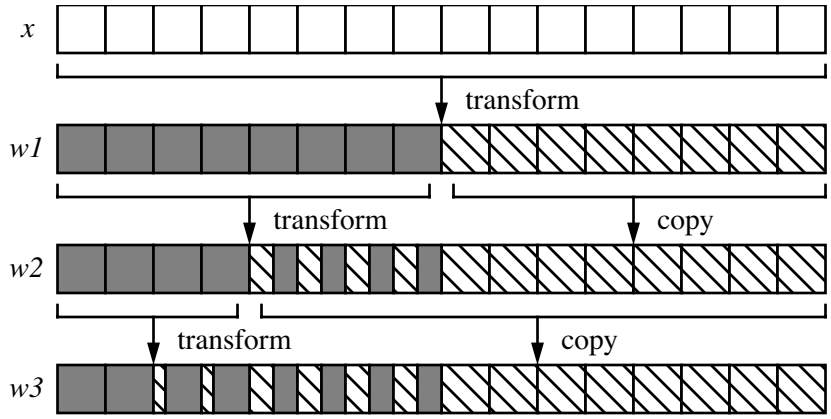


Figure 3-4: Three level DWT of the input vector x. It is important to point out that for the second level DWT, $w2$, the right half is a copy from the previously generated level. For the third level DWT, $w3$, only the first four entries were generated as a result of the transform from the previously generated level.The rest of the entries remain unchanged from the level above.

As can be seen in Figure 3-4, the second level DWT is performed only using the left hand portion of the previously generated transform, $w1$. The right hand portion is simply a copy of the coefficients from the previously generated level. To compute the third level DWT, $w3$, only the first four values of the second level transform are used in the calculations. As in the previous case, the remaining coefficients are copied from the previously generated level, $w2$.

It is useful to point out what the various regions of the third level DWT represent. The first two coefficients are the result of three consecutive low pass filters on the input signal x (downsampling by a factor of two between each level). Coefficients three and four represent the result of two passes of a low pass filter and then with a high pass one. This is indicated by the shaded portions of the "boxes" that represent the coefficients of the third level DWT. Continuing with this analysis, coefficients five through eight are the result of a low pass filter followed by a high pass one. Finally, the remaining coefficients are the result of high pass filtering the input signal x.

From the analysis point of view, the lowest frequencies will be isolated in the first four coefficients of the third level DWT. Moderate frequencies will be isolated in the mid-portion of the

transform (namely coefficients 5 through 8). High frequencies are then isolated in the right half of the transform.

Figure 3-5 shows the DWT structurally from a filter bank point of view. The symbols *H* and *G* represent the low and high pass discrete-time filters respectively. Also shown in the figure is the downsampling by a factor of two that occurs when computing the DWT.

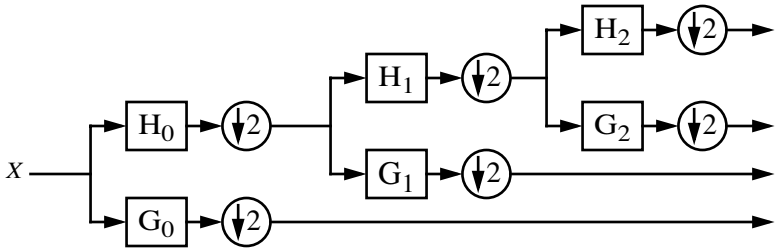


Figure 3-5: Structural (filter bank point of view) diagram of the DWT. The circles containing the down arrow and the number 2 represent downsampling by a factor of two.

3.2.4 The Balanced Tree Wavelet Decomposition

As can be seen in Figure 3-5, only the output of the low pass filter is decomposed further. This is by the definition of the DWT. Also note that the three-level DWT shown in Figure 3-5 results in one coarse approximation (filter output H_2) and three detail approximations (filter outputs G_0, G_1 and G_2) [27]. This is not the only type of wavelet decomposition that exists. Figure 3-6 shows a wavelet decomposition (filter bank point of view) that allows further decompositions on the outputs of the high pass filters. Such a decomposition is referred to as a "balanced" tree wavelet decomposition [27].

A balanced tree wavelet decomposition imposes the requirement that the underlying wavelet used be orthonormal [10, 27]. However, this type of decomposition does allow for more flexibility. Instead of being confined to only decomposing the output of the LPF, the output of the HPF is also decomposed further. The basis functions that result from such a decomposition are commonly referred to as wavelet packets [10, 27]. Given this, this structure is know as the discrete wavelet packet transform (DWPT).

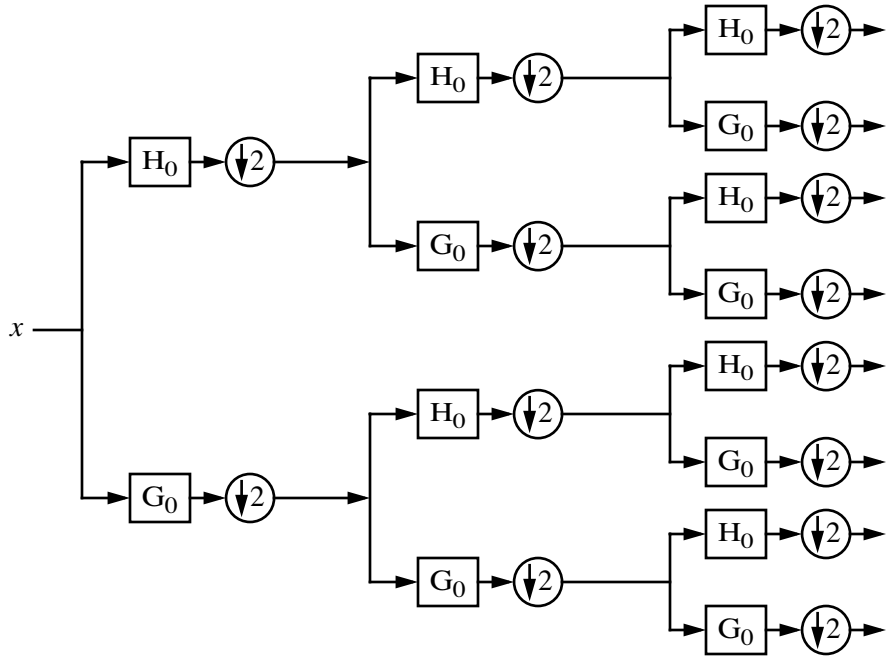


Figure 3-6: Three level, balanced tree, wavelet decomposition.

3.2.5 Arbitrary Wavelet Decompositions

Alternatively, instead of performing the full DWPT, an intermediate form can be chosen. Wavelet subbands that show very little activity (low frequency content) could be discarded resulting in a pruned tree. Only those particular subbands that result in a level of high activity would be chosen for further wavelet decomposition. Figure 3-7 shows an example of this selective type of wavelet decomposition.

All of the methods of wavelet transform decompositions (DWT, DWPT, etc.) discussed thus far subsample the output of the low and high pass filters by a factor of two. This results in decompositions that are not translation invariant [11, 32]. However, when performing the process of texture analysis, a method that is translation invariant is highly desirable. One way to accomplish this is to use an overcomplete wavelet decomposition [11].

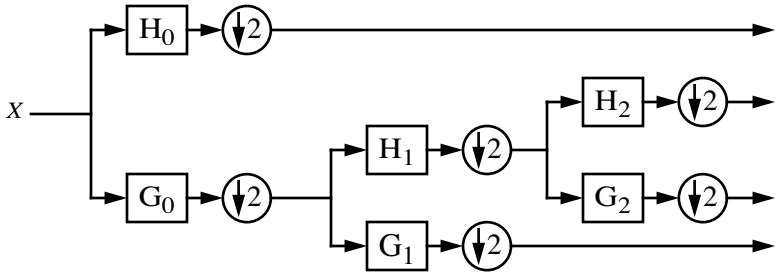


Figure 3-7: Arbitrary discrete wavelet packet transform (DWPT) decomposition.

To perform an overcomplete wavelet decomposition, the output of the high and low pass filters is not subsampled. This relates to the mathematical concept of a frame and is thus known as a discrete wavelet frame (DWF) decomposition. Figure 3-8 shows a discrete wavelet frame decomposition. It should be pointed out that an overcomplete wavelet frame decomposition can also be applied to the balanced tree decompositions (known as a discrete wavelet packet frame, DWPF) and arbitrary wavelet decompositions.

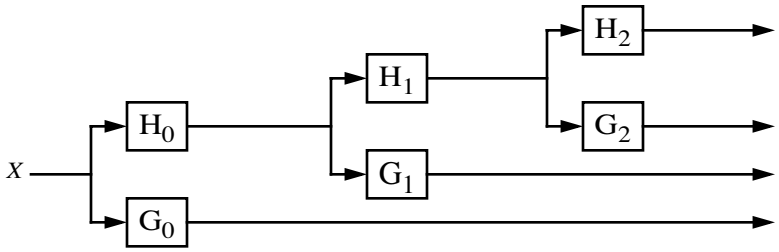


Figure 3-8: Discrete wavelet frame decomposition

3.2.6 The Two-Dimensional Wavelet Representation

In order to deal with 2D images, a 2D wavelet representation needs to be employed. This amounts to a tensor product extension which can be viewed as a cross product of the low and high pass filters [10, 11]. Thus, four types of 2D filters (basis functions) result from the cross products of the high and low pass filters. This operation can be carried out by first applying the 1D wavelet transform along all of the rows in the image. Next, the 1D wavelet transform is applied to the columns of the resulting transform from the first step.

This results in four distinct types of 2D filters or basis functions. Specifically, these are: LL, LH, HL, and HH. For example, the LH filter is the result of first low pass filtering on the rows of the image followed by high pass filtering on the columns of the results of the first step. Figure 3-9 shows this process graphically for performing the level one, 2D wavelet transform. It should be noted that Figure 3-9 applies to the DWT and balanced tree (DWPT) decompositions since the output of the high and low pass filters are subsampled by a factor of two.For an overcomplete wavelet representation (DWF or DWPF), no subsampling would occur at the output of the filters. This process, like the one-dimensional transform case, can be re-applied to the previous level. However, this time the 2D wavelet transform is re-applied to the output of one of the 2D filters. Typically this is reapplied to the LL filter.

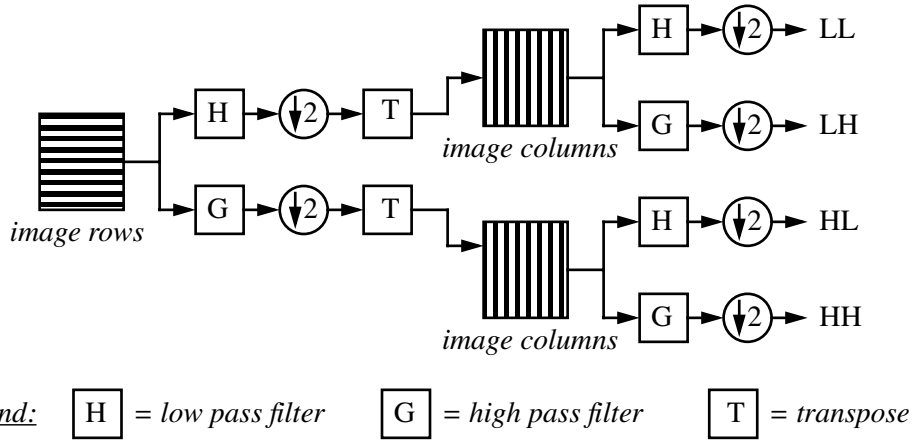


Figure 3-9: Graphical representation of the one-level 2D wavelet transform. First the one-dimensional DWT (or DWPT) is applied to the rows of the input image. The resulting data is then transposed and the one-dimensional DWT (or DWPT) is then applied to the columns. This results in four distinct 2D filters.

These 2D filters exhibit orientation selectivity if certain typed of basis functions are used. The LH filter tends to preserve horizontal features in the image, the HL filter tends to preserve vertical features, and the HH filter tends to preserve diagonal features in the image. The LL filter results simply in a low passed version of the input image. Figure 3-10 represents this graphically. Essentially this can be viewed as partitioning the input image into a 2D frequency plane.

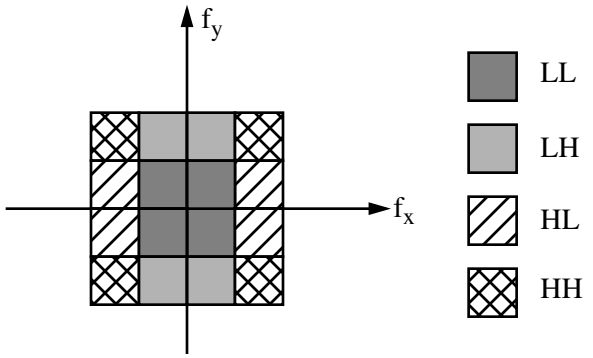


Figure 3-10: Frequency plane partitioning as a result of the two-dimensional wavelet transform.

3.2.7 Filter Selection

Up till now, there has been no mention on what types of discrete-time filters (prototype wavelets) are appropriate for performing a wavelet analysis. Many different types exist (Haar, Mexican hat, Morlet, Daubachies, LeMarie-Battle, etc. [27]), each appropriate for a particular application. For example, certain filter types are orthonormal which result in wavelet decompositions in which there is no correlation (no redundancy) between scales [12]. This property is particularly useful when designing perfect reconstruction filter banks and when performing a multiresolution analysis.

Equally important, once a filter is specified, is how the low and high pass versions of this filter are generated for the different levels in the wavelet transform. The methods differ for filter generation depending whether or not one is using the standard DWT or a DWPF transformation. These subjects will be addressed in the following subsections.

3.2.8 Filter Choice

To perform texture analysis, the filter type one chooses should have certain properties. First and foremost, the filter should be orthonormal. Thus when performing the wavelet transform, the input signal will be decomposed onto an orthonormal basis and correlation between scales (redundancy) can be avoided [12]. Secondly, the underlying filter should be symmetric. If the filter if symmetric then no phase distortion will occur. This results in the preservation of the spatial

localization (no time shifting will occur) of the resulting wavelet coefficients after processing [10, 11, 12]. This property is highly desirable when performing texture segmentation where the boundaries between two different textured regions need to be accurately determined. Third, the filter should be of such a type that when a two dimensional wavelet analysis is performed (using this filter as the prototype filter) orientation selectivity results. This should augment the reliability of the texture segmentation process, especially when determining the boundary between the cut and uncut lawn surface. Given the above criteria, the LeMarie-Battle wavelet/filter has been chosen for use when performing a wavelet analysis.

3.2.9 Filter Generation

In this section, we focus on how to generate the resulting filters for the filter bank from the prototype filter. This prototype filter is usually specified in terms of a low pass, discrete-time filter, h. The complimentary high pass filter, g, is obtained by a shift of the low pass filter. This is shown in Equation 3-5 below. Alternatively, Equation 3-6 shows this operation in the frequency domain.

$$g_o(n) = (-1)^n h_o(n) \tag{3-5}$$

$$Go(\omega) = Ho(\omega + \pi) \tag{3-6}$$

Because we will use a discrete wavelet packet frame (DWPF) wavelet representation as the first step in performing texture analysis, the high and low pass filters do not remain the same for each level of the transform as is the case for the DWT representation. In the case of a DWT analysis, the same low and high pass versions of the filters are used for every level in the transform followed by a downsampling of two at the output of the filters.

To generate the filters for the different levels for the DWPF representation, the following formulas are used to generate the filters for the different levels:

$$h_{l+1}(n) = [ho(n)]_{\uparrow 2^l} \tag{3-7}$$

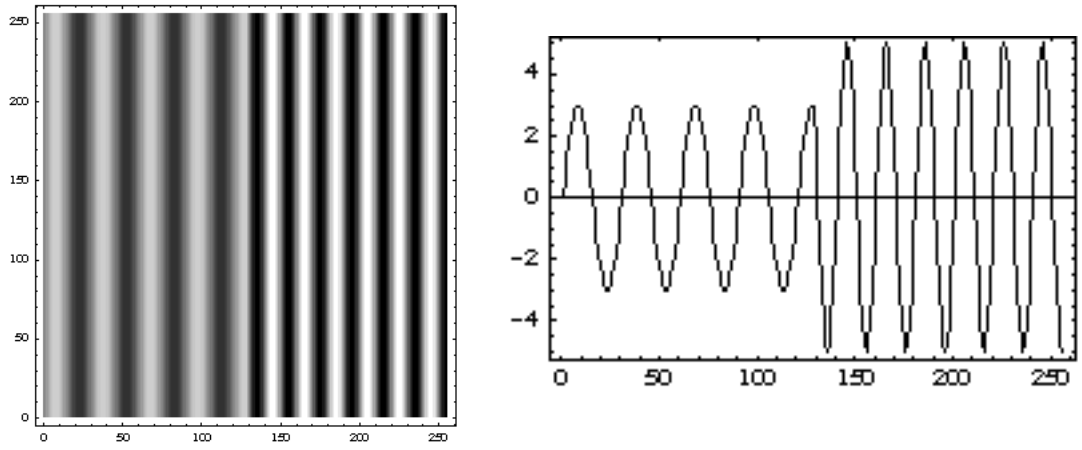
$$g_{l+1}(n) = [go(n)]_{\uparrow 2^l} \tag{3-8}$$

Equations 3-7 and 3-8 can be interpreted as follows: we insert the appropriate number of zeros between the filter taps based on the level of the DWPF we are interested in computing [14]. This is indicated by the up arrows followed by the number 2 to the *lth* power.This amounts to an expansion of the filter (dilation) in the time domain, but a contraction in the frequency domain.

3.3 Feature Enhancement

Feature enhancement is an essential part of the texture analysis process. This is due to the fact that the data (wavelet coefficients) that is generated as a result of the wavelet transform are unsuitable for use in clustering algorithms [10, 11, 20, 36]. Thus, the feature enhancement process can be viewed as an interface layer between the wavelet transform and clustering algorithm. Whichever method we choose to perform the feature enhancement process, it should produce "features" that are acceptable to the chosen classifier (in the classification stage) but not at the expense of decreasing the information content of the features themselves [20]. In addition, we would expect the feature enhancement algorithm to be consistent among the pixels within a class but posses a high degree of discernability between classes.

Typically the feature enhancement process involves the following steps: (1) filtering, (2) applying a nonlinear operator, and (3) applying a smoothing operator [25, 26]. To illustrate the process of feature enhancement, we present the following example. For the purposes of this example, a "synthetic" texture was generated as can be seen in Figure 3-11(a). This texture was created by appending two sinusoids of different frequency. Specifically, the sinusoid on the left is of low frequency and the one on the right is of high frequency. Figure 3-11(b) shows a horizontal slice through the image showing the two sinusoids.



(a) (b)

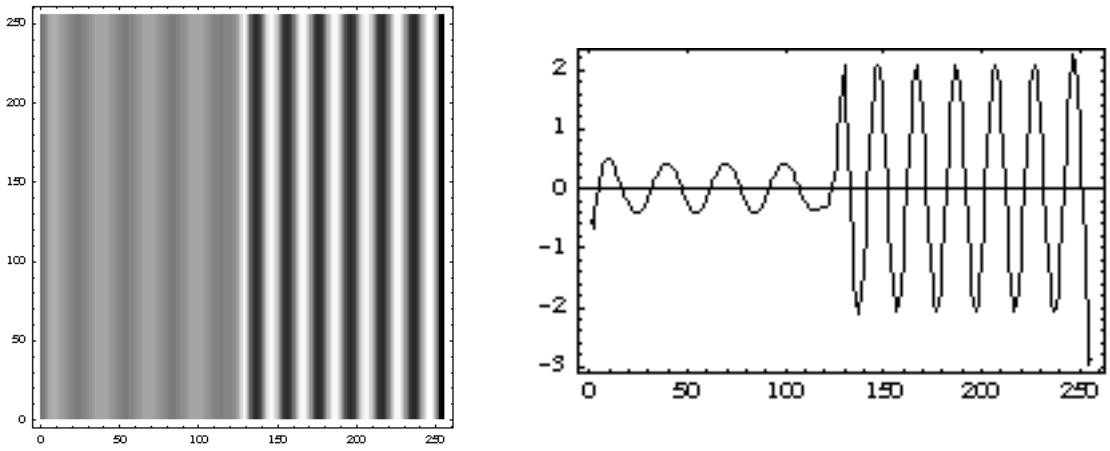
Figure 3-11: Generation of a synthetic texture. (a) Synthetic texture generated by the combination of two sinusoids. (b) A horizontal "slice" through the image shown in (a) showing a low frequency sinusoid on the left and a high frequency sinusoid on the right.

Our first step in this process is to filter the image. For the purposes of this example, all of the rows of the image will be high pass filtered using a level 2 high pass filter obtained by using a LeMarie-Battle prototype filter. The next figure, Figure 3-12, shows a density plot of the result of filtering the image with the high pass filter. Also shown is a horizontal cross section of the image. As expected, the high frequency sinusoid remains (dominant) with the remnants of the low frequency sinusoid visible as well. The main point that can be inferred from this figure is that even though we filtered the image, the result is still unacceptable for any type of classifier.

In order to obtain a result that is capable of being of being useful to a classifier, it must be transformed. As stated earlier, a non-linear operator such as taking the magnitude or squaring the data, can be applied to the results. However, the result of taking the magnitude or squaring the data alone may still yield a result that is unacceptable. Figure 3-13 illustrates this point.

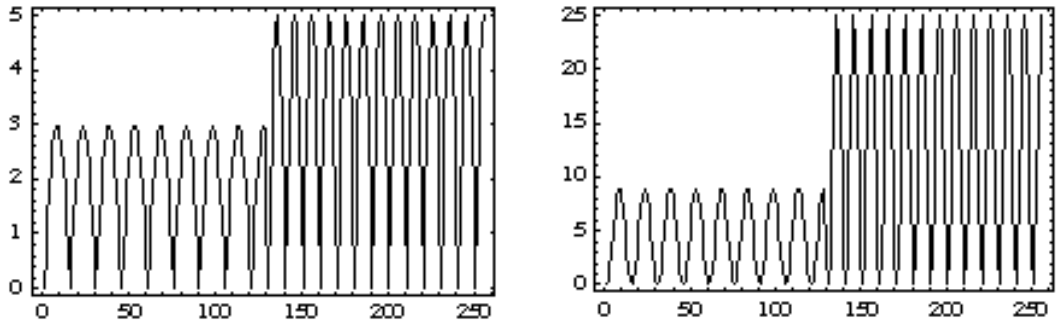
As one can see, these results still are not suitable for clustering. The next step in the process of feature enhancement is to smooth this data by means of low pass filtering. However such a procedure may result in the blurring of texture boundaries. An alternative to this is to apply the concept of an envelope detector to the data [10, 11]. An envelope detector can be thought of a peak

detector. Figure 3-14 shows the result of applying an envelope detector to the filtered data shown in Figure 3-12.



(a) (b)

Figure 3-12: Result of high pass filtering on the synthetic texture shown in Figure 3-11. (a) Density plot of the result of high pass filtering on the synthetic texture. (b) Horizontal slice through the density plot. Note that even filtering alone does not yield results that are suitable for classification.



(a) (b)

Figure 3-13: Horizontal slice result for the application of a non-linear operator applied to the filtered synthetic texture. (a) Result of taking the magnitude. (b) Result of squaring the data.

As can be seen, the envelope (thick line above the peaks) tracks the magnitude of the signal. The envelope is generated by calculating the maximum value between two zero crossing points and assigning that value to all points within that interval. A two-dimensional plot using the data generated as a result of the envelope detector is shown in Figure 3-15. As one can see, we now have a result in which the resulting data could be reliably clustered.

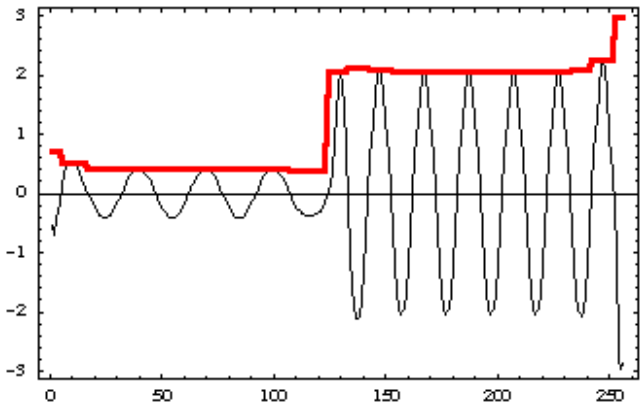


Figure 3-14: Result of applying an envelope detector to the data presented in Figure 3-12. The envelope is shown by the thick line above the peaks.

This same process can be applied to real images as well. However, the wavelet transform will be used to filter the input image instead of a simple high pass filter as was the case in the previous example. One important thing to note should be the fact that a vertical or horizontal envelope detector may be applied depending on the orientation selectivity of the particular wavelet basis function.

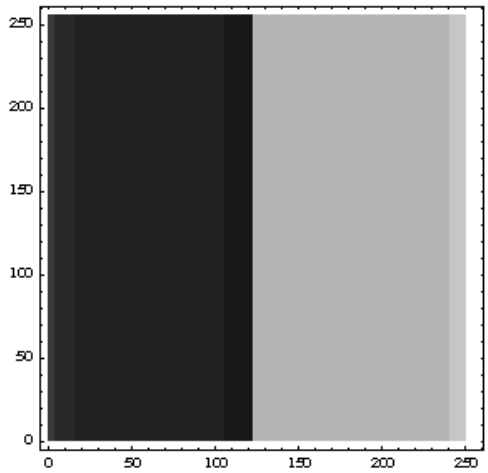


Figure 3-15: Result of applying envelope detection to the results shown in Figure 3-12. This data is capable of being clustered. It should be noted, however, that the vertical edges of this figure vary somewhat slightly due to the abrupt change (starting and ending) of the signal.

3.4 Clustering

Clustering is the process of finding natural groupings within a set of data. The data contained within a particular group (or cluster) should posses a high degree of similarity to one another [3]. For instance, when talking about an image, one might want to cluster the data based on the different colors of the objects present in the image. With respect to this research, our goal at this stage is to cluster the data generated by the feature enhancement phase. Clustering the data from the feature enhancement phase will result in an segmented image based on the different textures present in the image.

Data cannot be clustered without first classifying it. In order to classify the data, we must have some means of comparing an unknown element of data with something that is known. This brings us to the topic of statistical modeling. In statistical modeling, we are interested in learning the statistical properties of a distribution of data, so that input vectors, x, are mapped to probabilities $P(\dot{x})$ [18, 38].To classify an object, we would follow this basic procedure:

1. Collect sample data $\mathrm{X}^i = \{\mathrm{x}^i_j\}$,$j \in \{1, 2, \ldots, n_i\}$, for each class ω_i , $i \in \{1, 2, \ldots, k\}$.
2. Extract meaningful features from the sample data collected.
3. Build statistical models λ_i of the distribution of features for each class ω_i .
4. For an unknown input x, or set of inputs $\mathrm{X} = \{\mathrm{x}_j\}$,$j \in \{1, 2, \ldots, n\}$, perform the same feature extraction as in step 2 and evaluate: $P(\lambda_i|\mathrm{X})$, $\forall i \in \{1, 2, \ldots, k\}$.
5. Classify the unknown data X as the class with the highest probability.

Several types of clustering algorithms exist such as K-means, C-means, fuzzy C-means, hierarchical trees, etc. [2, 3, 8, 37]. For the purposes of this research, the vector quantization (VQ) method will be used. Specifically, the LBG (Linde, Buzo, and Gray) vector quantization algorithm will be used [19]. We first discuss the VQ algorithm in a generic sense and then discuss the LBG algorithm in detail.

3.4.1 Vector Quantization

To introduce the concept of vector quantization, assume that you are given a data set $X = \{x_j\}$, $j \in \{1, 2, \ldots, n\}$, of n d-dimensional vectors. The vector quantization problem requires that we find a set of prototype vectors $Z = \{z_i\}$, $i \in \{1, 2, \ldots, L\}$, $L \ll n$, such that the total distortion D is minimized [19]. The distortion equation is shown in Equation 3-9.

$$D = \sum_{j=1}^{n} \min_{i} dist(x_j, z_i) \tag{3-9}$$

The expression, $dist(x_j, z_i)$, in Equation 3-9, is a distance metric usually (but not always) given by the Euclidean distance, $\|x - z\|$. The Z vectors are known as the VQ codebook. In other words, our original data set is now described entirely by the codebook. In this sense, the application of the VQ algorithm results in a reduced data space. Instead of having to work on a large set of data for comparison, we now only have to work with the VQ codebook. Thus, we have generated a statistical model of our data.

3.4.2 The LBG Vector Quantization Algorithm

The LBG vector quantization algorithm addresses the problem of VQ codebook initialization by iteratively generating codebooks $\{z_i\}$, $i \in \{1, \ldots, 2^m\}$, $m \in \{0, 1, 2, \ldots\}$, of increasing size. This is a main advantage over the traditional K-means algorithm in that no parameters are required for initialization [19]. Presented on the next page is the LBG vector quantization algorithm.

Now that we have presented the LBG vector quantization algorithm, we will illustrate how this algorithm works by a simple example. Suppose we have an RGB (red, green, and blue) image in which we want to cluster (or segment) the objects in the image based on the colors present in each object. Our first task is to obtain some training data for each individual object we are trying to classify. Figure 3-16 shows the training data for each object we are trying to recognize in an unknown image.

The LBG vector quantization algorithm:

1. Initialization: Set $L = 1$, where L is the number of codes in Z, and let z_1 be the centroid (or mean) of the data set X.

outer loop

2. Splitting: Split each code in the VQ codebook, z_i, into two codes $\{z_i, z_{i+L}\}$, where $z_i = z_i + \tilde{\varepsilon}$ and $z_{i+L} = z_i - \tilde{\varepsilon}$, $\forall i$ where $\tilde{\varepsilon} = \varepsilon\{b_1, b_2, \ldots, b_d\}$. The variable ε is some small number, typically 0.0001. The b_k can be set to all 1's or some combination of ± 1. At this stage, because the number of VQ codes in Z has doubled, let $L = 2L$ and:

inner loop

   1. Classification: Classify each x_j into cluster or class ω_i such that $dist(x_j, z_i) \leq dist(x_j, z_l)$, $\forall l$.

   2. Codebook update: Update the code for every cluster ω_i by computing its centroid,

$$z_i = \frac{1}{n}\left(\sum_{x_j \in \omega_i} x_j\right)$$

      where n_i is the number of vectors x_j in cluster ω_i.

   3. Termination #1: Stop when the distortion D has decreased below some threshold level or when the algorithm has converged to some constant level of distortion.

3. Termination #2: Stop when L is the desired VQ codebook size.

(a) (b)

Figure 3-16: Training data for our example. (a) Picture of flowers. (b) Picture of leaves

Because the image we are dealing with is in RGB format and given the fact that we will be using the R, G, and B features of each pixel to cluster the data, the dimensionality, $d = 3$. Given this, our data set will consist of vectors of the form $\tilde{x}_j = \{x_{jr}, x_{jb}, x_{jg}\}$, $j \in \{1, 2, \ldots, n\}$ where n

represents number of vectors in the *combined* training set. Figure 3-17 below shows the training data in Figure 3-16 plotted in RGB color space.

As can be seen in Figure 3-17, the data from each training object, tends to fall into distinct clusters in the RGB color space. We will now apply the LBG vector quantization algorithm to this combined data set. The intermediate results of each iteration of the outer loop (where the clusters are split) are shown in Figure 3-18 below.

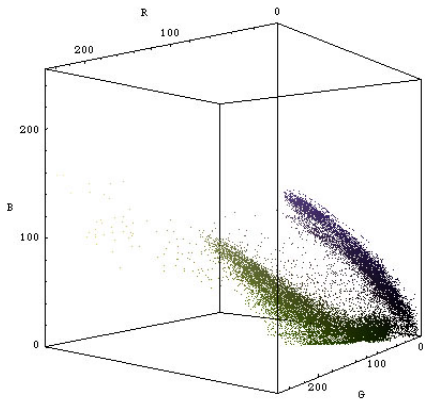


Figure 3-17: Combined Training data shown in Figure 3-16 plotted in RGB color space.

As can be seen, as the number of codes in the VQ codebook increases, the better the clusters conform to the natural groupings within our training data. This can be taken too far however. If we allowed the VQ codebook to increase to 32 or 64 for example, misclassification may result.

Now that we have generated our VQ codebook, we can now utilize it in classification. In order to classify the objects in an unknown image, we now need to build histograms for each object. In order to do this, the VQ codebook is used to quantize both data sets separately. This is accomplished by counting the frequency of occurrence of each VQ code in each data set and normalizing to fit probabilistic constraints. Because we are training the system, we know exactly from which object (or class) each data point came from. This allows us to form histograms for each object in our training set. Figure 3-19 shows the resulting histograms for both training objects.

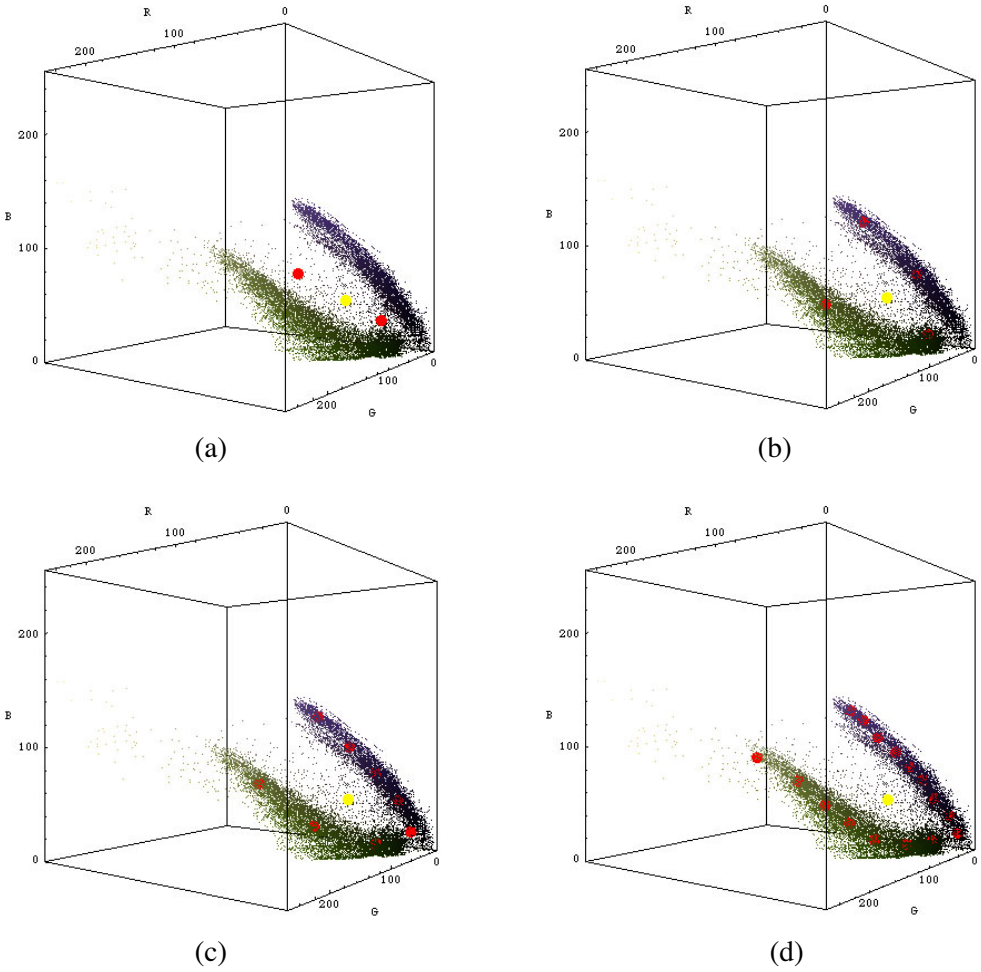


Figure 3-18: Intermediate steps for the LBG vector quantization algorithm for our training data. (a) Step L = 2. (b) Step L = 4. (c) Step L = 8. (d) Step L = 16. The yellow dot near the center of the data represents the mean of the data. The red dots represent the actual values of the VQ codebook.

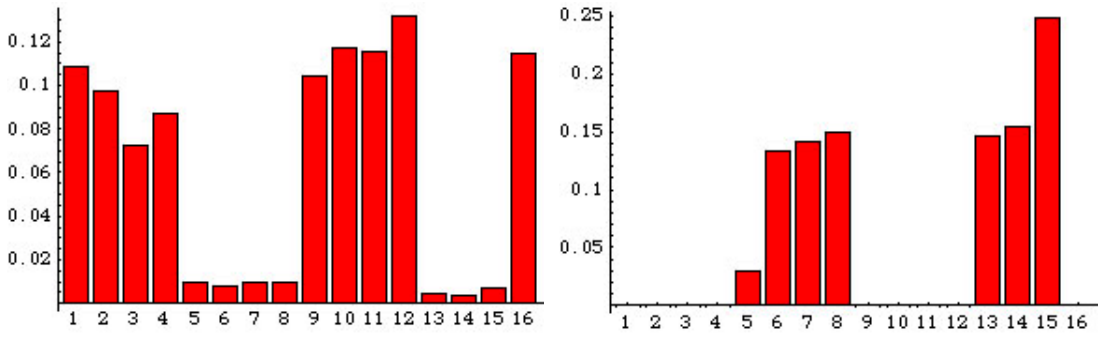
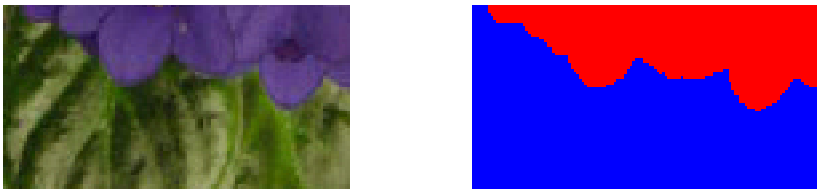


Figure 3-19: Histograms generated for each object in our training set. (a) Data from the flower training set. (b) Data from the leaf training set.

As we can see, the VQ codebook codes 1, 2, 3, 4, 9, 10, 11, 12, and 16 have a high frequency of occurrence for the data contained in the flower training set and similarly codes 5, 6, 7, 8, 13, 14, and 15 have a high frequency of occurrence for the data contained in the leaf training set. Now if we have unknown data we wish to classify, we can evaluate the probability of the unknown data given each model (histogram). The data is classified to the model (histogram) that yields the higher probability. For instance, say that we have a pixel we wish to classify. We first see which VQ code (cluster) the unknown pixel is closest too. Knowing this, we examine the values of the histograms for that particular VQ code. For example, if our pixel was closest to VQ code 4, we see that for the flower training set, it has a value of about 0.09 or so. However for the leaf training set, the value is zero. Clearly, the probability of our pixel belonging to the flower training set is higher than that for our leaf training set. Thus we would label the pixel as belonging to the flower training set. Figure 3-20 shows the result of classifying a "unknown" image.



(a) (b)

Figure 3-20: Clustered and classified result for the flower and leaf example (a) Unknown image. (b) Clustered and classified result of the image shown in (a).

CHAPTER 4
TEXTURE ANALYSIS

4.1 Introduction

In this chapter, we combine all of the tools discussed in the previous section to develop a system for performing texture analysis through the use of the wavelet transform. We first give a graphical representation of the system and then discuss the specifics of each block where warranted. The next chapter will show the results of this system applied to two tasks.

As was the case of our clustering and classification example in the previous section, we can logically break our system down into two distinct stages. The first stage involves the generation of the VQ codebook which can be referred to as the training phase of our system. In the second stage, we can use the VQ codebook generated in the training phase to classify our unknown data. Thus our second stage can be referred to as the clustering and classification stage.

4.2 Training Phase

Figure 4-1 below shows a graphical representation of the training phase of our texture analysis system. As can be seen, the inputs to our system are composed of the various training data for the objects in which we are interested in identifying later in the classification stage. We must generate the VQ codebook on the combined training data. However, we must remember from which object each element of training data belongs to so that we can generate the histograms.

As can be seen in Figure 4-1, we first start with a series of training images. These images represent the different objects that we wish to identify in an unknown image. For our texture analysis algorithm, we operate only on greyscale images. Thus, all color information is ignored.

Features are extracted from these greyscale images, one image at a time, by means of the wavelet transform. The next subsection describes in detail how this is accomplished.

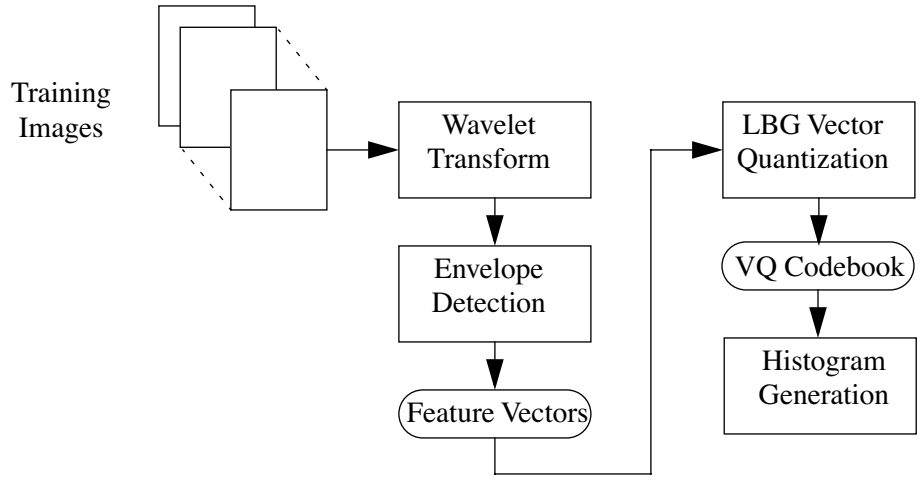


Figure 4-1: Training phase of the texture analysis system.

4.2.1 Wavelet Transform Stage

Because we are dealing with images which are two-dimensional, we will carry out our analysis by means of a two-dimensional wavelet transform. As stated in the previous section, we will use the LeMarie-Battle wavelet/filter with an arbitrary wavelet decomposition of a discrete wavelet packet tree. Thus, not all sub-bands of a particular level will be used for the purposes of generating the VQ codebook which will later be used for classification. This will be discussed further in the next chapter where we show how the wavelet sub-bands are chosen for each task.

4.2.2 Envelope Detection Stage

After features have been extracted from a particular training image by means of the wavelet transform stage, the envelope detection stage is used to make the output of the wavelet transform stage more acceptable to the clustering and classification stages. We should note several important points when applying the envelope detector to a wavelet sub-band. In our case, the four distinct wavelet sub-bands generated as a result of the two-dimensional wavelet transform have different characteristics. For example, the LL sub-band is a down sampled version of the original image (or

in a multiple level wavelet transform, a down sampled version of the previous node in the tree). Thus, application of the envelope detector to this band would not yield any significant results. However, the LH, HL, and HH bands do tend to isolate horizontal, vertical, and diagonal features respectively. It would therefore make sense to apply the envelope detector either column-wise or row-wise depending on the particular isolation properties of that particular wavelet sub-band. Given this is the case, we will apply the envelope detection algorithm according to the below table:

Table 4-1: Orientation application of the envelope detector.

Wavelet Sub-bands	*Features Isolated*	*Application Orientation of Envelope Detector*
LL	None	None
LH	Horizontal	Column-wise
HL	Vertical	Row-wise
HH	Diagonal	Column-wise

4.2.3 Feature Vectors

While the generation of feature vectors is not a process or stage in the overall process, it is important to discuss how the "feature" vectors are formed. The feature vectors are formed in the following manner: A vector is constructed for each pixel in the input image consisting of the values of the of the various wavelet sub-bands after the envelope detection process has taken place. Figure 4-2 denotes this graphically. It is important to point out, that in the figure, we show *all* wavelet sub-bands contributing to the formation of each vector. Because we are using an adaptive wavelet decomposition, not all wavelet sub-bands will be used in construction of the feature vectors. This will be explained further in the next chapter. Also it is important to point out that the envelope detection stage is only applied to the last level in the wavelet tree. We assume that the

envelope detection stage has been applied to the last stage in the wavelet tree before the vectors are formed.

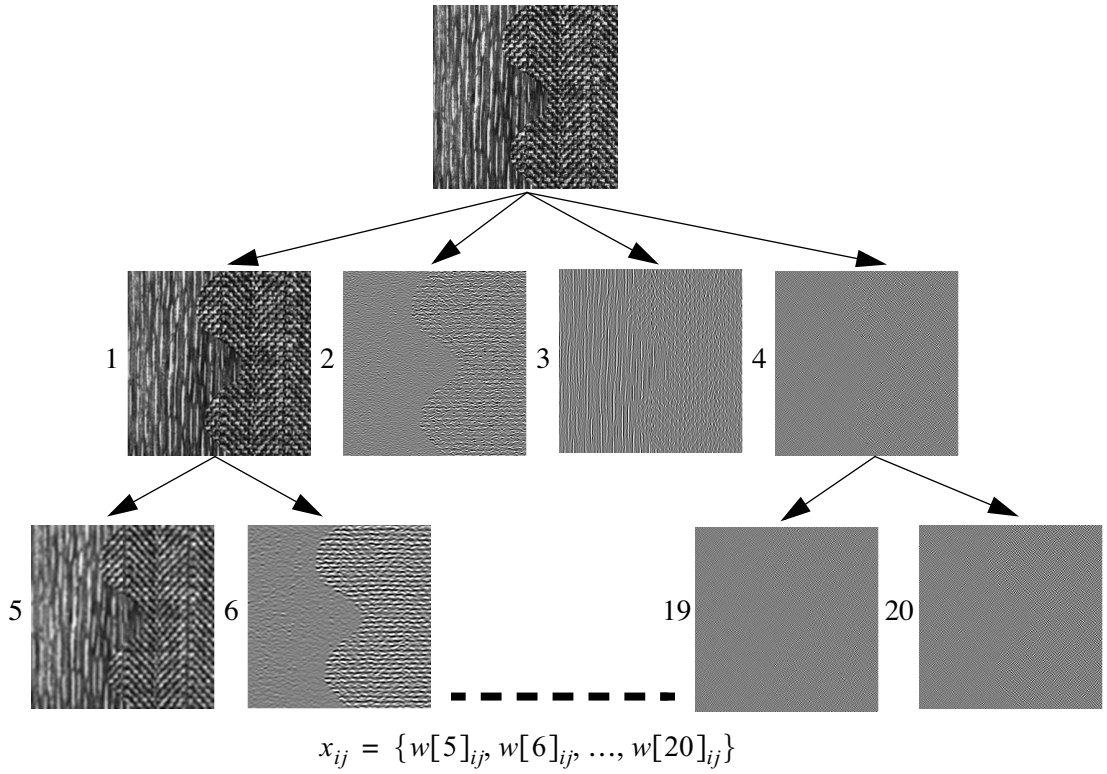


Figure 4-2: Graphical illustration showing how the feature vectors are formed.

4.2.4 Vector Quantization Stage

After the feature vectors have been generated for all of the training data, the LBG, vector quantization algorithm is applied to these vectors. What results is the VQ codebook. What we have not mentioned so far is what to choose for the size of the VQ codebook. We will defer this discussion until the next chapter.

4.2.5 Histogram Generation

The histograms are generated for each class of training data. These are generated in the same manner as described in the previous chapter. As mentioned earlier, because we are training the system with known data we have provided, we know from which object each feature vector came from. It is this knowledge we use to construct the histograms.

4.3 Clustering and Classification Phase

Now that we have the VQ codebook and histograms generated for our training data, we can now turn our attention to clustering and classifying unknown data. The unknown data is greyscale image data just as was the case for the training data. However, we will limit our unknown image size to 320 pixels horizontally by 240 pixels vertically. An image size of 320 by 240 pixels is large enough to maintain the textural features of the objects present in the image and is smaller than the traditional image size of 640 by 480 pixels. The reduced image size significantly reduces the computational time of the texture analysis algorithm. Figure 4-3 below shows a graphical representation of the clustering and classification stage.

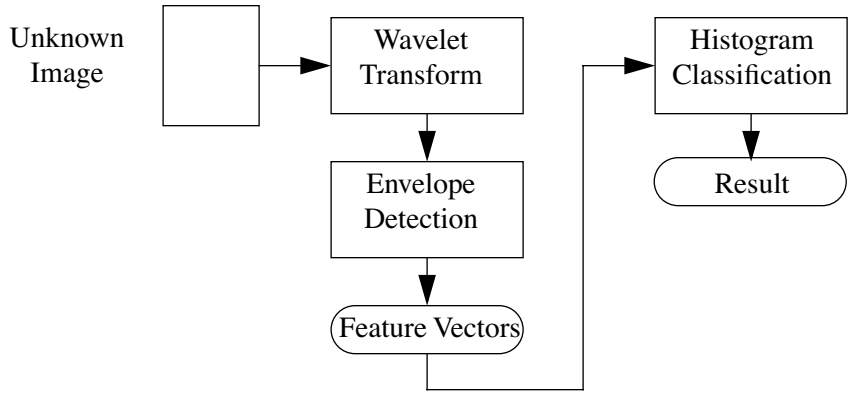


Figure 4-3: Clustering and classification stage of the texture analysis system

It is important to point out that the wavelet transform, envelope detection, and feature vector generation stages are identical to that of the training phase. The same meaningful features that were extracted from the training data must be extracted from the unknown data in the same manner in order for our analysis to work properly. Once we have generated the feature vectors for our unknown data, we can then use the VQ codebook and find out which cluster (or code) the unknown vectors are closest to and then use the histogram data to make a determination to which class the unknown pixels belong. This stage is labeled “Histogram Classification” in Figure 4-3 above.

CHAPTER 5
LAWN TEXTURE CLASSIFICATION

5.1 Introduction

In this chapter, we show how the texture analysis system presented in the previous chapter is used to perform the task of differentiating between the cut and uncut surfaces of a lawn. Thus, if the mower knows where the boundary is between the cut and uncut lawn surfaces, it can track this boundary as it mows. In this manner, a system is produced that is capable of mowing similarly to how a human would mow a lawn. This is the main emphasis of this research. For the robot mowing task, lawn images are collected and processed offline (not in real time) to avoid the hazards associated with actual lawn mowers. In the next chapter we will show how this same system can easily be tailored to the task of tracking a sidewalk. Because of the non-hazardous nature of this task, this task is implemented on an autonomous robot.

We begin this chapter by discussing how the system described in the previous chapter was tailored to perform the task of lawn texture classification. Specifically, we will discuss which wavelet subbands we chose to perform the analysis and how we went about their selection. Also, we will discuss the setup of the VQ algorithm. Finally results are presented for the system by showing several images of lawn texture classification. Several algorithms are also presented for determining the linear boundary between the two regions. This is important because this boundary becomes the main input for an autonomous lawn mowing agent.

5.2 Criteria for Wavelet Subband Determination

In order to determine which wavelet sub-bands are relevant for the task of lawn texture classification, we need to have some type of criteria to judge each wavelet subband individually. Those that meet our criteria will be kept for use in the formation of the feature vectors. Listed below are

the criteria we chose to determine if a particular wavelet subband is suitable for our purposes.Later on in Section 5.4, we will graphically and numerically show how these criteria are used to select the wavelet sub-bands for our test data.

1. In order to keep the processing time down to a minimum, we would like to keep the number of wavelet subbands down to a minimum and still produce acceptable results. This also relates to which level of wavelet tree we generate since the number of nodes in the wavelet tree increases exponentially as one descends from one level to the next. Thus, we will limit our analysis to a level 2 wavelet tree producing 16 wavelet subbands.

2. Because of the strong vertical features exhibited by the lawn surface, we will disregard any HL sub-band (and any of its descendants) for analysis except if its parent was from a LH band. This HL band tends to isolate vertical features that are present in the uncut and (to some degree) the cut lawn surfaces, thus, this band is not useful to add to the content of the feature vectors. An LH band is beneficial due to the fact that this band isolates horizontal features which are not contained in the uncut lawn texture but are contained in the cut lawn texture. Thus this band (and any of its descendents) will be particular useful for our classification purposes.

3. From a mathematical perspective, we can compute the amount of energy contained within each wavelet subband. A discrete wavelet analysis of a signal involves convolving that signal with various filters. These filters in turn, isolate various frequencies in the input signal and group them into banks of low, middle, and high frequency content. The outputs of the various filters represent the amount of frequency content contained within that particular subband or channel. Thus, if we could calculate the amount of frequency information contained within a subband and compare it with other subbands, we could make a decision on whether or not to decompose it further. We will use Equation 5-1, known as the L1-norm or variance, to accomplish this [1]. In Equation 5-1, M and N represent the dimensions of the image. This equation sums up the absolute values of all the coefficients present in the particular wavelet subband. After that, the sum is then divided by the number of pixels in the image.

$$e = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} |x(m,n)| \tag{5-1}$$

4. For a final determination, we can visualize a particular wavelet sub-band by performing a list density plot. A mathematical package such as Mathematica or other similar tool can be used for this purpose.

5.3 VQ Parameter Determination

Due to the self initializing nature of the LBG, VQ algorithm, only one parameter needs to be determined. This parameter is the number of codes in the VQ codebook. As stated earlier, too few codes or too many codes can lead to unsuccessful classifications. Also, because of the high

dimensionality of our feature vectors, the larger the codebook, the longer it will take to generate to total codebook. Given these considerations, and experimental results, we will use a codebook size of 8 codes. Larger codebook sizes did not yield a significant improvement in the classification of the lawn textures.

5.4 Results

In this section results are shown for the lawn texture classification system. We begin our discussion by showing a sample lawn image and then we choose the appropriate wavelet sub-bands based on the criteria outlined earlier. Next, we describe how we obtain the training data for the system. After this, we present the clustered and classified results for several lawn images. We also discuss three algorithms for determining the boundary between the cut and uncut lawn surfaces.

5.4.1 Collection of Lawn Image Data

To collect data for our analysis, a wheeled cart was constructed that allowed a standard camcorder to be mounted to it. A manual, gasoline powered, push mower was used to mow a strip of lawn in a typical lawn environment. Next the cart was pushed along the right edge of the path created by the push mower, thus capturing images consisting of the cut lawn surface on the left and uncut lawn surface on the right. After capturing the video, it was then input into a computer where it was isolated into individual image frames, downscaled to 320x240 pixels, cropped, and converted to greyscale. We crop the images vertically to get rid of the upper portion of the image. This has to do with the fact that the camera sits less than a foot off the ground and is slightly angled to the ground. Thus, the top of the image appears (and is) further away and does not lend itself to good classifications because the strong vertical features of the cut side are not as prevalent as they are in the lower portion of the image. Figure 5-1 below shows a lawn image captured from our system. As one can see, due to the low height of the camera, the vertical features of the uncut portion the image are highly emphasized lending itself to a better classification.

Figure 5-1: Captured lawn image. The cut lawn surface is on the left. The uncut lawn surface is on the right.

5.4.2 Determination of Wavelet Subbands for Lawn Texture Analysis

Figure 5-2 shows the list density plots for each of the 16 wavelet subbands that are generated in the level 2 DWPF wavelet tree decomposition. In Figure 5-2, below each subband is text describing what type of subband it is (LL, LH, HL, HH) and from what type of subband it is descended from. The number at the end is the value of the L1, norm energy measure as described in Equation 5-1. It is important to note that this energy measurement is computed before the application of the envelope detection stage.

Now that we have some information from which to make an assessment, let us use the criterion we listed previously to justify our selection of the wavelet subbands used for the lawn texture classification system. First, let us state the subbands that we do intend to use for analysis purposes and then explain why we eliminated the others. We will use subbands 6, 8, 10, 11, and 12.

Now we will describe how we eliminated the other subbands. First of all, we will eliminate subbands 13 through 16 because they are descendants of the top level HL[3] band. Secondly, we will eliminate bands 17 through 19 based on their low energy content. Although band 20 may be useful for classification purposes, we eliminate this band to keep the number of subbands to a minimum. Next, we eliminate band 5 because this is a lowpassed version of the original image.

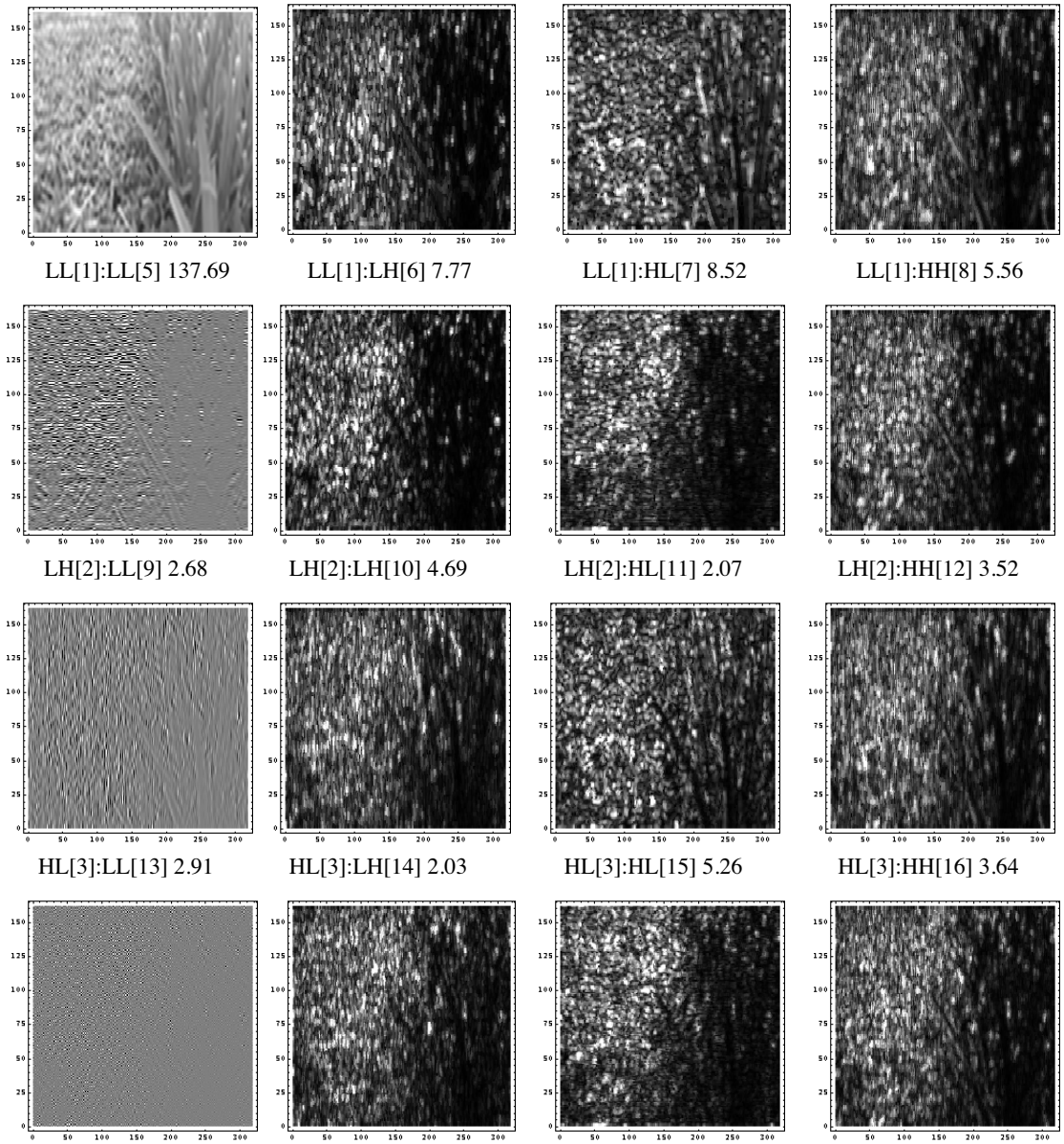
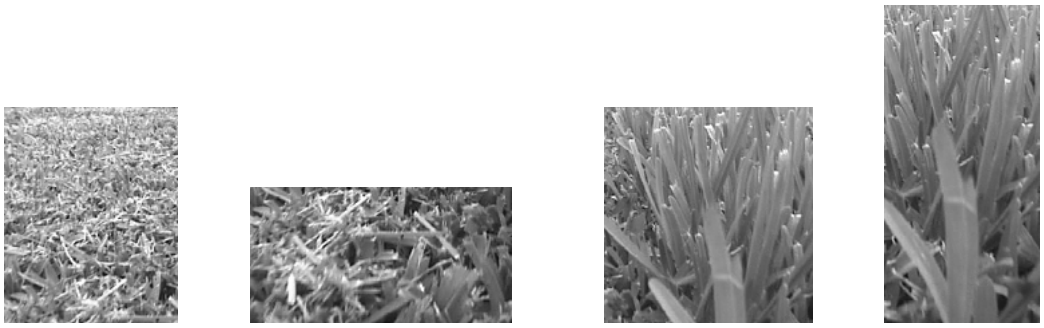


Figure 5-2: Wavelet sub-bands contained in level 2 of the DWPF for the image shown in Figure 5-1. The text, XX[A]:YY[B] C, below each subband can be interpreted as follows: XX represents the parent node from which this sub-band is generated, YY represents what type of subband this is, A and B represent the index number for each subband, and C represents the L1, norm energy measurement for each subband.

Band 7 gets eliminated because it is a HL band which is descended from the LL[1] band which is a lowpass version of the original greyscale image. Finally, we eliminate band 9 since it is a LL band.

5.4.3 Selection of the Training Data

Now that we have determined which wavelet subbands will be used for the purposes of lawn texture analysis, we need to obtain training data in order to generate the VQ codebook. This codebook, as stated in the previous chapter, is generated during the training phase. To obtain training data, we cropped small samples, 2 for the cut lawn surface and 2 for the uncut lawn surface from one frame of image data. This was the only training data used for the entire series of lawn images that were classified. Figure 5-3 shows the training data used to generate the VQ codebook.



(a) (b) (c) (d)

Figure 5-3: Training data used to generate the VQ codebook for our results. Figures (a) and (b) represent the cut training data and figures (c) and (d) represent the uncut training data.

5.4.4 Clustered and Classified Results

Below we present several images showing the results of our texture analysis system for determining the boundary between the cut and uncut lawn surfaces. We will also discuss three separate methods used to calculate the linear boundary between the cut and uncut lawn regions in each image. Figure 5-4 and Figure 5-5 show some results of our system.

5.5 Line Boundary Determination

Three methods were employed to determine the linear boundary between the cut and uncut classified regions. If the mower is supposed to mow as a human would mow, it needs to be able to determine the boundary between the cut and uncut regions. What follows is a description of the three methods used.

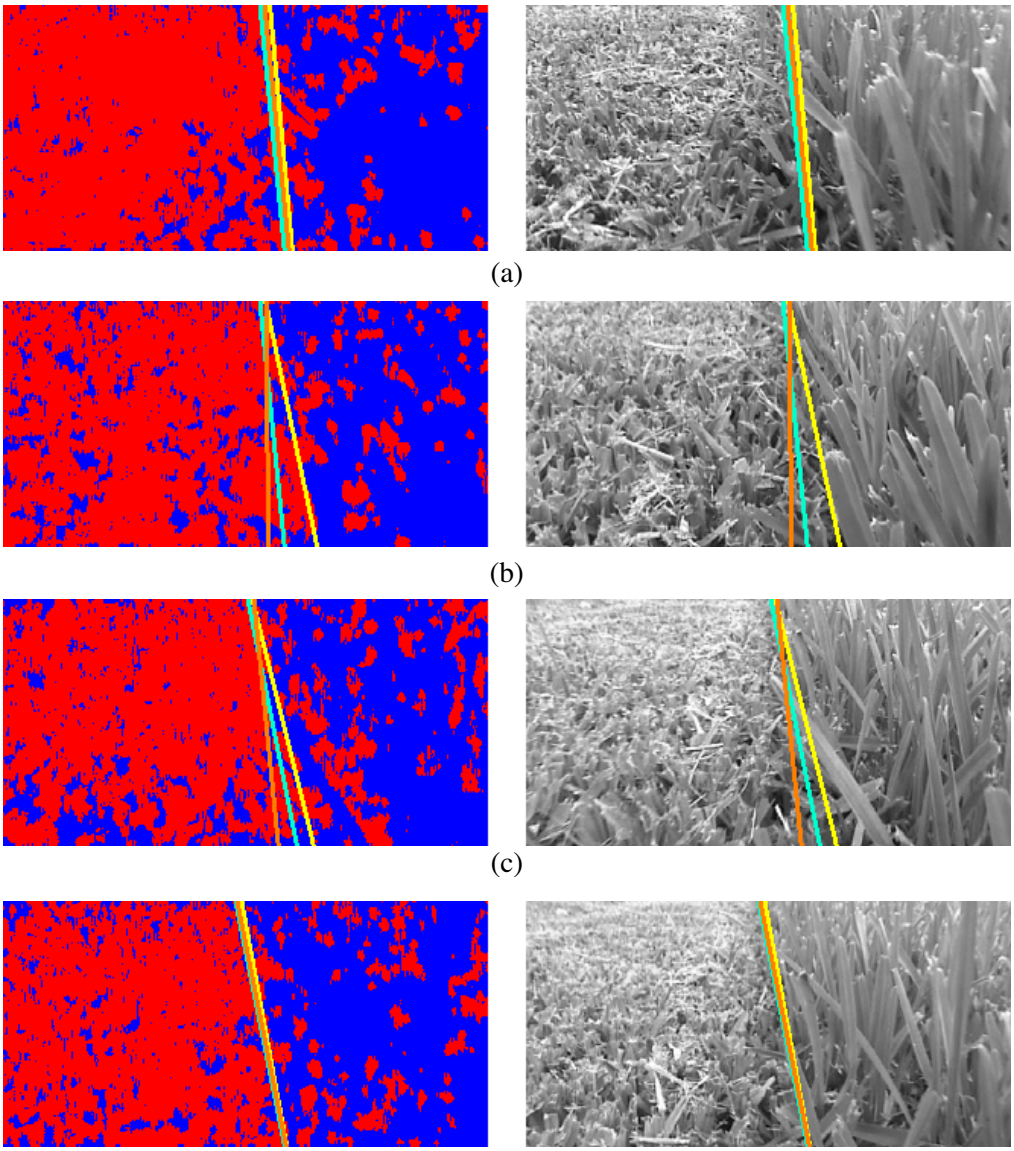
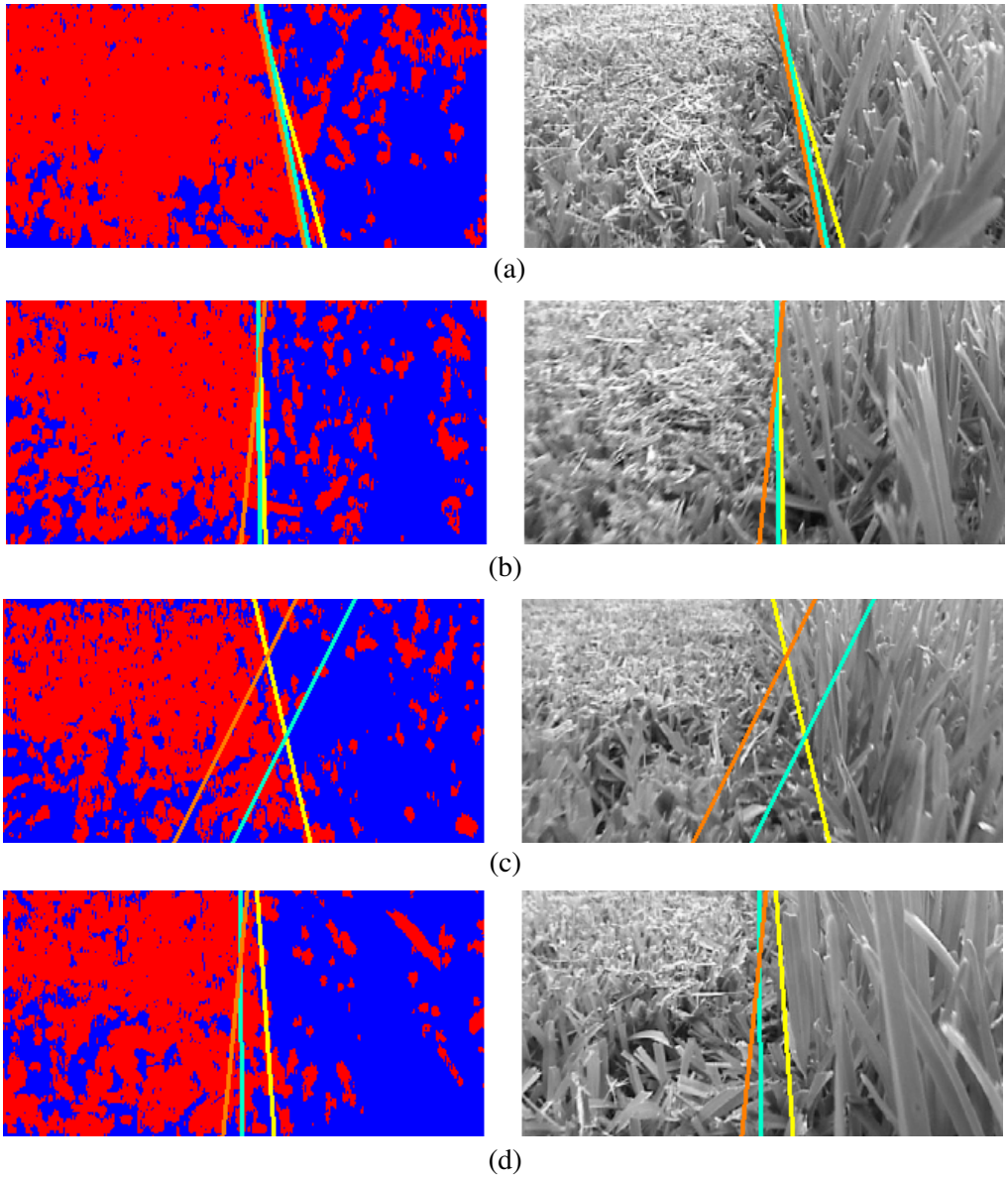


Figure 5-4: Output results for our lawn texture classification system. The images on the left hand side represent the classified and clustered result of the images presented on the right hand side.

5.5.1 Best Fit Step Function Method

This first method is performed on a line by line basis by attempting to fit the data of the two classes to the best fit (lowest least-squared error) step function [21]. This is depicted graphically in Figure 5-6 below. The parameter $d(i, j)$, is referred to as the discriminant and in our case is simply

the value given to each class. For instance, $d(i,j) = 1$ for the pixels classified as the cut lawn texture and $d(i,j) = 0$ for pixels classified as the uncut lawn texture.



(a) (b) (c) (d)

Figure 5-5: More results from the lawn texture classification system.

To compute the best fit step function we exhaustively compute the least squared error function for each jd in line i. This involves computing the means *ml* and *mr* which represent the mean of the data on the left side of jd and the mean of the data on the right hand side of jd respectively.

The value of jd which yields the lowest error is assigned to be the best fit for that particular line. Equation 5-2 shows the least-squared error function.

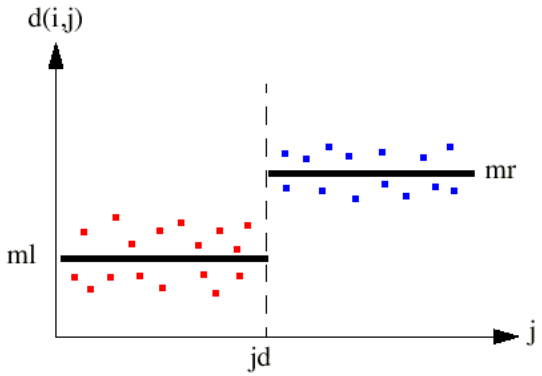


Figure 5-6: Graphical representation of performing the best fit step function method.

$$error = \frac{\sqrt{\sum_{j=0}^{jd}[d(i,j)-ml]^2 + \sum_{jd+1}^{jmax}[d(i,j)-mr]^2}}{jmax+1} \tag{5-2}$$

Once we find the best fit step for every line in the image, we use linear regression to plot a line through the best fit step points. To take care of any points that may be outliers, after performing linear regression, we go back and calculate the distance from each best fit step point and the line itself. A point that fits within our prescribed tolerance is kept, all other points are discarded. After this, we perform linear regression again using only these acceptable points thus producing a line that better fits the given data points. In Figure 5-4 and Figure 5-5 above, this line is represented by the yellow line.

5.5.2 Maximizing the Minimum Method

In this method, we begin by partitioning the classified result into two parts by separating them linearly by a candidate line. We refer to the line as a candidate line since we will try several lines and see which one gives us the least error according to this method. The line that yields the least error will be the boundary between the cut and uncut lawn textures. The algorithm is outlined below:

1. Set $max = 0$.

2. Linearly partition the classified result into two by the candidate line.

3. Let *p1* be the percentage of pixels from class 1 on the left hand side of the candidate line and let *p2* be the percentage of pixels from class 2 on the right hand side of the candidate line.

4. If $p1 < p2$ then $min = p1$, else $min = p2$.

5. If $min > max$ then let $max = min$.

6. Go to Step 2 until the classified result has been partitioned by all possible candidate lines.

The candidate line where the condition in Step 5 of the algorithm was last met is the one chosen to be the boundary between the cut and uncut lawn surface. This line is represented in Figures 5-4 and 5-5 by an orange line.

5.5.3 Maximization of Area Method

This method is similar to the "maximizing the minimum" method in that we partition the classified result into two parts by the selection of an appropriate candidate line. In this method, we try to linearly partition the boundary between the two classes in such a manner that maximizes the area of the two classes on either side of the line. For instance, the best possible line would be one where all the pixels from one class were on one side of the line and pixels from the other class were on the other side of the line. Obviously we may never have an ideal case, but we are interested in finding the line which maximizes this criterion. We formally state the algorithm below:

1. Set $maxavg = 0$, $min = 1$.

2. Linearly partition the classified result into two by the candidate line.

3. Let *p1* be the percentage of pixels from class 1 on the left hand side of the candidate line and let *p2* be the percentage of pixels from class 2 on the right hand side of the candidate line.

4. Let $diff = fabs(p1 - p2)$. If $diff \leq min$ then $min = diff$ and $avg = (p1 + p2)/2$.

5. If $avg > maxavg$ then $maxavg = avg$.

6. Go to Step 2 until the classified result has been partitioned by all possible candidate lines.

The candidate line where the condition in Step 5 of the algorithm was last met is the one chosen to be the boundary between the cut and uncut lawn surface. This line is represented in Figure 5-4 and Figure 5-5 by an aqua line.

5.5.4 Line Boundary Methods Conclusions

The results of all three methods used for determining the boundary between the cut and uncut lawn surface can be seen in Figure 5-4 and Figure 5-5. As can be seen, for the most part, all three methods tend to produce results which are similar to one another. However, when viewing Figure 5-5(c), the maximizing the minimum and maximization of area methods do not yield an appropriate result. This is a result of the general nature of these methods. Both of these methods rely on the fact that pixels from one class should be present on one side of the line and that pixels from the other class be present on the other side of the line. When there are patches of mis-classification (as can be seen in Figure 5-5(c) toward the bottom left), this can have a detrimental effect on these two algorithms. The best fit method can overcome this due to the fact that the line is determined by a series of points. Even if some points are not correct, the boundary can still be determined with reasonable success as is evident in Figure 5-5(c).

To gain further insight into the effectiveness of these line boundary methods, we calculate an error measurement for each of the described methods. This is accomplished by comparing the line generated by each method to a line that has been generated through visual inspection by a human. Thus, this serves as a basis to judge the efficacy of our described methods.

To determine the error between a line generated by one of our methods and the line generated by visual inspection, we calculate the number of pixels between the two lines and then divide this number by the total number of pixels present in the image to get a percentage of the pixels contained between the two lines. If the lines happen to intersect one another (forming two triangular regions) then we count the number of pixels contained within these two triangles and then divide by the total number of pixels in the image. Thus, the lower the error, the closer the line

generated by one of the boundary detection methods and line determined by visual inspection should be to one another.

To illustrate this, the boundary line was determined by visual inspection for 266 consecutive frames. The boundary line was also computed using the three line boundary methods. The error was then computed for every frame for each line boundary method. Figure 5-7 shows a plot of the error measurements for each of the line boundary methods for the 266 consecutive frames.

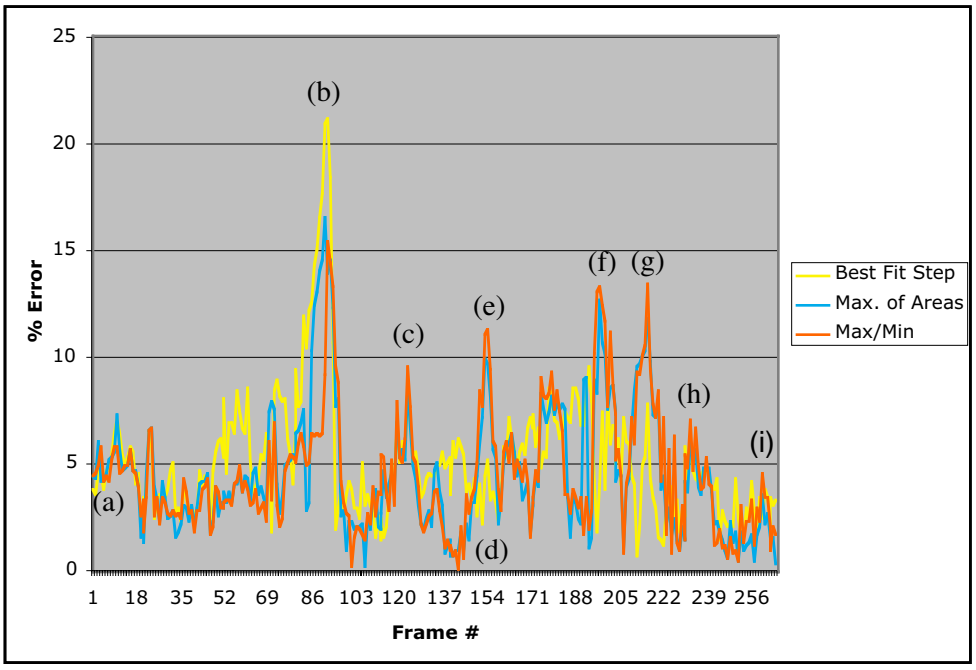


Figure 5-7: Graph of the calculated error measurements for each line boundary detection method for consecutive image frames.

Points of interest on Figure 5-7 have been labeled by the letters (a) through (i). We will show the output frames for these particular error measurements and make comments about them. To begin, let us consider Figure 5-8. This figure shows the output frame for point (a) in the error graph. As the error graph indicates, the errors for all three methods is low (around 5%) and that all three methods tend to agree with one another. The output frame for point (b) is shown in Figure 5-9. As can be seen, this image is partly occluded by a single blade of grass and is the reason for the particularly high error rate. The best fit step method shows the highest error rate of all three

methods considered. However, even though the error associated with the other two methods is lower, these lines are far from being ideal.

Figure 5-10 shows the output frame for point (c) on the error graph. For this frame, it can be seen that the error measurements for all three methods is roughly around 10%. From viewing the frame, this may seem a bit high since the lines seem to correspond well to the boundary between the cut and uncut regions. However, this error is due to the choice of the visually detected line from which the error calculations were made.

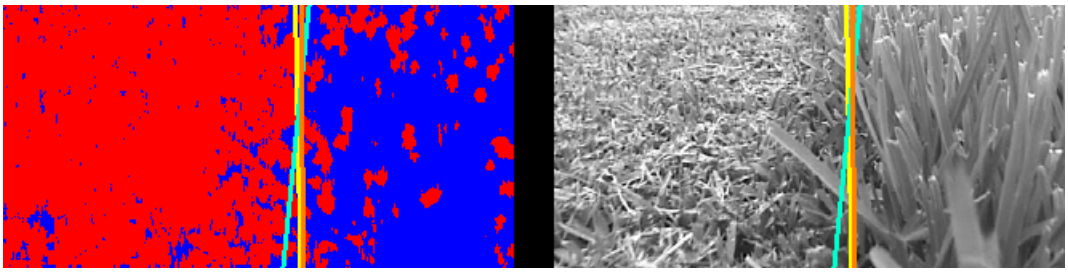


Figure 5-8: Output frame for point (a) on Figure 5-7.

The output frame for point (d) on the error graph is shown in Figure 5-11. As can be seen, all three lines agree fairly well with the boundary that can be seen in the figure. However, the lines generated as a result of the minimizing the maximum and maximization of areas methods have a lower error rate (close to 0%) as opposed to the best fit step method with an error rate of about 5%.

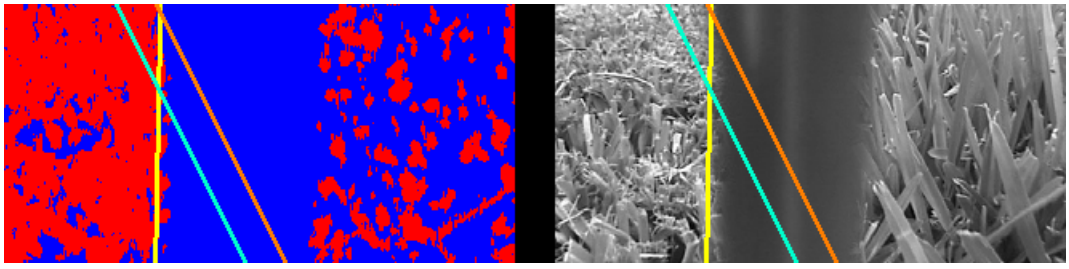


Figure 5-9: Output frame for point (b) on Figure 5-7.

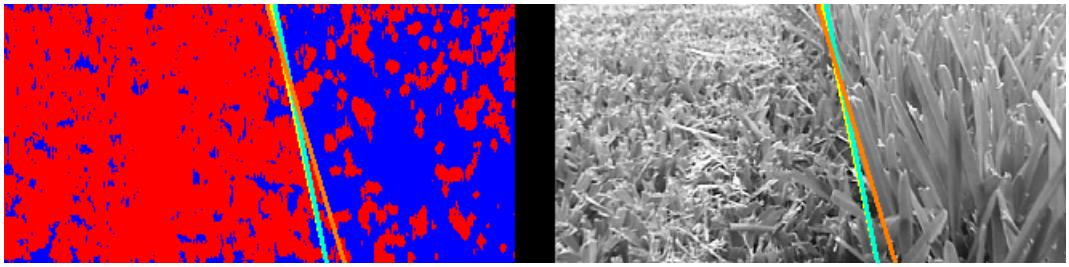


Figure 5-10: Output frame for point (c) on Figure 5-7.

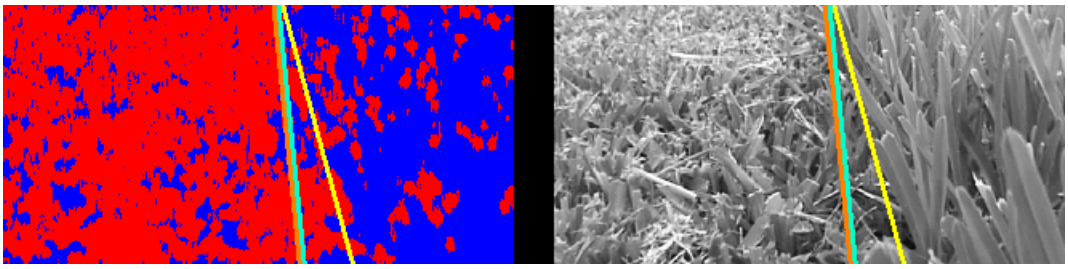


Figure 5-11: Output frame for point (d) on Figure 5-7.

Figure 5-12 shows the output frame for point (e) on the error graph. As can be seen in the right half of the concatenated image, the image is blurry do to the inability camera's auto focus to compensate quickly enough for the changing conditions. The condition that changed is the height of the camera due to a high spot in the mowing environment. This has a result on the classified result portion of Figure 5-12. As can be seen, there are more mis-classifications on the cut side of the boundary between the cut and uncut lawn surface. This obviously effects our ability to calculate the boundary as is evident from the error graph. Both the maximizing the minimum and maximization of area method both incur an error of roughly 12%. However, the best fit step method yields an error of about 3% making it the better choice in this instance. This seems reasonable since the maximizing the minimum and maximization of area methods expect the majority of pixels from one class to be on one side of the line and the pixels from the other class to be on the other side of the line. This is not the case with the best fit step method due to the fact that the line is computed from multiple points in the image. The error results obtained for points (f) and (g) can be

explained for the same reasoning that has been given for point (e). Figure 5-13 and Figure 5-14 show the output frames for points (f) and (g) respectfully.

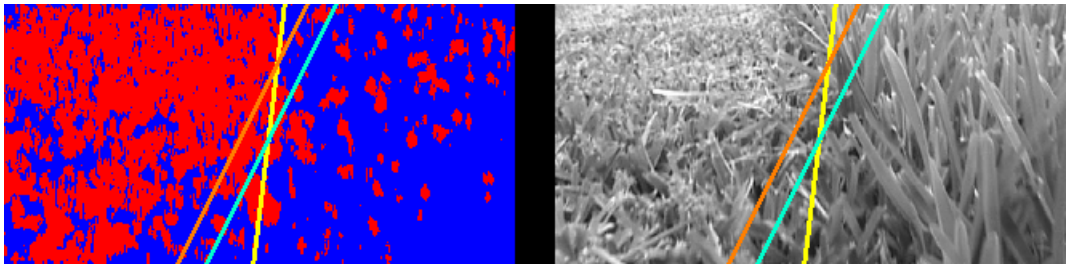


Figure 5-12: Output frame for point (e) on Figure 5-7.

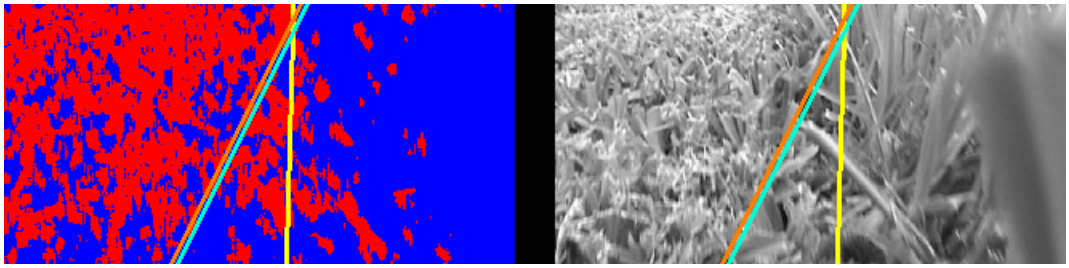


Figure 5-13: Output frame for point (f) on Figure 5-7.

Finally, we turn our attention to the last two error points, (h) and (i). For both of these points, the error measurements for all three methods are roughly identical. However, the error measurements for point (h) are slightly higher than that for point (i). The output frames for error points (h) and (i) are shown in Figure 5-15 and Figure 5-16 respectfully. As can be seen, all three methods yield excellent results for both frames. The difference in error is due to the choice of the visually determined boundary used in the calculation of the error criterion.

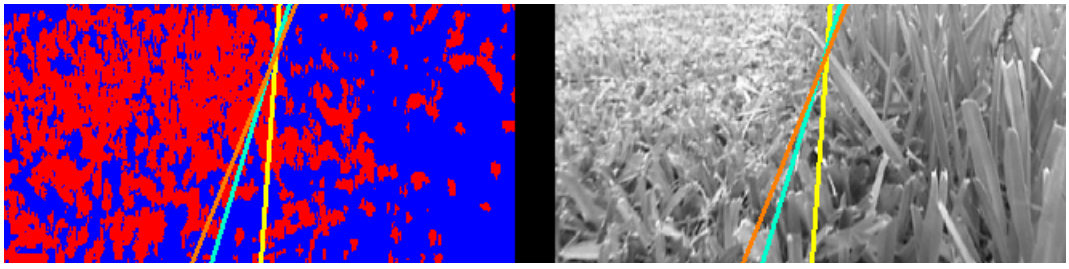


Figure 5-14: Output frame for point (g) on Figure 5-7.

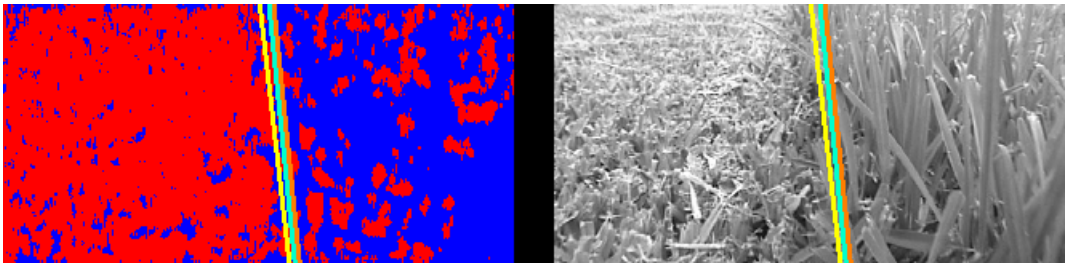


Figure 5-15: Output frame for point (h) on Figure 5-7.

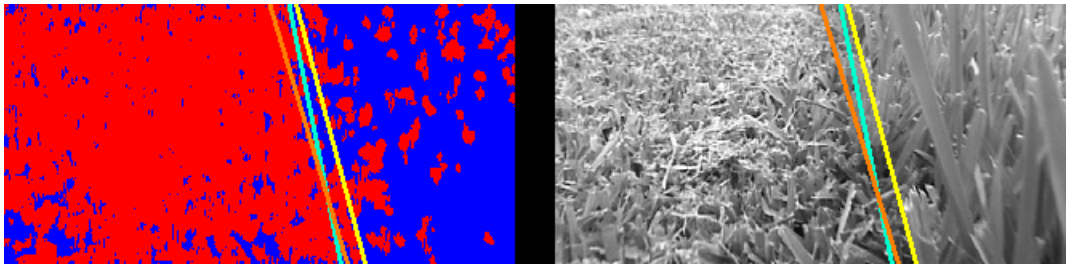


Figure 5-16: Output frame for point (i) on Figure 5-7.

From Figure 5-7 we can infer some important points about which method works the best. First, the best fit step method appears to generally have the lowest error throughout the series of processed frames. The exception to this is point (b) where a single blade of grass occluded the boundary and led to a missclassification. This high error rate could not be avoided no matter what boundary detection method we employed. Secondly, both the maximizing the minimum and maximization of area methods tend to produce error rates that are consistent with one another. Because of the underlying nature of these two methods, they are prone to high error rates when there is a high degree of missclassification of the input image.

CHAPTER 6
SIDEWALK TEXTURE CLASSIFICATION

6.1 Introduction

In this chapter, we adapt the texture analysis algorithm to perform sidewalk identification. This in turn, will be used to allow an autonomous mobile robot to follow a sidewalk as it moves forward. A system which is designed to operate an autonomous robot should operate close to real time as possible. Otherwise, the robot may miss pertinent information necessary to carry out its task.

We begin this chapter, as with the last one, by describing how the texture analysis system was adapted to that of performing sidewalk texture classification. We then discuss the experimental platform and describe the test system. Results are then presented.

6.2 Criteria for Wavelet Subband Determination

As was the case for our lawn texture classification system, it is necessary to chose appropriate wavelet subbands for our classification purposes. Instead of lawn textures, which consisted of the cut and uncut lawn surfaces, we will be dealing with textures such as concrete, asphalt. and some type of ground cover on the periphery of the sidewalk. Thus, we must consider different criteria for determining which wavelet subbands to use. Listed below are the following criteria.

1. Because this is a real-time application, we want to limit our analysis to as few wavelet subbands as possible.

2. We will use the same energy metric as was used in the previous chapter for computing the amount of energy (indicator of frequency content) in a wavelet subband. Those bands that posses high frequency content will be considered for analysis.

3. Finally, we will visually examine each subband by generating list density plots.

6.3 VQ Parameter Determination

As stated in the previous chapter, the only parameter we must specify in the VQ algorithm is the codebook size. Because of the real-time nature of this application, we will use a codebook size of 4 for our analysis.

6.4 Robot Platform and System

To test our algorithm, we modified an existing robot platform to make it suitable for this task. The platform in question is a four wheeled robot in which the two wheels in the back of the robot are the drive wheels and a linear actuator type steering controller is used to control the front two wheels. In order to obtain images, a camcorder was mounted in a wooden frame that allowed the camera to be adjusted in terms of height and camera angle. This frame was attached to the front of the robot. A picture of the robot is shown in Figure 6-1.

To acquire image data from the camcorder, a PCI based video frame grabber card was used. In addition, we required a considerably fast system to run the texture analysis algorithm. These two requirements led to the decision to use a desktop PC. Specifically, we used an Intel P4 - 1.4GHZ computer with 256MB of RAM running the LINUX Mandrake 8.0 operating system. Because of this, the PC was placed on a cart that was pushed behind the robot. This can be seen in Figure 6-1. On top of the PC (as can be seen in Figure 6-1) a laptop computer was used to remotely login to the PC so that the program could be started and stopped and the results of the texture analysis algorithm could be viewed. To control the speed and direction of the robot, commands were sent to the robot via a serial cable connected to the PC. Finally, to provide power to the system, a small, portable generator was used. A graphical breakdown of the system is shown in Figure 6-2.

Figure 6-1: Picture of the robot platform and support equipment.

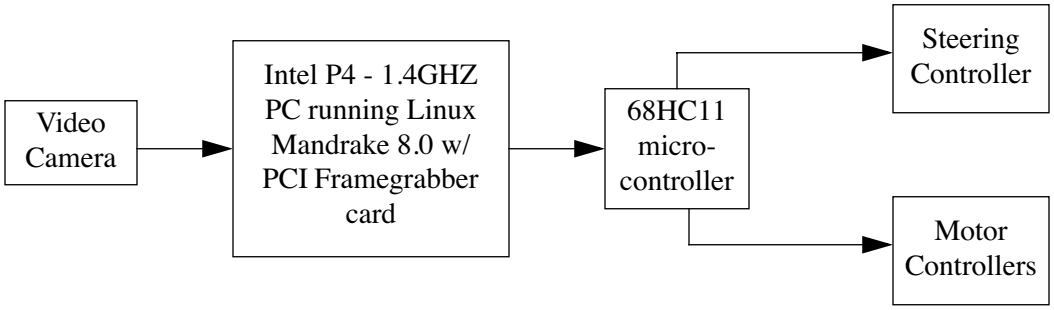


Figure 6-2: Graphical representation of the system used for autonomous sidewalk navigation.

6.5 Determination of Wavelet Subbands for Sidewalk Texture Analysis

As in the previous chapter, we present in Figure 6-4, the list density plots for each of the 16 wavelet subbands that are generated in the level 2 DWPF wavelet tree decomposition of the sample sidewalk image shown in Figure 6-3. Below each sub-band is text describing what type of sub-band it is (LL, LH, HL, HH) and from what type of subband it is descended from. The number at the end is the value of the L1, norm energy measure as described in Equation 5-1. It is important to note that this energy measurement computed before the application of the envelope detection stage.

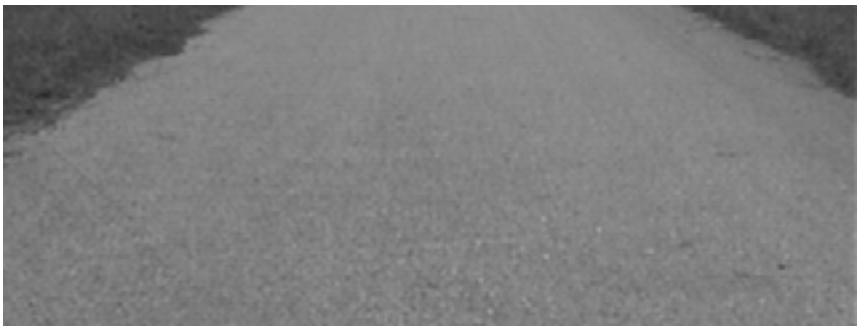


Figure 6-3: Sample sidewalk image. This image has been cropped to eliminate the bottom half of the image. This results in a faster processing time for the algorithm.

As can be seen in Figure 6-4, subband 5, the second level low passed version of the original image, yields the highest distinction between the sidewalk and the periphery area. Thus, it will be included in our analysis. It should be noted that we did not consider this particular subband for our lawn texture classification system. For that application, this band did not yield a great distinction between the cut and uncut lawn surfaces. Also, the height and angle of the camera were different in the lawn classification application making the textures of the lawn surface more pronounced. With our sidewalk system, the camera is much higher off the ground, thus limiting the amount of textural information we can view from the objects in the image.

We will also include subbands 6, 8, and 10 for use in our analysis primarily based on the energy metric. To summarize, we will use subbands 5, 6, 8, and 10 for our sidewalk texture classification system.

6.6 Training Data

Shown in Figure 6-5 below is the training data set used in all the experiments conducted with this system. The training data was acquired by capturing a single frame of video after the robot system was setup. We then cropped two regions of texture to form the training set.

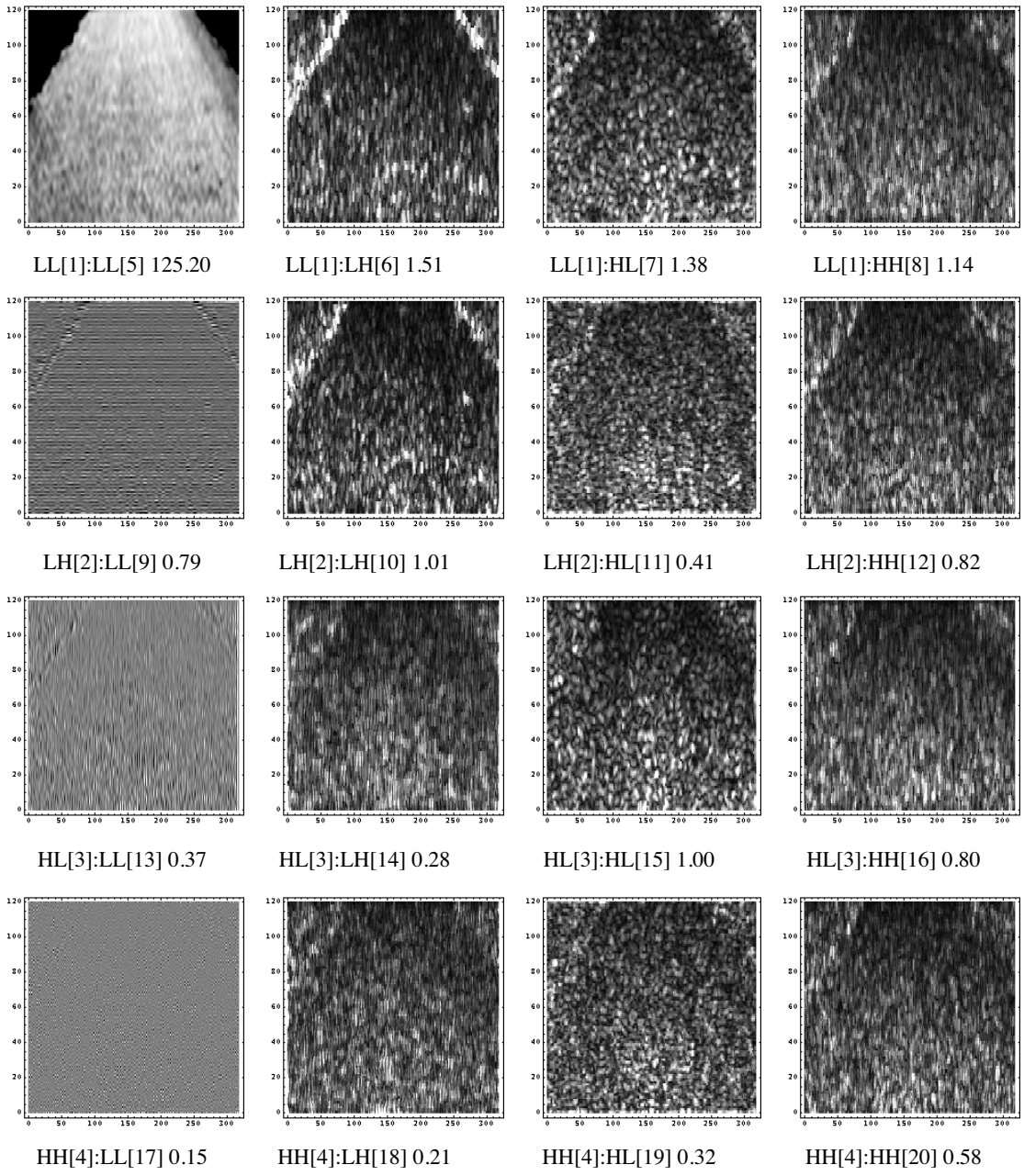


Figure 6-4: Wavelet subbands contained in level 2 of the DWPF for the image shown in Figure 6-3. The text, XX[A]:YY[B] C, below each subband can be interpreted as follows: XX represents the parent node from which this subband is generated, YY represents what type of subband this is, A and B represent the index number for each subband, and C represents the L1, norm energy measurement for each subband.

Figure 6-5: Training data set for the sidewalk texture analysis system. Figures (a) and (b) show the training data set for the sidewalk texture and figures (c) and (d) show the training data set for the periphery texture.

6.7 Boundary Detection

As was the case for the lawn texture classification system where we needed to determine the boundary between the cut and uncut regions of the lawn surface, we need to determine the boundary between the sidewalk and periphery areas of the sidewalk. In this case, we must find the boundary on the left hand side and on the right hand side. To accomplish this, we divide the image into two halves vertically and then use the best fit step method to detect the boundary in each half. Once we have located both boundaries, we then find where the lines intersect the top of the image. Knowing this, we calculate the center of the intersection of the two lines. This will be the center of the sidewalk.

We have so far described an ideal situation in which we are in the center of the sidewalk and have a left and right boundary. However, if for some reason the robot veers to far off course, only one boundary line will be present. In order to deal with this case, we must constantly check the condition of the two lines. Under normal circumstances, one line should have a positive slope, the other negative. If only one line is present in the entire image, the slope of both lines should be the same polarity, namely either positive, positive or negative, negative. If this is the case, we then

know we are in a situation where we have veered off course and are only seeing one boundary line. To get a better estimate of the line, we compute the best fit step for the entire image.

6.8 Results

Now that we have discussed the robot platform, choice of parameters, and training data for our texture analysis algorithm, we now present some results of our system. We conducted 7 runs of our system. Our test area was an asphalt sidewalk surrounded by thinly bladed grass. The speed of the robot was set to the lowest speed possible (about 1 foot per second) and the algorithm processed roughly 3 frames per second. The area where the tests occurred was at a slight incline which aided in testing. Going up the incline had the effect of slowing the robot down to about .5 foot per second. This enabled the robot to process more frames per foot as the robot traveled making its tracking more accurate. Correspondingly, when going down the hill, the robot moved at about twice the speed as it would on a level surface. Because of this, the robot tended to oscillate from side to side as it traversed down the sidewalk. The figures below show some sample output of our system. It is important to keep in mind that the robot was able to track the sidewalk as it moved in each of the scenarios. The robots steering was completely controlled by the algorithm, thus it was moving autonomously along the sidewalk. Figure 6-6 and Figure 6-7 show some sample results of our system.

6.9 Analysis of Results

As can be seen in Figure 6-6 and Figure 6-7, the robot tracks the sidewalk to a high degree of accuracy. The pictures contained in Figure 6-6 were collected from an uphill run of the robot. During this run, the robot stayed on course very well and oscillation from side to side was a minimum. This was due to the fact that the robot was slowed down enough to where it could keep up to the speed of the algorithm. However, as we can see in the pictures contained in Figure 6-7 where the

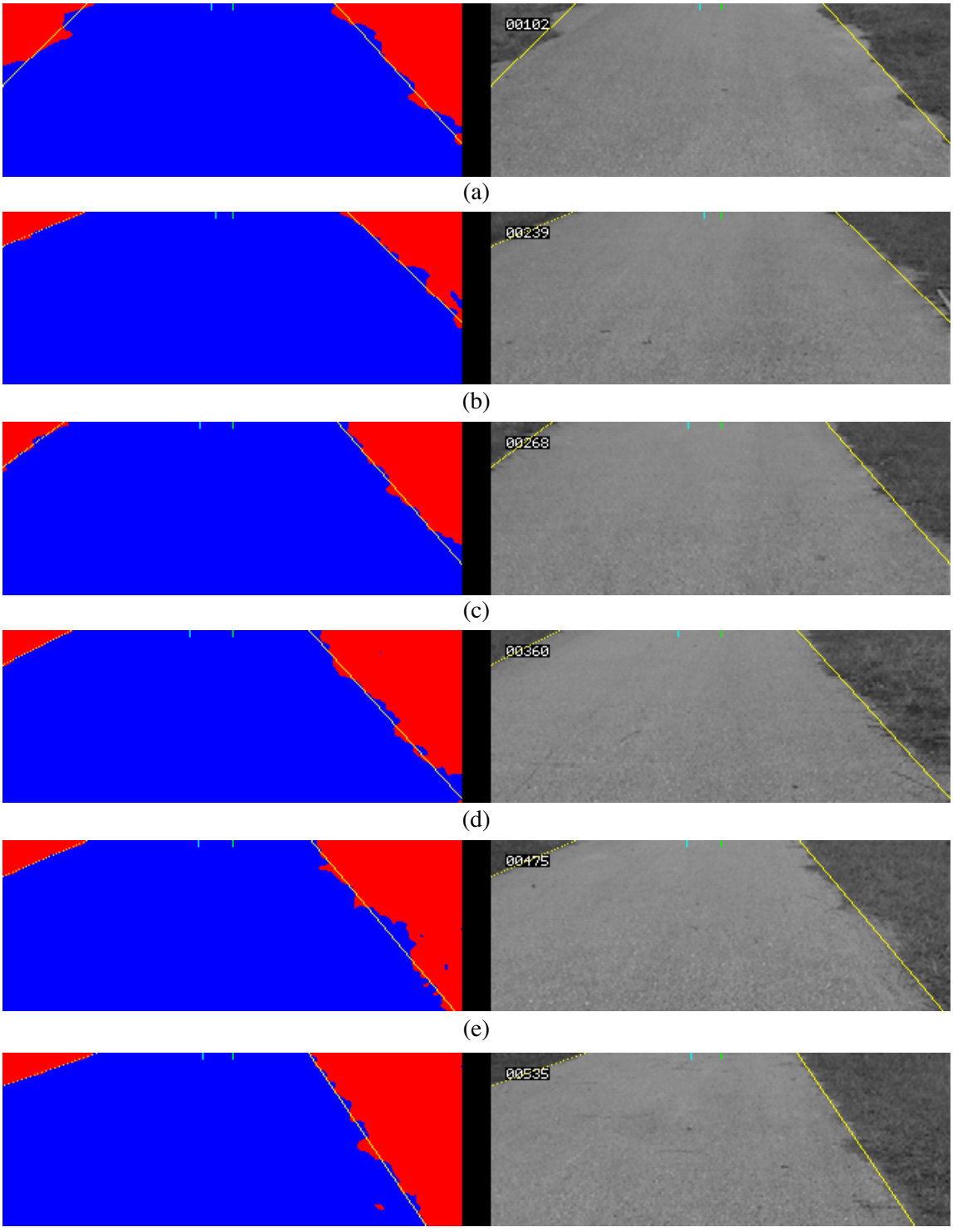


Figure 6-6: Output results for our sidewalk texture classification system. The images on the left hand side represent the classified and clustered result of the images presented on the right hand side. These images are taken going up the hill.

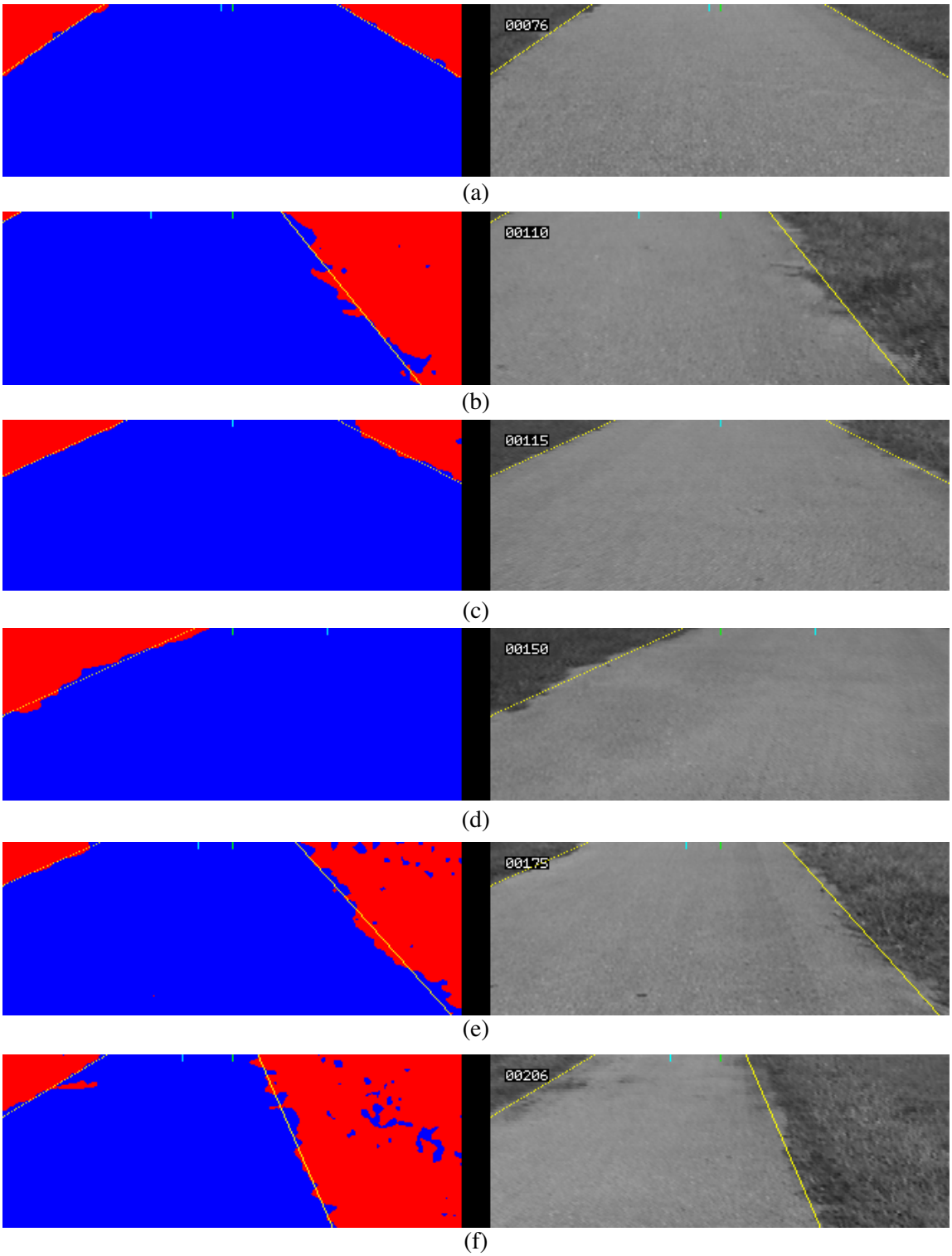


Figure 6-7: Output results for our sidewalk texture classification system. The images on the left hand side represent the classified and clustered result of the images presented on the right hand side. These images are taken going down the hill.

robot travels at a faster rate downhill, the robot tends to oscillate from side to side. In fact, in Figure 6-7 (d), the robot gets off track to where only the left boundary line is present. The robot doemanage to correct itself later on as is evident in Figure 6-7 (e).

CHAPTER 7
CONCLUSIONS AND CONTRIBUTIONS

Our main conclusion, as is the premise of this dissertation, is that robots can indeed navigate outdoor environments through the judicious use of computer vision. Specifically, we acquired images and used our texture analysis algorithm to identify specific attributes in these images. This was demonstrated by our lawn texture analysis system where we were able to successfully determine the difference between the cut and uncut lawn surfaces. In this way, we could determine the boundary between these two surfaces and use this to allow an autonomous lawn mower to mow in a pattern. We further demonstrated our claim by adapting our algorithm for the alternate application of sidewalk tracking. In this case, however, we implemented this on a mobile robot platform. That robot was successfully able to track a sidewalk as it moved along it. Several successful runs were performed on this system.

The texture analysis algorithm developed is a highly flexible one with few parameters to specify. This dissertation dealt predominately with the task of developing a system that could be used to allow a robotic lawn mower to navigate through its environment by recognizing key features of its environment. However, as demonstrated by our sidewalk tracking application, this system can easily be adapted to other tasks as well. All that is required is the choice of the appropriate wavelet subbands and the choice of the codebook size for the vector quantization algorithm.

From our experiments, we can draw several conclusions. First of all, the use of texture analysis for image segmentation allows for greater flexibility of the environmental conditions in an outdoor environment. One of the greatest concerns for computer vision in an outdoor environment is the changing lighting conditions due to clouds, going under trees, etc.. Because we are identifying objects based on the physical property of texture, this system is more robust to changes in

lighting conditions as wouldn't necessarily be the case if we were identifying the objects based on their color. In addition, for our lawn application, color would not be an applicable discriminant to use because we are trying to differentiate the same lawn surface (i.e. grass) in two different forms (cut and uncut).We can also judge the flexibility of our system based on the training of the system. For all our lawn and sidewalk experiments, we trained the system only once. The system showed remarkable flexibility in being able to successfully segment the textures in the images for long experimental runs. For the sidewalk tracking experiments, all 7 runs were performed using the same training data set. Secondly, we have demonstrated that the wavelet transform provides a valuable tool for extracting textural features from images. While much research has gone into the use of wavelet transforms for textural analysis, no one has applied this research to the area of robotics as was done for this dissertation. The use of the wavelet transform was mainly confined to the task of analyzing samples of textures or samples of micro-textures.

The most significant contribution of this research is the development of a system capable of distinguishing between the cut and uncut surface of a lawn. In this manner we now have the necessary foundation for designing an autonomous lawn mower robot that is capable of mowing in a pattern as how a human would mow. Robotic lawn mowers up till now largely rely on randomness to mow an area. We now have the basis to construct an intelligent autonomous lawn mower robot. In addition, we have demonstrated how this research can be applied to other navigation tasks as well. In doing so, we have contributed to the research of outdoor navigation. For further information on this research, please visit http://www.mil.ufl.edu.

CHAPTER 8
FUTURE WORK

This study uncovers a few areas of subsequent research. First and foremost, this has to do with our lawn texture analysis system. Because of the hazardous nature of lawn mowing in general, we chose not to implement this system on an actual robotic lawn mower platform. Clearly, this should be done. A system similar to the one developed for the sidewalk texture analysis system could be put together. The main problem would be coming up with a platform large enough to carry all of the electronics. Also, we did not focus on the problem of initialization of such a system. In order for the robot to track the boundary between the cut and uncut lawn surface it must initially exist. This could be handled in one of two ways. Obviously one way would be for a human operator to make the initial pass on the outside perimeter of the mowing area. After that, the lawn mower could autonomously mow the rest of the mowing area. In addition, if the outside perimeter of the mowing area were of a different texture from the main mowing area, the mower could then use this information to make the initial pass.

Once an autonomous lawn mower robot has been implemented with the use of the lawn texture analysis system, this system could be extended to more than just a lawn mower. Instead of only recognizing the cut and uncut lawn textures, it could be trained to recognized ant hills, bare patches of earth, etc.. In this way if the robot recognized an ant hill, it could release ant killer or in the case of detecting a bare patch of lawn, it could release fertilizer to promote the growth of grass. Thus, the lawn mower now becomes a lawn maintainer.

One avenue for improvement would be in the area of wavelet subband determination. As was the case for both applications, we manually chose the appropriate subbands based on the criteria mentioned for each application. A system for automatically determining these bands would be

highly desirable. For instance, we could use other energy criteria for determining the frequency content of each wavelet subband and then see if they tend to agree with one another. Also, we could design a system whereby the user enters parameters about the particular application being performed. For instance, the user could be asked if their are strong horizontal or vertical features in the objects the application is supposed to recognize. In this way, we can easily tailor our system to different applications.

In terms of the texture analysis system discussed in this dissertation, we relied exclusively on greyscale images. It might be interesting to see if the discrimination can be improved by operating on the individual color channels or a combination of the color channels. This might yield a better classification system.

Finally, one should attempt different wavelet basis functions to see what effect they have on the classification of textural features. For this research, we relied on the LeMarie-Battle basis function to generate all subsequent wavelets. As stated earlier in this dissertation, wavelets (unlike Fourier transforms) have an unlimited number of basis functions. Therefore, this aspect should be further studied.

REFERENCES

[1] K. W. Abyoto, S. J. Wirdjosoedirdjo, and T. Watanabe,"Unsupervised Texture Segmentation Using Multiresolution Analysis for Feature Extraction," 東京情報大学研究論集 , Vol. 2, No. 1, pp 49-61, 1998.

[2] K. K. Chintalapudi and M. Kam, "A Noise-Resistant Fuzzy C Means Algorithm for Clustering," *IEEE World Congress on Computational Intelligence, Fuzzy Systems Proceedings*, Vol. 2, pp. 1458-1463, 1998.

[3] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis, J. Wiley and Sons, Inc., New York, 1973.

[4] D. Dunn and W. E. Higgins, "Optimal Gabor Filters for Texture Segmentation," *IEEE Transactions on Image Processing*, Vol. 4, No. 7, pp. 947-964, Jul. 1995.

[5] F. Espinal, T. Huntsberger, B. Jawerth, and T. Kubota, "Wavelet-Based Fractal Signature Analysis for Automatic Target Recognition," *Optical Engineering, Special Section on Advances in Patten Recognition*, Vol. 37, No. 1, pp. 166-174, 1998.

[6] A. Graps, "An Introduction to Wavelets," *IEEE Computational Science and Engineering*, Vol. 2, No. 2, pp. 1-18, 1995.

[7] K. J. Hakala, "An Autonomous Lawn Mower and Navigation System," Master's Thesis, Department of Electrical and Computer Engineering, University of Florida, 1997.

[8] A. K. Jain and R. C. Dubes, Algorithms for Clustering Data, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[9] A. K. Jain and F. Farrokhnia, "Unsupervised Texture Segmentation Using Gabor Filters," *Pattern Recognition,* Vol. 24, No. 12, pp. 1167-1186, 1991.

[10] A. Laine and J. Fan, "An Adaptive Approach for Texture Segmentation by Multi-channel Wavelet Frames," *Center for Computer Vision and Visualization,* Technical Report No. TR-93-025, Computer and Information Sciences Department, University of Florida, Aug. 1993.

[11] A. Laine and J. Fan, "Frame Representations for Texture Segmentation," *IEEE Transcations on Image Processing*, Vol. 5, No. 5, pp. 771-779, May 1996.

[12] A. Laine and J. Fan, "Texture Classification by Wavelet Packet Signatures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 11, pp. 1185-1191, Nov. 1993.

[13] C-T Li and R. Wilson, "Textured Image Segmentation Using Multiresolution Markov Random Fields," *IEE Colloquium on Applied Statistical Pattern Recognition*, pp. 8/1-8/6, 1999.

[14] J. Liu and J. C. Lee, "An Efficient and Effective Texture Classification Approach Using a New Notion in Wavelet Theory," *Proceedings of the IEEE ICPR '96*, pp. 820-824, Vienna, Austria, 1996.

[15] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, pp. 674-693, Jul. 1989.

[16] D. P. Mital, "Texture Segmentation Using Gabor Filters," *IEEE Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies*, pp. 109-112, Aug. 30 - Sept. 1, 2000.

[17] M. C. Nechyba, "Brief Introduction to Wavelets," *EEL6668 Intelligent Robot Systems class notes*, Department of Electrical and Computer Engineering, University of Florida, Fall 2000.

[18] M. C. Nechyba, "Introduction to Statistical Modeling," *EEL6668 Intelligent Robot Systems class notes*, Department of Electrical and Computer Engineering, University of Florida, Fall 2000.

[19] M. C. Nechyba, "Vector Quantization: a Limiting Case of EM," *EEL6668 Intelligent Robot Systems class notes*, Department of Electrical and Computer Engineering, University of Florida, Fall 2000.

[20] B. W. Ng and A. Bouzerdoum, "Supervised Texture Segmentation using DWT and a Modified K-NN Classifier," *IEEE 15th International Conference on Pattern Recognition*, Volume 2, pp. 545-548, 2000.

[21] M. Ollis, "Perception Algorithms for a Harvesting Robot," PhD Dissertation, The Robotics Institute, Carnegie Mellon University, Aug. 1997.

[22] R. J. Palmer, "Guiding Principles in Controlling Small Automated Vehicles," *Proceedings of the IEEE WESCANEX '95 Conference*, pp. 366-370, Winnipeg, May 1995.

[23] W. Pieczynski and A-N Tebbache, "Pairwise Markov Random Field and its Application in Textured Images Segmentation," *Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 106-110, Austin, Texas, 2000.

[24] G. Poggi and A. R. P. Ragozini, "Image Segmentation by Tree-Structured Markov Random Fields," *IEEE Signal Processing Letters*, Vol. 6, No. 7, pp. 155-157, July 1999.

[25] T. Randen and J. H. Husoy, "Filtering for Texture Classification: A Comparative Study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 4, pp. 291-310, Apr. 1999.

[26] T. Randen and J. H. Husoy, "Texture Segmentation Using Filters with Optimized Energy Separation," *IEEE Transactions on Image Processing*, Vol. 8, No. 4, pp. 571-582, Apr. 1999.

[27] R. M. Rao and A. S. Bopardikar, Wavelet Transforms: Introduction to Theory and Applications, Addison Wesley, Reading, Massachusetts, 1998.

[28] G. Ravichandran and M. M. Trivedi, "Circular-Mellin Features for Texture Segmentation," *IEEE Transactions on Image Processing,* Vol. 4, No. 12, pp. 1629-1640, Dec. 1995.

[29] O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE SP Magazine*, pp. 14-38, Oct. 1991.

[30] Friendly Robotics, Operating Manual for Robomower, Friendly Robotics, Irving, Texas, 2000.

[31] A. Sarkar, M. K. Biswas, and K. M. S. Sharma, "A Simple Unsupervised MRF Model Based Image Segmentation Approach," *IEEE Transactions on Image Processing*, Vol. 9, No. 5, pp. 801-812, May 2000.

[32] M. Unser, "Texture Classification and Segmentation Using Wavelet Frames," *IEEE Transactions on Image Processing*, Vol. 4, No. 11, pp. 1549-1560, Nov. 1995.

[33] M. Unser, "Texture Discrimination Using Wavelets," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 640-641, 1993.

[34] G. Van de Wouwer, "Wavelets for Multiscale Texture Analysis," PhD Dissertation, University of Antwerp, May 1998.

[35] M. Vetterli, "Wavelets and Filter Banks: Theory and Design," *IEEE Transactions on Signal Processing*," Vol. 40, No. 9, pp. 2207-2232, Sep. 1992.

[36] B. Wang, Y. Motomura, and A.Ono, "Texture Segmentation Algorithm Using Multichannel Wavelet Frame," IEEE 1997, pp. 2527-2532.

[37] J-H. Wang and C-Y. Peng, "Optimal Clustering Using Neural Networks," *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 2, pp. 1625-1630, 1998.

[38] R. Wang, "General Concepts of Pattern Recognition," *E186 Computer Image Processing and Analysis class handouts*, Harvey Mudd College, 2000.

[39] S. C. Zhu, "Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multi-band Segmentation," *Harvard Robotics Laboratory*, Technical Report No. 94-10, pp. 1-50.

BIOGRAPHICAL SKETCH

Rand Chandler was born in Fort Lauderdale, Florida, in 1973. He has earned a Bachelor of Science in electrical engineering degree from the University of Florida in 1995. He also earned a Master of Engineering in electrical engineering degree from the University of Florida in 1998 specializing in computer engineering. During his time in graduate school, he was a member of the Machine Intelligence Laboratory where he participated in the field of robotics. Also during this time, he was also a research assistant at the Center for Intelligent Machines and Robots, a research laboratory in the Mechanical and Aerospace Engineering department. During this time, he worked toward a Doctor of Philosophy degree.