Digital Image Processing in Measurement of Ice Thickness on Power Transmission Lines: A Rough Set Approach

by

Maciej Borkowski

A Thesis Submitted to the Faculty of Graduate Studies
in Partial Fulfilment of the Requirements for the Degree of

Master of Science
in Computer Engineering

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba R3T 5V6

Digital Image Processing in Measurement of Ice Thickness on Power Transmission Lines: A Rough Set Approach

by

Maciej Borkowski

A Thesis Submitted to the Faculty of Graduate Studies
in Partial Fulfilment of the Requirements for the Degree of

Master of Science
in Computer Engineering

Abstract

This thesis introduces a new system for measuring ice accumulated on power transmission lines. The main objective and difficulty is to gain enough information from a two dimensional image of wire covered with ice taken with a simple digital camera and to approximate the actual thickness of the ice covering the wire. The proposed solution combines two methods in a non-standard way. Rough Set Theory is used to process digital images. In the context of filtering digital images an extension of Rough Set Theory called the δ-mesh is applied to wire images. The δ-mesh is designed to combat noise in selected directions without blocking the important information in a wire image. In the second step, a classical digital image processing algorithm, namely, the Gouraud shading algorithm is used to approximate parameter values describing the dependency between actual and measured ice thickness from the image. The core of the system is based on extraction of information from a bare wire with known diameter to approximate the actual ice thickness. A sample application of a complete system to measure ice accumulated on power transmission lines is given. The contribution of this article is an approach to using rough set theory in digital image processing as an aid in directional oriented image filtering.

Keywords: Approximation, digital image processing, Gouraud shading, ice, linear interpolation, pattern recognition, power system transmission line, rough sets.

Acknowledgements

The author wants to gratefully thank for help and support from his supervisor Dr. J. F. Peters. The thesis would have never been finished without his valuable comments, directions and corrections.

This research project has been greatly helped by the comments, suggestions, information (e.g., Manitoba Hydro distribution system map, specifications for equipment, reports and data) and power systems equipment (e.g., sample wire and insulators) made available by Wes Moeller, Siu Cheung, Lyle Crowe, Mitra Tirandez and Dr. David Swatek at Manitoba Hydro during the past several years. This research has also been greatly aided by the generosity and insights of Prof. P.N. Shivakumar, Prof. W.H. Lehn, Prof. Z. Moussavi and Prof. M. Pawlak at the University of Manitoba; Prof. Z. Pawlak, Polish Academy of Sciences, Warsaw, Poland; Prof. Andrzej Skowron and Dr. Marcin Szczuka at the University of Warsaw, Warsaw, Poland; Prof. Zbigniew Suraj at the University of Information Technology and Management, Rzeszów, Poland; and, Wojciech Rzasa at Rzeszów University. Special thanks are extended to Hania Marczak for her help in taking pictures of transmission lines, helping to prepare experiments and gather information related to this project. The author also wishes to express his gratitude to Manitoba Hydro for its funding of this research project.

Table of Contents

List of Tables

List of Figures

1. Introduction

This thesis presents an approach to using a combination of rough set methods and classical digital image processing to estimate the thickness of ice accumulation on power system transmission lines. Rough set methods play a role in pre-processing digital images of conductors by filtering out noise in wire images. Rough sets were introduced in [1-2]. Classical rough sets present an approach to approximating, measuring and reasoning about subsets that belong to a finite universe. The basic idea was to partition a finite universe into subsets containing elements that are mathematically equivalent to each other. Recently, an approach to rough sets that makes it possible to measure closeness and inclusion of sets in an infinite universe (e.g., reals, typical sensor signal values) has been introduced [3]. This second approach to rough sets is briefly introduced in this thesis because it has been found to be useful in solving the ice thickness problem based on an analysis of pixels in a wire image. First, a method of partitioning subsets of the reals into subsets of equivalent elements is given. This is made possible by using an equivalence relation that was introduced in the early stages of this research [4]. The resulting partition of sets of reals in the plane (of an image) results in superimposition of a mesh over a wire image. By virtue of the way the mesh is constructed, each of the pixels in each cell of the mesh are in some sense equivalent. Second, a measure of inclusion of one set (e.g., set of pixels is a wire image) in another set (e.g., pixels belonging to a wire in a digital image) presented in [5] is briefly introduced in this thesis. It is this second approach to rough sets that provides a mathematical basis for the digital image filtering technique introduced in this thesis. The new filtering technique is also compared with classical digital image filtering. Also, this thesis includes a brief introduction to a toolset called δ-mesh which implements the approach to partitioning sets of values associated with a signal.

Application of classical digital image processing methods introduced by H. Gouraud [6] and others [7-8] provide a basis for measuring wire diameters and the extent of accumulation of ice covering transmission lines. The approach presented in this thesis makes use of the Gouraud shading algorithm in a context different from the one originally envisioned by Gouraud. That is, the Gouraud algorithm is not used in its

classical application, namely, to produce shaded pictures of curved surfaces. Instead the Gouraud algorithm proves to be very effective tool in approximation of internal adjustable system coefficients used in the analysis of power system transmission line digital images. Therefore, the use of the basic idea in Gouraud's shading algorithm is part of what can be described as a pattern recognition technique rather than classical digital image processing. This thesis gives a detailed formulation of the ice-thickness measurement problem relative to the identification of non-wire and non-wire pixels in digital images of transmission lines. It has been found that it is possible to obtain an accurate estimate of the thickness of ice accumulation on a wire based on an analysis of wire images obtained with a low-cost digital camera. Remarkably, it is a shading method introduced by Gouraud in 1971 that provides a key component in the ice-measurement method introduced in this thesis. As already mentioned, the Gouraud algorithm provides a form of pattern recognition useful in the identification of patterns in wire images. It should be noted that in the study of patterns in wire images, the application of the basic idea contained in Gouraud's shading algorithm did not include the initial conditions needed to process wire images, namely, there is no triangle net set. To meet this requirements, two separate algorithms for creating a net of triangles from a given set of calibrating points (images) has been introduced in this thesis. Each algorithm tries to utilize different properties of the coefficients' space to make the approximation of ice-on-wire measurements more efficient. That is, the first algorithm can be seen as a static algorithm. It tries to create from a given set of points a partition made from triangles, such that the length of the triangles' sides is possibly shortest. The second algorithm is dynamic. Each time a new testing point is considered, a new triangle for this testing point is found. When a triangle is found, it is the closest triangle to a given point, and which encloses given point.

After a formulation of the ice-measurement problem solved in this thesis, each of the measurement algorithms used in this work are presented. For readability and future object-oriented implementation, the C++ code for these algorithms has been given. For prototyping, digital image processing and plotting purposes, these algorithms have been implemented in a toolset called the Transmission Line Ice Measurement (TLIM) system. The TLIM system has been completely written in version 6 of Matlab. For user

convenience a Graphical User Interface has been designed to make it a straightforward task to use the TLIM system without need to understand all of the underlying theory.

Finally, this thesis includes a fairly comprehensive study of set of wire images. Various measurement experiments with wire images, especially images containing ice-coated wire, are presented in this thesis. These experiments were performed on a collection of wire images taken during the past 12 months as well as on photographs of ice-clad transmission lines during the 1998 ice storm in Eastern Canada. The cumulative effect of these experiments provides a demonstration of the validity of the approach to ice-measurement and of the TLIM system. It should also be mentioned that one of the goals of this research has been to formulate a solution to the ice-thickness measurement problem in such a way that it can be easily implemented by a power system transmission line maintenance engineer or a robotic device using an ordinary digital camera connected to some form of computer such as a laptop such as an IBM Thinkpad or personal digital assistant such as a Compaq i-pack. In most experiments performed for this thesis, a digital camera has been used as a device to record images. However, the TLIM system capabilities of processing images are not limited to images made by digital cameras. The TLIM system can process digital images from any source, resolution and number of colors with the condition that the calibration and testing phases have been performed with the same device.

References

[1] Z. Pawlak, Rough sets, International Journal of Computer and Information Sciences, vol. 11, 1982, 341-356.

[2] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data. Boston, MA, Kluwer Academic Publishers, 1991.

[3] Z. Pawlak, J.F. Peters, A. Skowron, Z. Suraj, S. Ramanna, M. Borkowski, Rough measures: Theory and Applications. In: S. Hirano, M. Inuiguchi, S. Tsumoto (Eds.), Rough Set Theory and Granular Computing, Bulletin of the International Rough Set Society, vol. 5, no. 1 / 2, 2001, 177-184.

[4] J.F. Peters, S. Ramanna, Z. Suraj, M. Borkowski, Rough neurons: Petri net models and Applications. In: L. Polkowski, A. Skowron (Eds.), *Rough-Neuro Computing*. Berlin: Springer, 2002, 472-491.

[5] J.F. Peters, A. Skowron, Z. Suraj, M. Borkowski, W.Rzasa, Measures of Inclusion and Closeness of Information Granules: A Rough Set Approach. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 304-311.

[6] H. Gouraud, "Continuous Shading of Curved Surfaces", IEEE Transactions on Computers, vol. 20, June 1971, pp. 623-629. , June 1971

[7] P.V.C. Hough, Methods and means of recognizing complex patterns, U.S. Patent 3,069,654, 1962.

[8] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Prentice Hall, Upper Saddle River, New Jersey 07458, 2002.

2. Problem Definition

In this chapter we discuss more precisely the problems arising from an approach to using digital camera images as input sources in a system to estimate either wire thickness or the thickness of ice accumulated on a power system transmission line. Before we proceed it is important to mention one fact that makes this approach more general. To make the work clearer, we concentrated only on pictures taken from one type of digital camera, namely, a SONY CD Mavica Digital Still Camera. A camera is shown in Fig. 2.1, more specifications are given in Appendix E.

Fig. 2.1 Digital Camera Used For the Project.

However, it should be noted that all algorithms presented in this thesis can be applied to digital images obtained in other ways (e.g., scanned photographs taken with an ordinary camera). There are some properties specific for the SONY CD Mavica Digital Still Camera used in thesis, which may not exist when using a scanner or other types of digital cameras. When this takes place it will pointed out and a solution will be shown, which is independent of this problem about special camera features. Regardless of digital imaging device used all images must have a maximum of 256 grey levels.

2.1. Towards a Solution of the Ice Thickness Measurement Problem

First, consider a precise definition of the problem we have to solve and list of all possible means we are allowed to use to solve this problem. Later on, we divide the main task into smaller, easier-to-manage sub-problems.

Problem 2.1. Estimate the thickness of ice coating on a power system transmission line using only a digital image of the line taken from 10-500 meters (i.e., Fig. 2.2).

Fig. 2.2 Sample Wire Image

From the formulation of Problem 2.1, we see that all we have is a picture (i.e., a digital image) with no additional information. In particular, we do not know the distance to the wire by looking at the wire image. This makes the measurement of either the wire diameter or the thickness of ice accumulation on a wire difficult. Different distances result in *apparent* different sizes of the same wire in a picture and, yet, the thickness of the wire itself is always the same. In addition, we do not know by looking at a digital image the camera settings such as zoom factor or image resolution of the camera. Thus,

given a wire image such as the one in Fig. 2.2, we cannot trust any additional information other than the pure image. With an ice-coated wire such as the one shown in Fig. 2.3 (found in [1]), we must rely on a count of the number of pixels as a basis for measuring ice thickness on a transmission line.

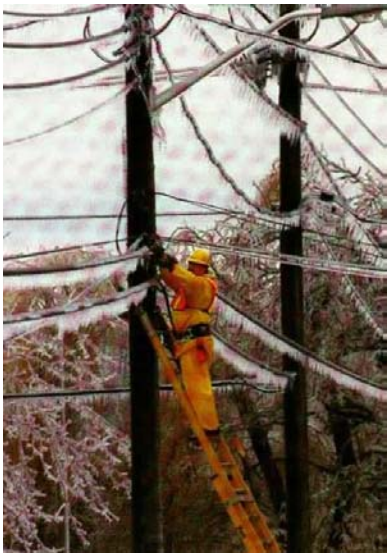


Fig. 2.3 Ice-coated Transmission Lines [1]

Once the number of pixels for an ice-coating wire is known, then there is still the unsolved problem of converting a pixel count to centimetres or some other unit of measurement. Therefore, such a problem requires one more pre-step before we could actually worry about measuring ice-thickness. A pre-step should let us determine the proper factor to convert pixel values into real world measurement units. In this thesis, an algorithm to facilitate automatic calculation of this factor is proposed and implemented in a complete Transmission Line Ice Measurement (TLIM) system.

The information contained in every image depends on various factors. To name only few important ones, we need to take into account a variety of factors such as

- Distance from an object when taking a picture,
- Location of an object,
- Scene light (i.e., lighting conditions at the time that a picture is taken),

- Zoom used,
- Image resolution,
- Film speed,
- Exposure time,
- Aperture (i.e., variable opening by which light enters a camera).

If the values of all of these factors are known, it is a straightforward task to convert the pixels in any photographed ice-thickness into standard measurement units. Unfortunately, all of these information are not usually known. Hence, we can only measure ice thickness from an image in pixels. We call this measurement **O**bserved **I**ce **T**hickness (OIT). From what has been observed so far about a digital wire image, it is known that *OIT* is a function of several variables as in Eqn. (2.1).

$$OIT = f(\textit{ ice thickness, distance, location, light, zoom, ...}) \qquad (2.1)$$

The only known variable is *OIT* (the dependent variable in Eqn. (2.1)), and the only variable, which we would like *OIT* to depend on, is *ice thickness*. It is obvious that such a formulation of the problem gives us no chance to succeed. We need to introduce some simplifications, which may cause some error to appear, but it is possible to solve the simpler problem with good accuracy.

First, consider the combination of three factors: distance from an object, image resolution and zooming. Notice: when looking at the picture such as the one in Fig. 2.3, we cannot tell what zoom was used nor how far the camera was from the photographed object**s**. Consequently, it is not necessary to treat these factors separately, but rather as one factor. Similar argument applies to resolution. For this reason, only one factor called *magnitude* is used instead of three factors (i.e., distance, resolution, zoom) to compute the OIT.

In the design of a model for OIT, next consider: light, exposure time and aperture. This case is not so straightforward as the previous one. These factors have different influence on picture and it is probably easier to estimate their values from selected features of an image. Indeed, the more light in a scene the brighter picture we obtain, and

the longer exposure time the brighter picture as well as the greater contrast. And finally, aperture controls the range where objects are sharp. Even though these factors are nicely explainable, they introduce a huge space of possible conditions. We do not want to be distracted by them but rather it is more effective to consider characteristic of a camera. Since all humans perceive light in a similar way, cameras are made to create pictures with some standard brightness. Therefore, all three listed factors are used by a camera itself to make a "good looking" picture for the human eye. This means that the resulting image is "normalised" in some way, making it *independent* of scene light and exposure time. Of course, nothing can be done for two pictures, where one is taken at noon and the other at midnight, but this is an extreme example and will not be considered in the design of the prototype of the TLIM system. As a result of what has just been discussed, we do not have to consider scene light and exposure time as factors. What has left is aperture. This cannot be simplified too much. Especially when considering ice edge, we have to be very careful about how much of and what might termed the quality that this edge is recorded, i.e., as a sharp edge or blurred image. This factor is included in the OIT model and is given the name *aperture* and is responsible for aperture setting and average brightness produced by camera. This factor is totally camera-dependent and must be adjusted for each camera.

The location factor denotes all changes resulting from relative differences of an object and camera positions. In particular, *aperture* can denote the angle at which an object is seen. For any object, we can discern three different angles describing location of the object relative to a camera. For a wire covered with ice the situation is fairly simple. The wire with ice might be modelled by a cylinder with a small diameter. This reduces its description to only two angles (since rotation along a wire centre makes no difference). In order to reduce the description even more, we require taking pictures such that the wire is perpendicular to the direction in which the camera is pointed. This is not a very restrictive assumption. Usually, an inspection vehicle or robot is moving along transmission lines (on one side or beneath the lines) and any particular point on a wire can be photographed without violating this assumption. Advantages coming from this restriction are very important: we can use only one number to describe wire location, namely, its angle with respect to the image borders, and the wire is approximately at the same distance from the

camera and is not deformed by panoramic projection. The latter assumption is not true for the whole picture, especially on the sides in an image, but taking into account distance between wire and camera can be considered to be true in the middle part of an image.

Hence, we end up with four independent variables: *ice thickness*, *magnitude*, *location* and *aperture*. Thus Eqn. (2.1) can be simplified as in Eqn. (2.2).

$$OIT = f(\textit{ice thickness}, \textit{magnitude}, \textit{aperture}, \textit{location}) \tag{2.2}$$

This formula is much simpler than Eqn. (2.1), but we are still unable to solve it. In order to do so, we need to consider two more facts. First, all of the four variables in Eqn. (2.2) are independent of each other, in particular *ice thickness* is independent of all other factors. Second, from basic knowledge of camera construction and basic algebra we know that *magnitude* operations (by which we mean zoom, distance and resolution) are linear operations on a resulting picture. That is, these magnitude operations are linear, but still dependent on aperture and location. That is why Eqn. (2.2) can be rewritten as Eqn. (2.3).

$$OIT = f(\textit{aperture}, \textit{location}, \textit{magnitude}) * \textit{ice thickness} \tag{2.3}$$

At this point the OIT model cannot be simplified any more. In Eqn. (2.3), all of the required factors needed to measure ice thickness are important and removing any one of them would cause too big an error increase. On the other hand, from the above discussion it is clear that the introduced assumptions are reasonable and will not introduce too much error. For simplicity, we use the notation *SF* (**S**cale **F**actor) to denote the function *f*(*aperture, location*, *magnitude*).

From Eqn. (2.3) we can calculate *ice thickness* by computing *OIT*/*SF*. The only problem that must be solved is how to obtain *SF*-values. Eqn. (2.3) contains the beginning of a solution to the problem of how to compute *SF*. That is, solving Eqn. (2.3) for *SF*, we obtain *SF*=*OIT*/*ice thickness*. However, we cannot use ice since we do not know its thickness. Assuming that *SF* is the same for all objects in the same range, then it is enough to know anything with known dimensions to learn *SF*. In our case, the best

choice is the wire itself or a neighbouring object of known size such as a transmission line insulator. We know the diameter of the wire and we know its shape, therfore from all angles the wire will have the same thickness. Thus, we find *SF* from Eqn. (2.4).

$$OWT = SF * \textit{wire thickness} \qquad (2.4)$$

where *OWT* denotes **O**bserved **W**ire **T**hickness and it is measured directly from a picture.

This discovery leads us to a definition of another, easier problem.

Problem 2.2. Find out the Scale Factor (*SF*) relative to a picture of a wire of known diameter (*wire thickness*) taken from 10-500 meters.

Now, solution to Problem 2.1 must follow the solution of Problem 2.2. The complete ice measurement algorithm is presented next.

Algorithm 2.1 (A method to solve Problem 2.1)

1. Take a picture of a wire (partly bare and partly ice-covered).
2. Measure *OWT*(Observed Wire Thickness) and *OIT*(Observed Ice Thickness).
3. Find out Scale Factor from equation 2.4, given *wire thickness*.
4. Find out ice thickness from equation 2.3 where
   $f(aperture, location, magnitude) = SF$.

The partly bare wire in step 1 of Algorithm 2.1 provides a basis for a set of experiments used to validate the prototype of the TLIM system toolset. The bare wire in step 1 provides a frame-of-reference for preliminary ice-thickness measurements. Although it might be difficult to find in real life a transmission line that is partly bare and partly ice-covered. However, this method is not bounded to wires only. Instead of wire we can use any object (e.g., an insulator) which has known dimensions. The steadfast feature of a wire image in step 1 of Algorithm 2.1 is the presence of cylindrically shaped objects, as the transmission line wire dimensions are approximately the same from any angle. For

transmission lines we can use, for example, insulators instead of bare wire as a frame-of-reference. Insulators are cylindrical in shape and are always in place on transmission lines. As can be seen from what has been said so far, the proposed measurement method is quite powerful and independent of the object being measured.

2.2. Image Processing

In the last two subsections, we discussed some general issues related with the problem of measuring ice thickness on a transmission line wire or conductor using a camera image. This problem was divided into two smaller problems and it was pointed out that the first step is to process a picture with an object with known dimensions. This raises a question about the precision of measurements of wire thickness. This subsection is devoted to some preliminary problems related to this issue. What we are interested in is only wire thickness given in pixels. We will not use Scale Factor nor will we use the fact that we know the actual value of a wire diameter.

Fig. 2.4 Sample of Wire

Fig. 2.5 Zoom To Left End of Wire

A sample image of a transmission line is shown in Fig.2.4. This is the image we will be working with in this section. The wire in Fig.2.4 has the appearance of a straight black line clearly distinguishable from the sky. It may make the impression that if we want to know its thickness, all we have to do is to count black pixels across the wire. In fact, problem is much more complex. In the Fig 2.5 we see the same picture, but zoomed on the left side of the image. This zoomed-in image of a wire shows that a wire consists

not only of black-coloured pixels, but also of pixels with many shades of grey. In the following sections, some more issues related to the nature of digital images are discussed.

2.3. Image Compression Problem

Due to large memory requirements of digital images various methods of lossy compression are used by a digital camera. Instead of many advantages, such compression techniques also introduce some additional information into an image. This problem is specially visible in digital images of solid shapes with sharp edges. For example, the wire in Fig. 2.6 is surrounded by a whitish "shadow" that has no meaning in the real world and is the pure effect of the compression algorithm used by the digital camera used to take the picture of the wire. This phenomenon makes it difficult to distinguish between a wire and the sky. It is assumed that the sky is a uniform area with a small rate of change in colours. The wire "shadow" does not belong to a wire, but is a common occurrence in pictures with quite high changes in brightness. The presence of a shadow surrounding a wire image can result in a blurred transition between the wire and the sky.

There is a solution to the problem of white shadows due to digital compression algorithms. The software options for most digital devices make it possible to choose non-lossy format (e.g., tiff, targa or bmp). It produces files a few times larger, but the "shadow" problem is totally solved. For the TLIM system only part of an image is needed, which makes the amount of data to be stored reasonable. On the other hand, if a digital camera or some other digital device has software that makes it possible to set the compression level, it is enough to set the compression level to the best quality. Usually, the obtained image is clear enough to be processed by the TLIM system. In the experiments described in this thesis, jpeg compressed files were successfully used yielding low error-rate classification results. For example, in experiment described in section 5, where a scanned photograph was used, error rate obtained for ice measurement was 9.5%.

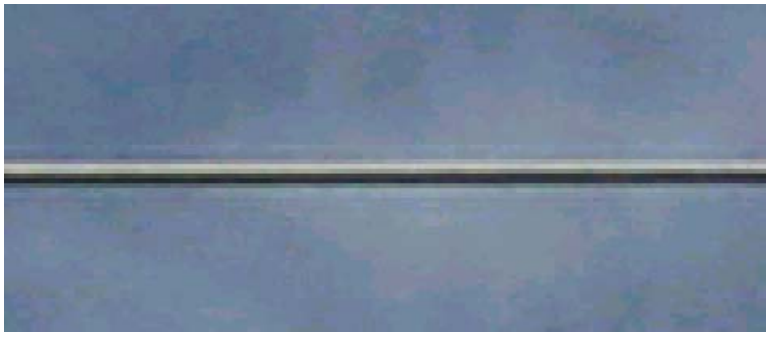


Fig. 2.6 Two Colour Wire And JPEG Compression.

2.4. Image Brightness Problem

Depending on the sun brightness and its position a wire image can consist of two colours (see Fig. 2.6). This makes the task of finding out which pixels belong to the wire much more difficult. Instead of one type of transition from a black to blue, we have second type, namely, a transition from a white to blue. In addition, there is one more transition within a wire between its black and white parts. These non-black parts of a wire image do not carry any useful information for us.

Since there is no straight forward software solution for this image brightness problem it is suggested that in taking pictures of transmission lines to locate the sun and make sure the wire is totally in shadow. This can be achieved in two ways: by choosing days with presence of clouds in the sky or by positioning the camera so that the sun is behind the wire. These methods can significantly reduce the wire reflection effect, although these methods cannot remove it completely. In the following chapters, we will see some remainings of this phenomena and a software solution.

2.5. Non-parallel Wire Image Problem

A transmission line in an image does not have to be perfectly straight nor do the edges of a wire have to be parallel. The first problem is related to the actual shape of wires stretched between power towers. The wire is not straight, but curved (deflected) because of earth's gravity. The second problem is entirely caused by camera perspective.

We know that the edges of a wire are perfectly parallel, but if one side of the wire is closer to the camera than the other, it looks a slightly wider. If this happens, a Scale Factor is not going to be the same for different parts of an image. This problem has to be avoided by some means.

The solution to the non-parallel wire image problem has two steps. First, at the time of taking a picture, the camera must be situated in a perpendicular direction to a wire. This eliminates the distorted wire shape problem. There is not much we can do about line curvature or deflection though. Thus, this is the subject of the second step. While selecting an area of a wire image to be measured, we should choose only a small part of the picture. A short wire has very small curvature (i.e., effectively straight) and when considering only a piece of one-meter long, for example, the curvature of a suspended wire can be neglected. The selection of a short wire image segment also combats the problem of non-parallel edges and the wire will have the same thickness on the entire image. This is a reason why in experiments described in this thesis only short wire-image segments were selected to calibrate the TLIM system. The average size was approximately 64-80 pixels for the longer side of the image. An alternative to wire segment image selection is to use image warping. That is, a line curvature can be calculated from mathematical equations and an image can be warped to yield a straight wire. It should also be noted that these manual wire image segment selection procedures can be automated, for example using thresholding [4-5] and Hough transform [2-3].

2.6. Picture Axis Problem

It may happen that the wire in a digital image is not parallel to any picture axis. That is, at different cross-sections a wire may consist of different number of pixels. Further, the wire pixels may not be black and white, but instead consist of a full range of colours. Therefore, it is hard to tell which pixels are part of a wire and which belong just to the sky.

The solution to the picture axis problem cannot change the fact that a wire's boundary is spread across more than one pixel. This is not how we would imagine an image produced by digital camera with response to an ideal zero-length transition

between dark wire and bright sky. There are other processes involved during picture taking with a digital camera such as image sharpening by a camera itself. So we must learn how to use this information during the TLIM system creation process. What can be done though is to design the TLIM system with a mechanism that responds to the fact that a wire is not parallel to image horizontal borders. To cope with this problem, the TLIM system must rotate the image so that it is parallel with the horizontal border of an image. Afterward, the resulting wire will have the same cross-section at any point, provided that the image was taken to satisfy all previously stated requirements.

References

[1] I. MacAlpine, Bell Canada lines coated with ice on Counter Street in Kingston, Ontario on January 9, 1998. In: M. Abley, *The Ice Storm: An Historic Record in Photographs of January 1998*. CA: The Montreal Gazette, 1998, p. 126.

[2] P.V.C. Hough, Methods and means of recognizing complex patterns, U.S. Patent 3,069,654, 1962.

[3] R.O. Duda, P.E. Hart, Use of the Hough transformation to detect lines and curves in pictures, Comm. ACM, 1972, vol. 15, no. 1, 11-15.

[4] R.C. Gonzalez, R.E. Woods, Digital Image Processing, 2nd Ed. NJ: Prentice-Hall, 2002, 595-612.

[5] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification. New York: John Wiley & Sons, 2001.

3. Measurement Method

We have already seen some preliminary discussion in chapter 2 concerning pre-processing of digital images as input for a digital image processing system used to measure the ice accumulations on a power transmission line. In this chapter, we assume that images are already prepared to satisfy all previously mentioned requirements, namely,

- Each image contains only wire with the sky in the background with no other objects;
- The angle between wire and horizon must be in a range -45º, 45º;
- Each pixel is described by one byte;
- A greyscale level is represented by 0 for black pixels and 255 for white pixels.

We concentrate now on the best method to measure precisely wire thickness based on pixel-based analysis of a wire in a digital image. The principal objective is to obtain better than one pixel precision from an input image.

An informal presentation of the idea of measuring the degree that one set of pixels is approximately a subset of another set (this is usually called a measure of rough inclusion [1-3] and will be defined in section 3.2.) is presented in this section. In rough set theory, different forms of what are known as rough membership functions are used to measure rough inclusion [4]. Rough set theory [5] itself is briefly explained in Appendix A. It is a particular form of rough inclusion measure that has been found useful in solving the particular digital image processing problem presented in this thesis. Before the form of rough inclusion measure used in this thesis is presented, a brief description of the digital image problem space is given in this section.

3.1. Setting for Measuring Rough Inclusion of Pixels

Given an image of a wire, we need to locate the edges of the wire. The easiest way to do this is to identify pixels which belong to the wire and to the sky. The edge of

the wire in an image must be just between the other two sets of pixels, namely, "sky" pixels and "wire" pixels. However, some pixels are difficult to classify. That is, some pixels are grey, so they may partially belong to the wire and partially to the sky. Because of this classification problem, the transition region between sky and wire in a digital image can be wider than one pixel. Hence, the pixels in a digital image containing a transmission line wire with the sky in the background can be labelled *wire*, *partially-wire-partially-sky*, *sky*. Therefore the problem of determining the diameter of a transmission line shown in a digital image can be formulated in the following way.

Problem 3.3. For each pixel in a picture of transmission line wire, determine the degree that the pixel belongs to the wire.

Thus, we are looking for a measure of rough inclusion of digital image pixels in wire-pixels that is a measure $\mu_w : \mathbb{N} \times \mathbb{N} \to [0,1]$, where the input space is a given picture. For example, consider the picture in Fig. 3.1(b), where the input space is $(m,n) \in \mathbb{N}^2$ and $m,n \in [0,255]$), and μ_w computes values close to 0 for pixels belonging to the sky and values close to 1 for pixels belonging to the wire. A plot of rough inclusion measurements for each pixel is given in Fig. 3.1(a).

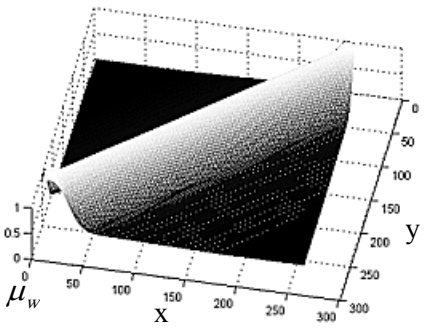


Fig. 3.1(a) Rough Inclusion Measurements

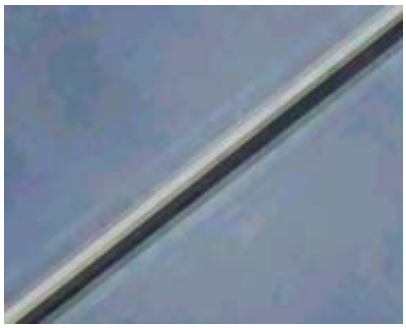


Fig. 3.1(b) Image of a Wire

A space of all possible rough membership functions is huge. If we do not use any properties of a wire, we will have too many parameters to set to make this problem solvable. Therefore, we use the assumption, that we suggested earlier, namely,

Assumption. A wire is straight in the entire picture and the sides of the wire are parallel.

In practice, this is a reasonable assumption, since a section of a digital image of a deflected (bent) wire may be selected in such a way that the wire in the selected image is approximately straight. As a result, there is a direction for which the measurement of rough inclusion of pixels in the set of pixels representing a wire is constant. This makes it possible to reduce the three-dimensional problem into a two-dimensional problem. This results in a significant reduction of the search space used to solve the wire measurement problem. Now, we can consider only the cross-section of a wire, because any cross-section at any point is the same.

At this stage precision is crucial in measuring the inclusion of any pixel in the set of wire pixels in the digital image of a transmission line. Hence, an extension of rough set theory is introduced to measure pixel inclusion in a set of pixels associated with a wire.

3.2. δ-mesh

In this section, a rather straightforward approach to partitioning sets of reals into subsets of equivalent real-values is introduced in the context of classical rough set theory [5]. This partition is called a δ-mesh, where δ is a parameter used to control the size of mesh cells. To facilitate understanding of the basic approach to measuring the degree of inclusion of one set of pixels in another set of pixels in a digital image, a very brief introduction to idea of a δ-mesh is given in this section. A brief introduction to one form of measure of inclusion is also incuded in this section. Measurements of pixel-inclusion are carried out in the context of a δ-mesh.

To begin, let $IS = (U, A)$ be an infinite information system where U is a non-empty subset of the reals $\Re$ and A is a non-empty, finite set of attributes, where $a : U \rightarrow V_a$ and $V_a \subseteq \Re$ for every $a \in A$, such that $\Re \supseteq V = \bigcup_{a \in A} V_a$. Let $a(x) \geq 0$, $\delta > 0$, $x \in \Re$ (set of reals) and let $\lfloor a(x)/\delta \rfloor$ denote the greatest integer less than or equal to $a(x)/\delta$ (also called the "floor" of a(x)/δ). The parameter δ serves as a "neighbourhood" size on real-

valued intervals. Reals within the same subinterval bounded by $k\delta$ and $(k+1)\delta$ are considered indistinguishable because the elements belong to the same subinterval of reals.

Definition 3.1 ***Indistinguishability Relation.*** For each $B \subseteq A$, there is an associated equivalence relation $\mathrm{Ing}_{A,\delta}(B)$ such that

$$Ing_{A,\delta}(B) = \{(x, x') \in U^2 \mid \forall a \in B. \lfloor a(x)/\delta \rfloor = \lfloor a(x')/\delta \rfloor\}$$

Proposition 3.1 $\mathrm{Ing}_{A,\delta}(B)$ is an equivalence relation.

The notation $[x]_B^\delta$ denotes equivalence classes of $\mathrm{Ing}_{A,\delta}(B)$. Notice that the parameter δ in the definition of the relation Ing makes it possible to adjust the coarseness or "granularity" of a partition of the subinterval of reals (universe) over which sensor signals (or digital image pixels) are classified. The relation Ing was introduced in [6].

Further, partition $U/\mathrm{Ing}_{A,\delta}(B)$ denotes the family of all equivalence classes of relation $\mathrm{Ing}_{A,\delta}(B)$ on U. This partition is called δ-mesh. The δ-mesh was introduced in [7] and applied in [9-11]. For $X \subseteq U$, the set X can be approximated only from information contained in B by constructing a B-lower and a B-upper approximation denoted by $\underline{B}X$ and $\overline{B}X$, respectively, where $\underline{B}X = \{x \mid [x]^\delta_{B\cup Id} \subseteq X\}$ and $\overline{B}X = \{x \mid [x]^\delta_{B\cup Id} \cap X \neq \varnothing\}$. In cases where instead of using x we use sensor reading y, we create equivalence class consisting from all points for which sensor readings are 'close' to y and define $[y]^\delta_B = [x]^\delta_B$ for such x that $a(x) = y$.

Definition 3.2 ***Measure of Inclusion.*** Let $S = (U, A)$ be an information system with non-empty set U and non-empty set of attributes A. Further, let $B \subseteq A$ and let $[y]^\delta_B$ be an equivalence class of any sensor reading $y \in \Re$. Let ρ be a measure of a set $X \in \wp(U)$, where $\wp(U)$ is a class (set of all subsets of U). Then the rough inclusion set function $\mu_y^{B,\delta} : \wp(U) \to [0,1]$ is defined in (3.1).

$$\mu_y^{B,\delta}(X) = \frac{\rho\left(X \cap [y]_B^\delta\right)}{\rho\left([y]_B^\delta\right)} \tag{3.1}$$

for any $X \in \wp(\mathrm{U})$.

Definition 3.2 is slightly different from the original definition where the argument of the rough membership function is an object x and the set X is fixed [1].

Example 3.1 Sample δ-mesh

For visualisation of a δ-mesh, see Fig. 3.2 that displays a sample δ-mesh provided by a tool developed for δ-mesh applications. In the main panel of Fig. 3.2 is a plot of a real function with a δ-mesh superimposed on it. The cell size can be adjusted independently for each dimension. This makes it possible to control amount of information we want to preserve in a δ-mesh cell before further processing. In this new space, each point represents information collected from other "equivalent" points in a δ-mesh cell, which are close enough. In other words, if in the original space any two points happen to be placed in the same δ-cell, they are considered to be indistinguishable. Changing parameter δ makes it possible to adjust the sensitivity to any required level.

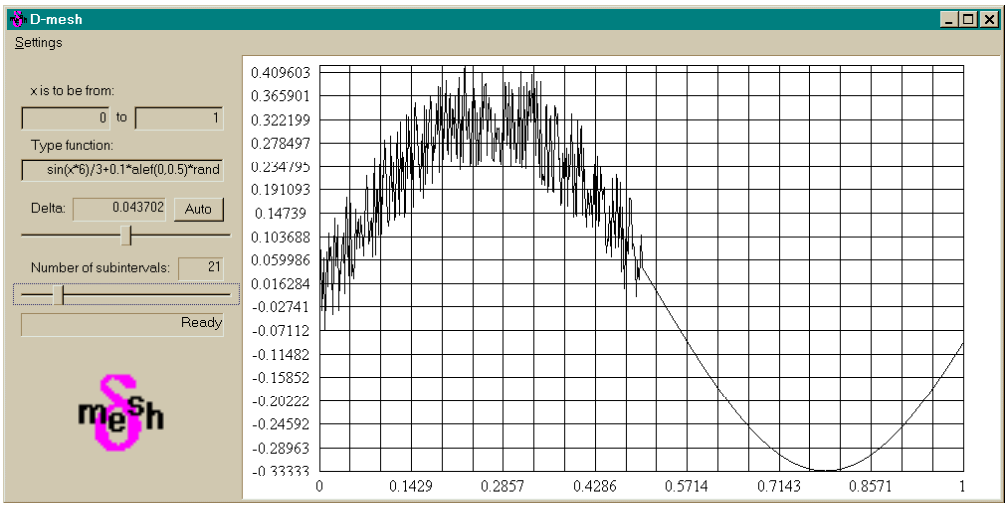


Fig. 3.2 A δ-mesh Tool.

Since the operation of applying δ-mesh causes to the loss of some information, it can have a beneficial side-effect in the case of a noisy signal inasmuch as the signal information contained in a δ-mesh cell can lead to a decrease in the noise level of a selected signal segment. Although, there are several applications of the δ-mesh (see [9-11]), we go no further with δ-mesh theory in this thesis. Instead, we concentrate on the use of a δ-mesh to control the noise level in making digital image measurements. That is, we will adjust mesh cells in certain way to preserve information we need and remove unwanted noise.

Example 3.2 δ-mesh calculation

In this example we will see, step by step, how a δ-mesh is applied to the digital image containing the wire. This application is somehow similar to applying the low pass filter, but the user has more parameters to control the direction in which information is preserved. For this thesis, δ-mesh is considered as a structure based on indistinguishability relation, which results in square mesh cells. In general case, mesh cells can have any particular shape, depending on characteristic of objects trying to be distinguished. For wire edges, the most important factor is the sharp transition in one direction. Therefore, squares with one side longer than the other where used.

Three parameters controlling δ-mesh are: vertical and horizontal mesh size and the angle, by which the mesh is rotated. The angle together with selection of longer mesh side decide where the process of noise reduction must be performed. Consider the image from figure 3.3. This is the input image of size 78 horizontal pixels by 44 vertical pixels. The same image, after levelling the wire is shown on figure 3.4. The minimum pixel's values have quite big variability and it is difficult to determine exact wire edges or the exact wire angle.

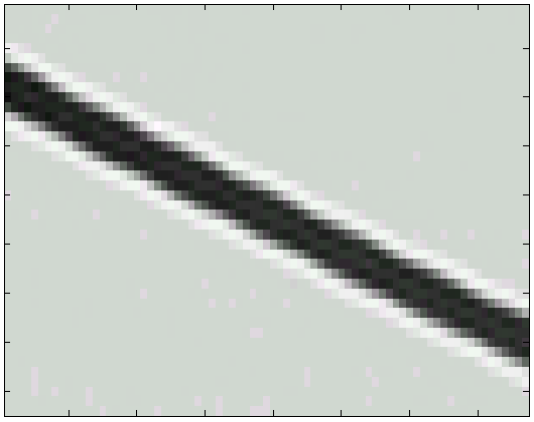


Fig. 3.3 Sample image with wire.

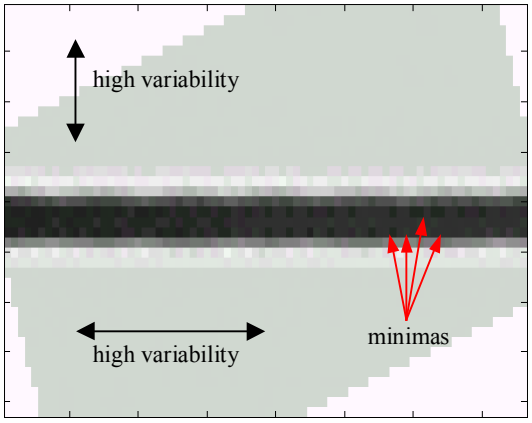


Fig. 3.4 Simple rotation. Image size 78×44.

In figure 3.5, we see the same image, but after applying δ-mesh of such adjusted mesh sides that in each direction we have 32 cells. As we can see, because of the fact, that the δ-mesh was applied in direction of the wire, image from figure 3.5 characterises with high variability in vertical direction and small variability in horizontal direction. This results in more precise measurement for measuring the wire edges. On figure 3.6, we see the same image, but this time in horizontal direction number of cells is 16 and in vertical 32. We preserved information needed for edge detection and removed the noise along the wire.

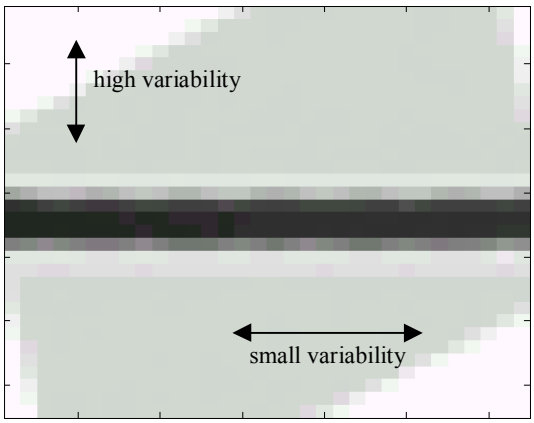


Fig. 3.5 δ-mesh applied. Image size 32×32.

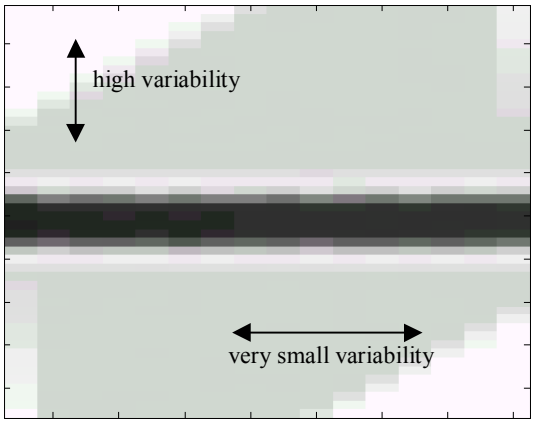


Fig. 3.6 δ-mesh applied. Image size 16×32.

The appropriate size of mesh cells were found during experiments. The system was run with various setting and error was calculated. The best setting were 32×32 for finding the angle and 16×16 for calculating angle criterion function.

The superimposing of δ-mesh is done in two stages. First, all points belonging to the same cells are localised. Then a point representing each cell is being calculated. In order to introduce noise attenuating effect, the colour of representing pixel is set as an average of all pixel's colours from given mesh cell. Then a new image, with smaller resolution, consisting from these averaged pixels is created.

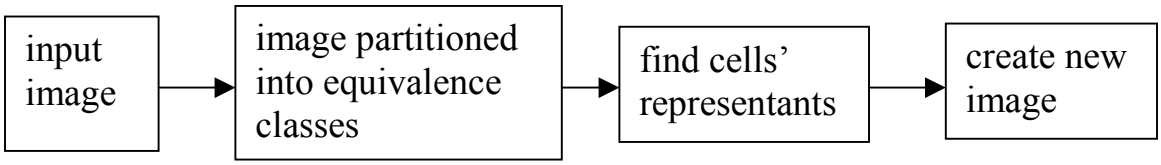


Fig. 3.7 Overview of δ-mesh steps.

3.3. Linear Interpolation

It is usually the case that a digital image of a conductor is tilted (not horizontal). Estimates of wire diameter and ice thickness on a transmission line are facilitated by adjusting the angle of a wire image such that it becomes horizontal (i.e., parallel with the horizontal axis of a δ-mesh containing a wire image). Before we show how to find the right angle, consider the process of rotating itself. In Fig. 3.8 a sample image and its rotation are shown. The image is an example of a low-resolution image – the same kind of low-resolution image we can expect to obtain from an average digital camera. In order to see differences in rotation methods, consider the test image in Fig. 3.8(a) that contains two words, one-stroke figures (circle and lines) and a regular grid of points. In the Fig. 3.8(b)we can see result of simple rotation of 36 degrees.

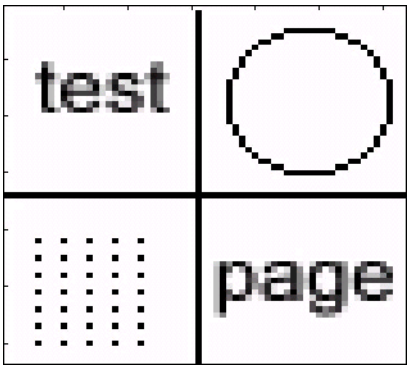


Fig. 3.8(a) Test image

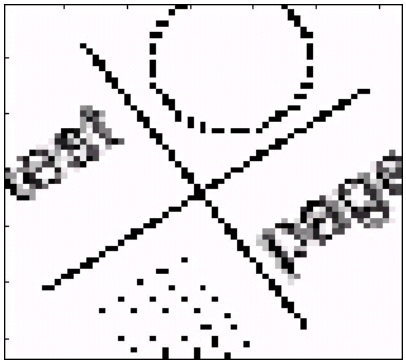
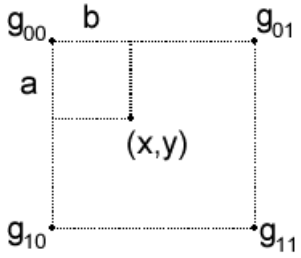


Fig. 3.8(b) Simple rotation.

The quality in Fig. 3.8(b) is very poor. That is, the rotated words have become blurred and hard to read, the perimeter of the rotated circle contains gaps and the rotated grid of points is not uniform any more. In sum, the simple 36 degree rotation causes loss of important information. In other words, after rotation we get the wire that is somehow different from the one we started with. One way to solve the problem of image distortion resulting from rotation is linear interpolation. The basic linear interpolation idea is presented in Fig. 3.9. That is, the colour of each pixel point is approximated from four neighbouring pixels. In addition, the resolution of image is increased to yield more differently shaded pixels. This increases computational time, but the final effect is rewarding. For example, Fig. 3.10(a) contains an image rotation with linear interpolation using the same information found in Fig. 3.8(b). The rotation in Fig. 3.10(a) preserves dependency between pixels and does not result in the loss of any valuable data.



$$f(x,y) = g_{00} + (g_{10} - g_{00})a + (g_{01} - g_{00})b + (g_{00} + g_{11} - g_{10} - g_{01})ab$$

Fig. 3.9 Linear interpolation.

Let (x, y) denote plane coordinates of considered rotated pixel, and let a, b denote distances to the pixel from original image, which coordinates are given by $(\lfloor x \rfloor, \lfloor y \rfloor)$. Let g_{00}, g_{01}, g_{10}, g_{11} denote values of the closest four pixels from original image to a given rotated pixel (x,y). In Fig. 3.9, f(x, y) computes the value of pixel (x, y) as a linear combination of four given pixel values g_{00}, g_{01}, g_{10}, g_{11} weighted by their corresponding distances to point (x, y).

The last step of rotation is superimposing a δ-mesh on top of the image. This smoothes the image and removes the rest of disturbances introduced by rotation. The final result is shown in Fig. 3.10(b). Although the image in Fig. 3.10(b) looks blurry, this

has no negative influence. The most important thing to notice in Fig. 3.10(b) is that there is less "noise" and the information in Fig. 3.10(a) has been preserved. As a result a smooth transition contains for us the same information as a sharp edge, but contains less noise. The δ-mesh makes it possible to obtain more accurate image measurements. For this reason, the δ-mesh has been found to be useful in the measurement of ice accumulation on a transmission line.

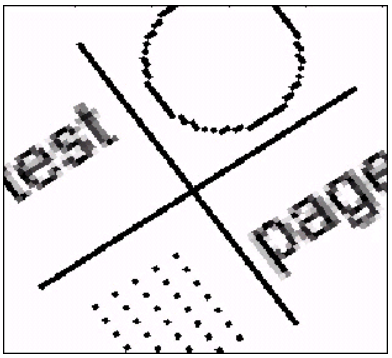


Fig. 3.10(a) Linear interpolated image

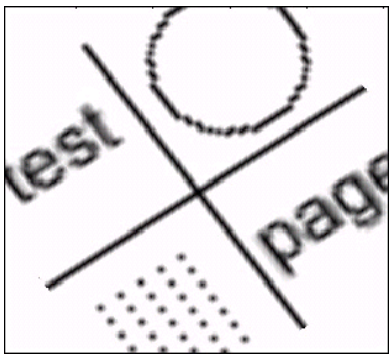


Fig. 3.10(b) Image inside δ-mesh.

3.4. Angle Detection

Angle detection is done in two stages. An overview of the angle detection method used in this thesis is given in Fig. 3.11.

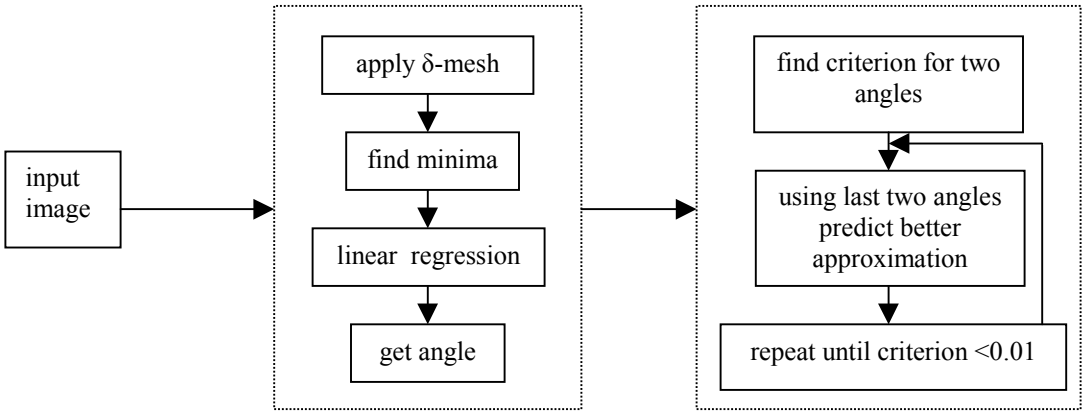


Fig. 3.11 Flow Chart for Wire Angle Detection Algorithm.

First, a rough approximation is calculated and then during an iterative process adjustments are made to yield an acceptable value. It is assumed that a wire can be situated at all angles except vertical. Angles in the range -45 to 45 degrees (with respect to horizontal direction) are preferable. Since positive and negative angles result in the same image distortion, after detecting the angle its absolute value is used for further calculations.

The first step in discovering the correct angle of rotation is to obtain a rough approximation of the angle using a δ-mesh. We need to find a rotation angle with tolerance of few degrees (relative to the horizontal). The application of a δ-mesh with 32 × 32 cells results in a very precise estimate of the desired rotation angle. This δ-mesh size is set regardless of an original image size. On the one hand, it has been found that 32 × 32 cells is enough to obtain the required accuracy and, on the other hand, each cell in a 32 × 32 mesh contains on average more than one pixel. This results in some noise attenuation.

The next step (from Fig. 3.11) is to find minimum values of pixel brightness in each column. From assumption that in the picture we have only wire and the sky and by selecting pixels of smallest values of pixel brightness we are sure, that we identify pixels belonging to the wire. At this point it is not important that each time we can choose a different point of wire cross-section. In a digital image of transmission line, the wire is very thin compared to its length, hence assuming it is actually one dimensional does not introduce extra error. Chosen points, form a piecewise linear function *y=f(x)* on an image. In order to find main direction determined by them, a linear last square fit (or linear regression) is applied to approximate this function. Linear regression formula is given by 3.1.

$$lr(x) = y_{mean} + S_{xy} (x - x_{mean}) / S_{xx} \qquad (3.1)$$

where x_{mean} is the mean of the x-values, y_{mean} is the mean of the y-values, S_{xx} is the variance of x-values, and S_{xy} is the covariance between x and y values, while x and y denote horizontal and vertical image coordinates respectively.

The angle between horizon and linear regression is obtained by computing the inverse tangent of S_{xy} / S_{xx}. In Fig. 3.12 we can see a sample plot showing a function f made of minimum values of pixel brightness and its linear regression.

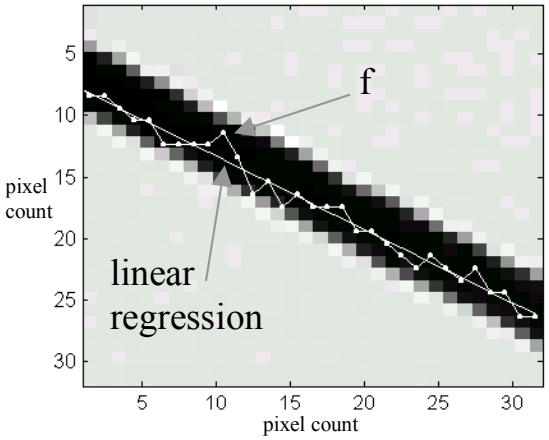


Fig. 3.12 Example of Measuring Wire Angle

In the second step, we determine exact angle of rotation of a wire image such that the wire is parallel with the horizontal axis of a δ-mesh. To do so, we use more complex methods to determine the position of a wire. Because of the previous step, namely, finding rough approximate of the angle, we can use the fact that a wire is approximately levelled. What follows, is that vertical direction of an image corresponds to cross-section of the wire and the horizon of the image corresponds to the direction along the wire. The information (i.e., pixel's brightness determining the degree that they belong to a wire) which is important for us is oriented along vertical axis.

As mentioned before, vertical cross-sections are very similar to each other. Variations between them are caused by the noise and the angle between the wire and horizon. We try to use only one source. Noise carries no information and is removed by averaging inside a δ-mesh cell. In order to preserve information in the vertical direction and attenuate noise in the horizontal direction, mesh cells are designed with a rectangular (instead of squared) shape with the horizontal side being longer than the vertical side of a mesh cell. By looking at the cross-sections of an image, we can see that the shift of the shape of the wire depends on the angle (see Fig. 3.13).

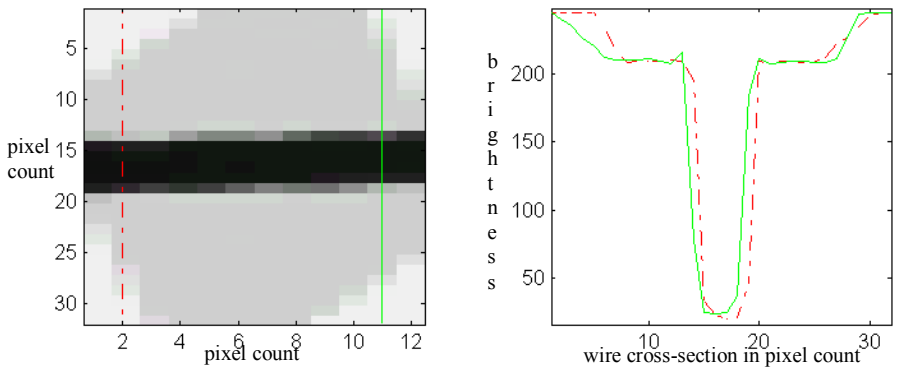


Fig. 3.13 Rotated Image After First Step And Two Shifted Cross-Sections.

The greater the angle the bigger the shift and, in particular, if the wire is totally levelled all cross-sections should overlap. This observation is very important and we use it for our calculations.

Before comparing cross-sections, we have to select points that belong to the wire or its close neighbourhood. From the right side of Fig. 3.13, we see that the extreme left and right parts of the plot are tending to the value 255. This is caused by white spots introduced by rotation. White spots due to rotation yield no information and can only lead to faulty calculations. We select important points (ones with above average rough inclusion values) by removing pixels belonging to white spots. With the remaining points, we repeat the same algorithm again. This "purging" procedure makes it possible to remove all outsiders, leaving only points from the wire or points (pixels) in the wire-sky neighbourhood. After that, the difference between cross-sections lying at the same distance from the centre is computed. The two extreme cross-sections in Fig. 3.13 are distorted by the rotation process, and are not taken into account. Thus, from 12 cross-sections, after taking away two and performing subtraction, we obtain only five cross-sections to consider. Then a mean value is computed that results in only one averaged cross-section. A plot illustrating this idea is shown in Fig. 3.14. Next, a middle point is found, and all non-zero points on each side of it are separately summed. The difference between these two sums provides a criterion for deciding on the angle of a wire image.

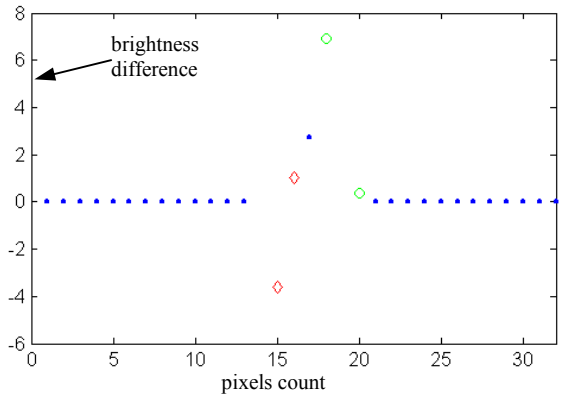


Fig. 3.14 Averaged Cross-section.

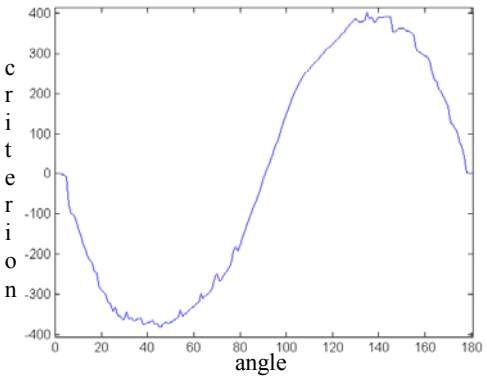


Fig. 3.15 Angle Criterion Plot.

The proposed algorithm requires many steps, but the resulting wire-angle decision criterion is a quite precise determinant of the exact wire angle. In Fig. 3.15, the plot shows decision criterion values with respect to the wire angle. The maximum and minimum values in Fig. 3.15 are 90 degrees apart. Yet the most interesting property is its behaviour close to zero. In the range ±10 degrees from an angle for which wire is levelled, decision criterion behaves like a linear function. This makes process of finding zero much easier. We do not need to search blindly using only the sign of the criterion function. Instead, we can predict the roots of the criterion function knowing its two values. If we assume that the specified region represents a linear function, then from two points in this subplot we can calculate the slope and find out for which angle the plot crosses zero. And of course, we can assume this because of the first step, namely, step where we used minimum values of pixel's values and linear regression to estimate the wire angle, which assures us that present angle is in the required range.

This approach in preparing a wire image for measurement turns out to be very efficient. That is, it has been found with experiments with hundreds of wire images that only four to five iterations are required to find angle for which the criterion value is smaller than 0.01. The resulting angle is exact with a precision of ±0.06° (this has been checked for seven different parts of the same image). This means that we are able to measure this angle with a precision of less then one pixel!

The importance of the δ-mesh can be observed in the behaviour of the criterion function. If we did not use it, this function becomes more random and its linear part

becomes twice as small. Consider Fig. 3.16, decision criterion crosses zero for angle close to 20 degrees, however decision criterion's values can be approximated by linear function only in a range 15 to 25 degrees. In plot segments for angles close to ±90 degrees from point where decision criterion crosses zero, namely, -10 and 50 degrees, decision criterion function is very unpredictable. This causes longer time of convergence and sometimes even divergence, when the first two guesses are outside linear part. Corresponding "bumps" in the Fig. 3.15 have different characteristic. When δ-mesh is used A plot of decision criterion has "bumps" located at approximately equal distances. Most probably this is result of some properties of the image, not just noise. For this reason δ-mesh proves to be effective tool in combating noise without destroying underlying process.

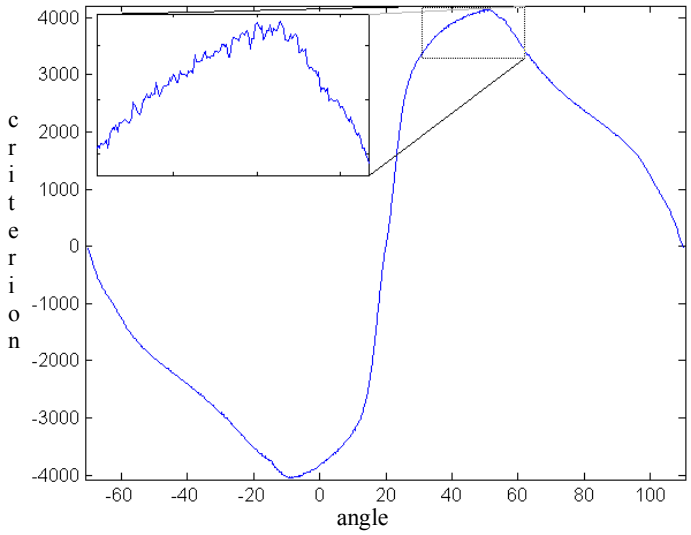


Fig. 3.16 Angle Criterion Plot Without δ-Mesh.

3.5. Observed Wire Thickness

In this section, we assume that the image has been rotated and contains an exactly leveled wire. Next task is to determine the exact edges of a wire. As in the previous case, we are interested in a much greater accuracy than one pixel. Therefore, we need to use information contained in pixel values, not only in their location. Since wire edges are

parallel to the horizontal sides of an image, we can stop considering the image as a two dimensional data set. The data can be averaged columnwise. As result, we get a function describing the average light intensity in each row of pixels in the image. Plot of this function is shown in Fig. 3.17. For the sake of simplicity the plot on the right side corresponds to image on the left, but after removing rows influenced by white pixels introduced by rotation (see the right side of Fig. 3.13 for comparison). This shows how this function differs from an ideal membership function we would like to see. All pixels belonging to a wire should have approximately the same brightness. But in the plot we observe phenomenon, which was mentioned earlier. That is, the wire is lit up by the sun and part of it is lighter than the rest. Notice, also, that there is small part of the image covered with shade. On the plot, the corresponding pixels have small brightness values (approximately 80). Notice how this influences border parts. We can see that there are two kinds of transition. The right wire-to-sky transition in Fig. 3.17 is as expected. That is, dark colours become gradually lighter and lighter until they have some level characteristic of the sky. Obviously there is some point where colour is bright enough to denote sky. But the left wire-to-sky transition in Fig. 3.17 is different. That is, the wire close to the edge is much darker resulting in higher contrast. And there is an interesting peak right in the middle of the transition. We did not expect this to occur, since there is no reason to see pixels brighter than the sky. All we see on this picture is wire and the sky, but some of the pixels obviously do not belong to either wire or sky. So what caused this phenomenon?

The answer to this question is digital camera itself as the source of the unexpected pixels. In order to produce sharper pictures camera uses the same trick as the human eye. That is, whenever there is a quick and flat transition between colours of different brightness, the camera adds some brighter (or darker) pixels to make this transition more visible. Since the human eye does it, pictures which already have it look nicer to us. Therefore, this algorithm was implemented in digital cameras and its result is seen on the left transition.

To sum up, in order to detect wire edges we have to be familiar with some characteristic features of the wire cross-section such as uneven wire brightness or two different kinds of wire-to-sky transition.

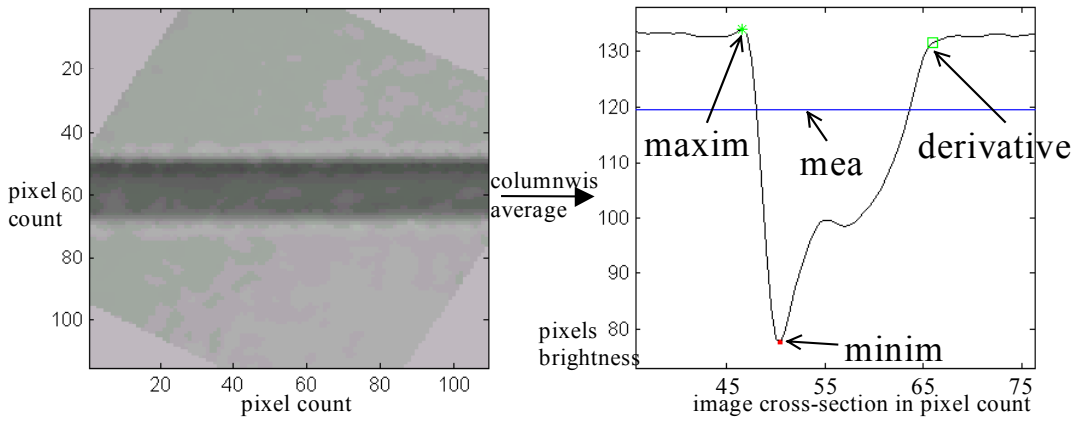


Fig. 3.17 Rotated Wire Image And Its Columnwise Average.

In what follows, we repeatedly refer to the image and plot Fig. 3.17, because it has been found after many experiments that this figure typifies what has been found in arriving at a method of measure wire diameters in hundreds of pictures of transmission lines.

Next, we start looking for edges by localising the middle of a wire. The simplest way to do this is to find the darkest point or the point with the smallest pixel brightness value from the given image cross-section. Now, assume we would like to detect the left edge first. We find the first local maximum on the left side of the minimum. This does not have to be the maximum made by camera, but just some distortion in the sky. Thus, we check if this maximum is the one we are looking for. If it is, it will be characterised with a large to moderate rate of change in brightness around the point representing a maximum. If this is the case as in our example, we consider this point to be the exact left edge. Now, we turn to the right side of the wire image and repeat the process again. But this time the situation is more complicated. The first maximum we find is located within the wire (e.g., around 40 on x axis on the right side of Fig. 3.17). But this is not the transition we are looking for because this is the middle of a wire, not the wire's edge. Therefore, we perform a check if the first maximum value is above average. If it is not, we discard this point and start looking for another one. The next point we find on the right side of Fig. 3.17 corresponds to an x value of around 78. We check the new maximum in the example on the right side of Fig. 3.17 and discover a difference between maximum and its neighbouring points is less than or equal to 1. This means that the slope

is too flat to be part of the wire-to-sky transition. It is again not the point we need. Even so, the information we gained during this search is not useless. That is, we have located a closed interval, where the function is monotonic and one end belongs to the wire and the second to the sky. A transition between wire and sky must be somewhere in this interval. Deciding which point exactly corresponds to the edge of a wire image is a very difficult problem. For sure, we cannot use some fixed value, since it is dependent on the average picture brightness. A better approach is to look only at the relative change in brightness. That is why in the next step a vector of differences between adjacent points from a given interval is calculated (this corresponds to finding the derivative for the continuous case). This vector of differences is constructed from the end belonging to the sky till the point where a value bigger than some threshold is found. This threshold indicates that the brightness is changing fast enough so we already reached the wire-to-sky transition part. This point is denoted on the right side of Fig. 3.17 by a square.

Using this algorithm we can detect two points denoting wire edges and calculate Observed Wire Thickness by simply finding the distance between them. At the beginning of this section, it was mentioned that the aim was to estimate a wire diameter with accuracy better than one pixel. The algorithm presented in this section allows us to measure thickness with a precision up to one pixel only. In order to detect Observed Wire Thickness we have to rotate the wire. But rotation is performed simultaneously with oversampling and linear interpolation. The resulting rotated image can have a few times better resolution. Thus the final resolution with respect to the original image is better than one pixel, as we expected.

The second comment addresses a problem of how close we are to the real wire edge. This algorithm tells how to find it, but does not justify why it is correct. In one case we trust in camera operations, in second we use some pre-selected threshold. Fortunately, we do not have to exactly know the real wire edge to get a good estimate of wire thickness (i.e., diameter). The more important issue is that we are always at approximately the same distance from the wire when pictures are taken (e.g., pictures taken from a vehicle parked in the same place or by a person or robot stopped in the same place each time pictures are taken). That is why the estimated wire thickness is called Observed Wire Thickness. It is assumed that what is seen in an image does not have to

correspond exactly to the actual wire. But it is possible to convert the observed thickness into actual wire size with low error. In other words we allow only for a linear dependency between observed and actual wire sizes and this assumption satisfies the proposed approximation method (i.e., "measuring roughness").

Bibliography

[1] Z. Pawlak, J.F. Peters, A. Skowron, Z. Suraj, S. Ramanna, M. Borkowski, Rough measures: Theory and Applications. In: S. Hirano, M. Inuiguchi, S. Tsumoto (Eds.), Rough Set Theory and Granular Computing, Bulletin of the International Rough Set Society, vol. 5, no. 1 / 2, 2001, 177-184.

[2] A. Skowron, J. Stepaniuk, J.F. Peters, Hierarchy of information granules. In: L. Czala (Ed.), *Proc. of the Workshop on Concurrency, Specification and Programming*, Oct. 2001, Warsaw, Poland, 254-268.

[3] J.F. Peters, S. Ramanna, M. Borkowski, A. Skowron, Z. Suraj, Sensor, filter and fusion models with rough Petri nets, *Fundamenta Informaticae*, vol. 34, 2001, 1-19.

[4] Z. Pawlak, A. Skowron, Rough membership functions. In: R. Yager, M. Fedrizzi, J. Kacprzyk (Eds.), Advances in the Dempster-Shafer Theory of Evidence, NY, John Wiley & Sons, 1994, 251-271.

[5] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data. Boston, MA, Kluwer Academic Publishers, 1991.

[6] J.F. Peters, S. Ramanna, Z. Suraj, M. Borkowski, Rough neurons: Petri net models and Applications. In: L. Polkowski, A. Skowron (Eds.), *Rough-Neuro Computing*. Berlin: Springer, 2002, 472-491.

[7] J.F. Peters, A. Skowron, Z. Suraj, M. Borkowski, W.Rzasa, Measures of Inclusion and Closeness of Information Granules: A Rough Set Approach. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 304-311.

[8] J.F. Peters, A. Skowron, Z. Suraj, W. Rzasa, M. Borkowski, Clustering: A Rough Set Approach to Constructing Information Granules. In: Z. Suraj (Ed.), Soft Computing and Distributed Processing, 6th Int. Conf., Rzeszow, Poland, 24-25 June 2002, 57-62.

[9] J.J. Alpigini Measures of closeness of performance map information granules: A rough set approach. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 293-299.

[10] M. Borkowski, Signal analysis using rough integrals. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 218-225.

[11] J.F. Peters, T.C. Ahn, M. Borkowski, Obstacle Classification by a Line-Crawling Robot: A Rough Neurocomputing Approach. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 606-613.

[12] J.F. Peters, T.C. Ahn, M. Borkowski, V. Degtyaryov, S. Ramanna, Line-crawling robot navigation: A rough neurocomputing approach. In: D. Maravall, D. Zhou (Eds.), Fusion of Soft Computing and Hard Computing Techniques for Autonomous Robotic Systems. Studies in Fuzziness and Soft Computing, J. Kacprzyk (Ed.). Berlin: Physica-Verlag, 2002 [to appear].

[13] S.K. Pal, L. Polkowski, A. Skowron (Eds.), Rough-Neuro Computing: Techniques for Computing with Words. Berlin: Springer-Verlag, 2002.

[14] M. Borkowski, J.F. Peters, Approximating sensor signals: A rough set approach. In W. Kinsner, A. Sebak, K. Ferens (Eds.), Proc. of the IEEE Canadian Conf. on Electrical and Computer Engineering 2002 (CCECE'02), 12-15 May 2002, Winnipeg, Manitoba, Canada, 66-72.

[15] S. Ramanna, Rough neural network for software change prediction. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 614-621.

4. System Calibration

We have described a method to measure Observed Wire Thickness. Thus, we are ready to compute the Scale Factor *SF* from Eqn. 2.4, namely,

$$OWT = \text{SF} * \textit{wire thickness} = f(\textit{aperture}, \textit{location}, \textit{magnitude}) * \textit{wire thickness}$$

Recall that the Scale Factor depends on three variables, namely *aperture, location* and *magnitude*. Here, we show how to combine this knowledge to predict the correct Scale Factor for an unseen wire image.

Location is nothing else than the angle between the wire and horizon. We know how to find it using the method described in the previous chapter (i.e., see section 3.4). The *magnitude* parameter in Eqn. 2.3 controls the size of the Observed Wire Thickness, which is also known. If we take a calibrating set of wire images containing pictures with wires of known diameters, we get many tuples (*location*, *magnitude, Scale Factor*). For more details on how to obtain these tuples see chapter 5. This information can be stored in a lookup table. However, we cannot take all possible pictures from all possible angles and distances. Therefore, we have to use a lookup table only as a source of guiding points. We try to approximate Scale Factor values between known points using some interpolation algorithms.

In a Fig. 4.1 a block diagram of the proposed measurement system is shown. This measurement system consists of two separate subsystems. They are represented by two groups of vertically connected blocks. The subsystem shown on the left side of Fig. 4.1 is responsible for calibration. The subsystem shown on the right represents a user's end module responsible for performing measurements of ice thickness based on information retrieved during the calibration process.

First consider the calibration module. Input data consists of a calibrating set of digital images representing transmission lines. It is assumed that input images were pre-processed prior to calibration. Therefore, they contain only a wire and the sky in the background. The first step is the calculation of the exact angle between the wire and the horizontal axis of the image. This is block (1) in Fig. 4.1. Implementation of this stage is

based on the use of a δ-mesh, linear regression and a decision criterion devised specially for solving the problem of determining the relative wire angle. The exact angle is crucial to the next step (2), where **O**bserved **W**ire **T**hickness is found. The detecting wire thickness algorithm is based on δ-mesh, minimum-maximum pixel's brightness comparisons and derivatives. The third box (i.e., box (3) in Fig. 4.1), denotes important logically step. A knowledge from an image, namely, Observed Wire Thickness measured in pixels, and background knowledge, namely, known real wire size in millimetres, is combined. This results in a tuple of two independent variables <*OWT*, *angle*> and one dependent Scale Factor *SF*. The fourth box in Fig. 4.1 represents the output table, where each row consists of three just-mentioned numbers.

User module's goal is to measure ice thickness accumulated on power transmission lines using information from the image a of wire with known diameter, partially covered with ice. The first two steps of the right-hand sequence of blocks in Fig. 4.1 are identical to those used in the calibration sequence in Fig. 4.1: the exact angle of a wire with respect to horizon is calculated and the Observed Wire Thickness if measured. Box (5) represents approximation of Scale Factor value for previously calculated *OWT* and *angle*. Boxes (6), (7) and (8) denote three different algorithms which may be used to find the best approximation of Scale Factor. These three algorithms are Nearest Neighbourhood, Gouraud shading and the improved Nearest Triangle Gouraud shading, respectively. In the last step (9), measured Observed Ice Thickness combined with the calculated Scale Factor gives actual ice thickness.

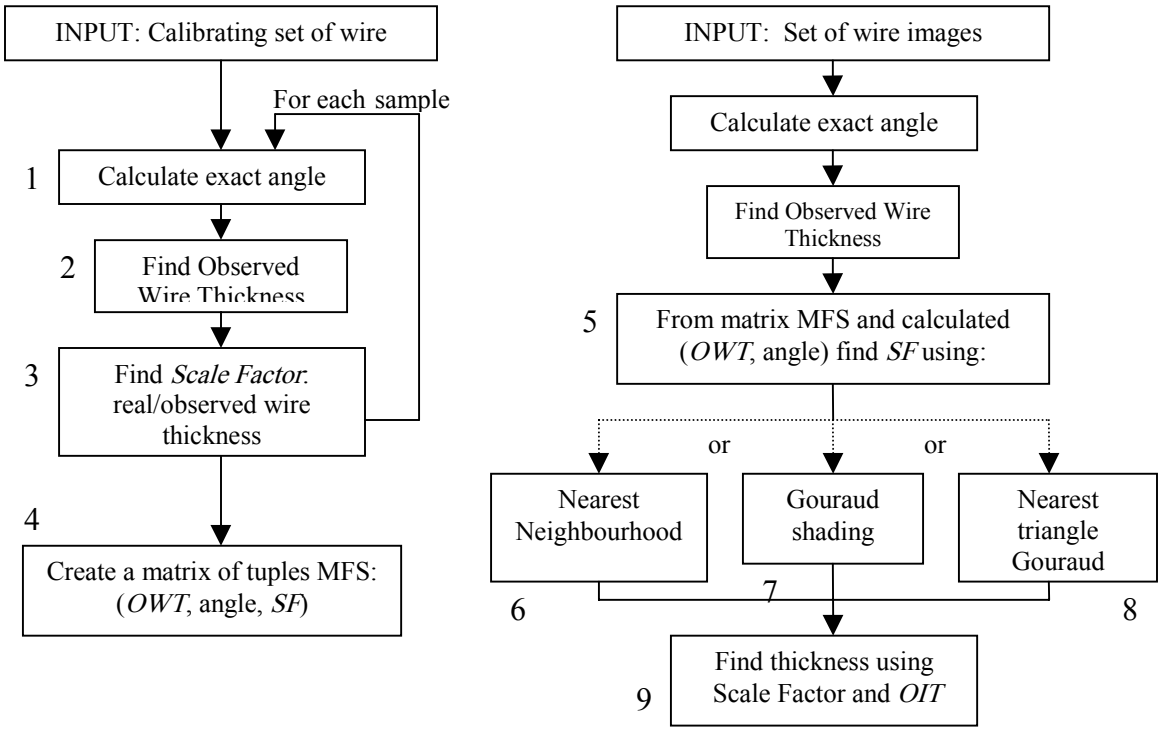


Fig. 4.1 Measurement System Block Diagram.

The main approach to approximating *SF* values for unknown points is to interpolate them from nearest known points. *SF* is a three dimensional function *SF* = *f*(*OWT*, *angle*). This raises a question: how shall we treat *OWT* and angle values if their mean values are different? If this is a case and, for example angle values are on average five times bigger than *OWT*, then by searching for the closest points we take into consideration only points with the same angle, but different *OWT*. This means that the angle factor is five times less important than *OWT*. Moreover, we do not have this information a priori.

To obtain deeper insight into this problem let us see how the proposed method is correlated to the physical process of taking a picture. We should concentrate on dependency between *OWT* and wire angle, and sharpness of wire edges. It is true that the sharper the image the smaller the transition on a wire-to-sky edge. On the other hand, the wire angle has nothing to do with image sharpness. The edge can be shifted, but the angle is still the same. This shows that there is some correlation between aperture settings and

OWT. Depending on the aperture, importance of *OWT* can be different, i.e., *OWT* is more trustworthy for sharper pictures and less for blurry pictures with the constant importance of wire angle. Therefore, we need some scaling parameter to adjust magnitudes of *OWT* and the wire angle. This parameter is called aperture and is the last one we need for Eqn. 2.3. Our final formula for Scale Factor is (4.1).

$$SF = f(aperture * OWT, angle) \quad (4.1)$$

where *aperture* is any real number. In the next subsection, an algorithm to find an appropriate aperture value is given.

4.1. Nearest Neighbourhood

A very simple method for approximating the closest value for a new point is given in this section.

Nearest neighbourhood method

Input: set of all points *TR*, coordinates of given point *p*;

Output: nearest point to *p*;

Algorithm 4.1:

- for each point in table TR find its distance to point *p*;
- select point from TR with the smallest distance to *p*;

A table *TR* contains all calibrating points, namely, all known for the system tuples: < *OWT*, *angle, SF* >. Yet in this algorithm only the first two variables are used. Point p, is a two dimensional structure < *OWT*, *angle* > and represents information about new wire for which we want to find Scale Factor. The word "distance" means standard Euclidean distance in two dimensions using the first two variables, that is *OWT* and *angle* (for more information about metrics, see Appendix E.I). A C++ implementation of Algorithm 4.1 is presented in Appendix C.

Assume we already have a matrix of Scale Factors (call it *SF* matrix). Given a new Observed Wire Thickness and angle of rotation to orient a wire image relative to the horizontal axis of an image, we find a point from the *SF* matrix such that the point found has the smallest distance to the new point. After finding this point, we copy the value of the Scale Factor of the data point found. This algorithm is fast and easy to implement.

In Fig. 4.2 a Scale Factor map is plotted for the nearest neighbourhood method. The dark colour denotes small *SF* values and white denotes big *SF* values. There is a noticeable dependency between Observed Wire Thickness and Scale Factor, since a pixels' brightness seems to gradually increase when moving to the right along horizontal axis.

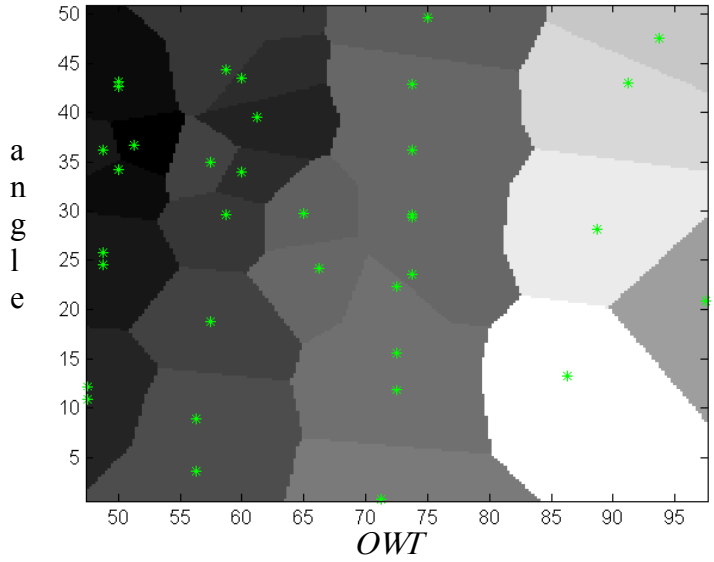


Fig. 4.2 Scale Factor Map for Nearest Neighbourhood Method.

4.2. Gouraud Shading

The nearest neighbourhood method seems to work well if we have many calibrating points. In that case, the error made by approximating the current point by the closest value is small. Nevertheless, if the closest point is far away, the Scale Factor value

may be quite different. The Gouraud shading method [2] was implemented to avoid this situation.

Briefly, Gouraud interpolation defines a way to linearly spread vertex values along a whole triangle. As input, we have three points on a plane and their values (in our case this would be Scale Factor values, though in most common application is light intensity, there are even parallel implementations of this algorithm [3]). From simple algebra, interpolated values can be calculated for all points inside a triangle. Let Ap denote the line connecting one of triangle vertices and point p. Point A' is a point where two lines Ap and BC cross. Let A, B, C represent the vertices of a triangle. See Appendix C for a C++ implementation of this algorithm.

Gouraud interpolation:

Input: three points with known values A,B,C, and a point *p*.

Output: value of point *p* as result of linear interpolation from points A,B,C.

Algorithm 4.2:

- find line A*p* and locate point A';
- from linear interpolation from points B and C, find value for point A';
- from linear interpolation from points A and A', find value for point *p*.

The shading algorithm itself is not very complicated nor computationally intensive. But in order to use this algorithm we have to create a net of triangles. Then, while finding approximating value to any point, we have to locate a triangle enclosing this point. The following is the algorithm and its detailed description.

Partitioning into triangles:

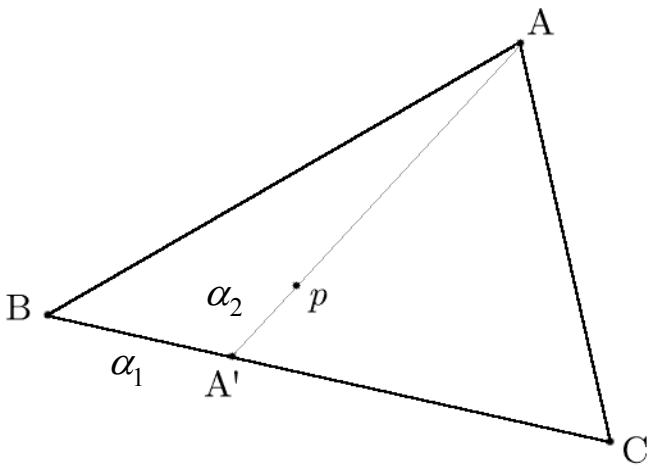
Input: Set of all calibration points TR.

Output: value of point *p* as result of linear interpolation from points A,B,C.

Algorithm 4.3:

- create a list of all possible lines LLTR between points from TR;
- sort LLTR in ascending order with respect to their length;

- initialise list of non-crossing lines LNCL with the first line from LLTR;
- remove first line from LLTR;
- for each line L in LLTR
    - if L does not cross with all lines from LNCL
        - add L to LNCL;



$$A' = \alpha_1 C + (1-\alpha_1) B$$
$$p = \alpha_2 A + (1-\alpha_2) A'$$
$$\alpha_1 = \frac{|BA'|}{|BC|}, \alpha_2 = \frac{|A'\,p|}{|A'\,A|}$$

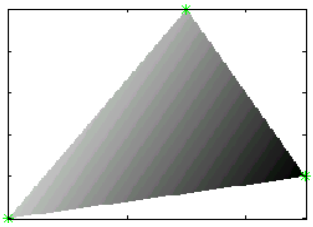


Fig. 4.3. Gouraud Shading.

First a list of all possible lines between calibrating samples is created. Then the lines are sorted in ascending order with respect to their length. This step assures us that the created triangles have possibly short sides. There is no unique partition into triangles for any set of points in a plane. From all possible partitionings, the one that contains small triangles was chosen. Now, only lines which do not cross with any other lines are selected. First the smallest line is chosen – it is for the time being the only element of a list of triangles' sides. Then each line from a list of all lines is checked if it crosses any line from list of triangles' sides. If not, it is moved to the list of triangles' sides. At the end, the triangles' sides list is a partitioning of the whole space of pairs (*OWT*, *angle*) on triangles (see figure Fig. 4.4). This list is created only once. It is part of the system and does not need to be recalculated for every new testing point. However, this list must be created again if a new point is added to the calibrating list.

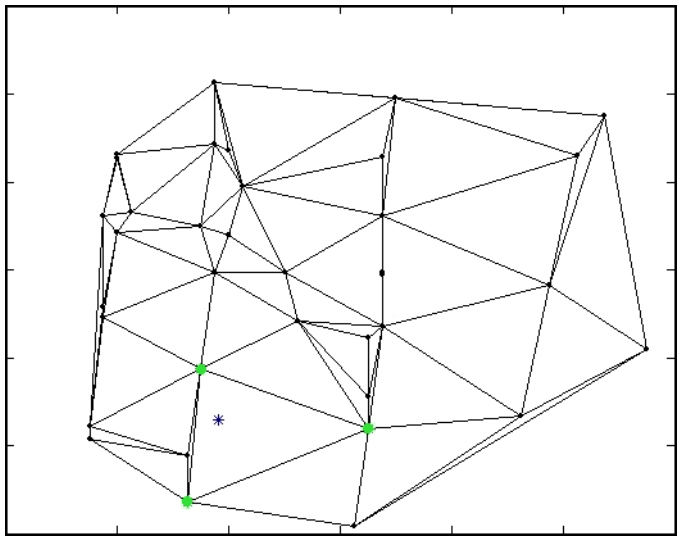


Fig. 4.4 Sample triangle net for Guraud shading algorithm

The second step in the Gouraud method must be applied for every test point. A triangle, made from points from table of all calibrating points, containing the test point must be found. In order to do this, a straightforward property of vertices of a triangle is used, namely, vertices are the only points, which connected with a given point do not don't cross any line from the list of triangles' sides. Three points having this property are chosen for Gouraud shading.

In Fig. 4.5, a map of Scale Factor values is plotted. Again, the dark colour denotes small SF values and white big SF values. Notice that the transitions between points are smooth compared to nearest neighbourhood method. The only exceptions, where transition is not continuous, can be found on edges and outside the convex hull of all calibration points. These discontinuities are caused by limitations of Gouraud shading algorithm. Using Gouraud shading is impossible to approximate Scale Factor values for point, which is located outside any triangle made from any calibrating points. However, it is possible to find some other techniques to extend information from three vertices of triangle to whole two dimensional plane. Nevertheless, in author's opinion, profits gained from implementing these additional approximation techniques are not worth with comparison to increased complexity. Therefore, for points outside any triangle made from calibration points, which is equivalent to the fact of being outside convex hull of all

calibration points, nearest neighbourhood method is used. For more information about convex hulls see Appendix E.II.

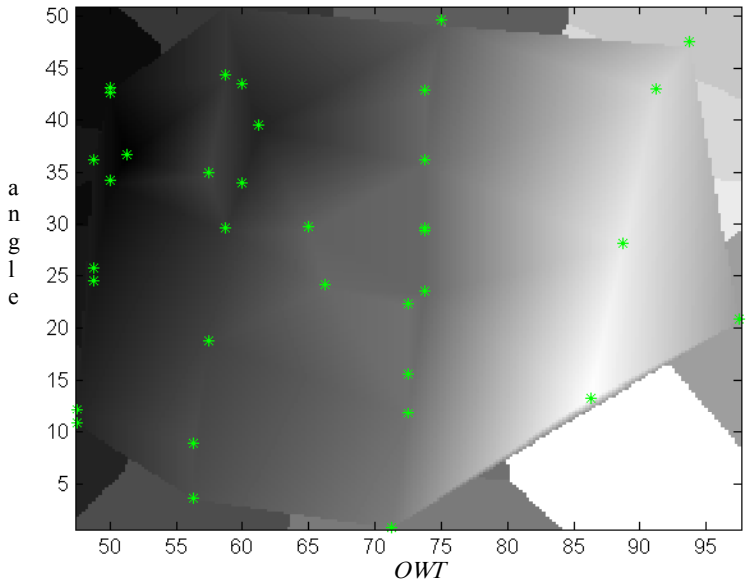


Fig. 4.5 *SF* Map for Gouraud Shading

4.3. Nearest Triangle Gouraud Shading

The Gouraud shading method of selecting triangles has one disadvantage in the study of wire images with or without ice covering. Consider a point denoted by a star in Fig. 4.6(a).

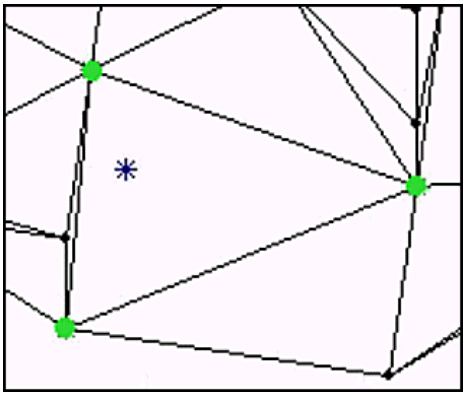


Fig. 4.6(a) Triangle for Gouraud Shading Alg.

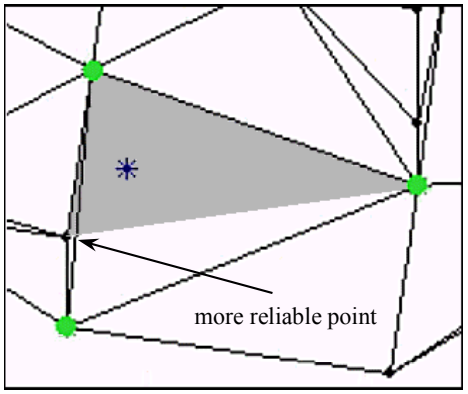


Fig. 4.6(b) Nearest Triangle (shaded)

A triangle used for Gouraud shading is denoted by big green dots. From the figure it is apparent that the chosen triangle is not the best one. A little bit lower and on the left from a star is another point, not used as a calibrating point (see Fig. 4.6(b)). The information coming from this point is more reliable for the star, since it is located much closer than lowest vertex being used. This algorithm uses the same Gouraud shading, but utilises a different algorithm for finding the surrounding triangle. That is, for every point, the best triangle is found. What makes this task difficult, is that given a testing point, it is not enough to find the three closest calibrating points. The selected calibrating points can form a triangle, which does not contain the given point.

Find closest triangle:

Input: Set of all calibration points TR, point p;

Output: the closest triangle n_1, n_2, n_3, which encloses point p;

Algorithm 4.4:

- make a list of distances DIST between all points in TR and point p;
- sort TR in ascending order of distances from DIST;
- select n_1 first points, which enclose point p;
- if $n_1 <> 3$

    from n_1-1 first points from TR, find n_2 first points, such that these n_2 points and point n_1 enclose point p;

    - if $n_2 <> 2$

        from n_2-1 first points from TR, find first point n_3, such that points n_1, n_2 and n_3 enclose point p;

Algorithm 4.4 starts with creating a list of all calibrating points ordered in ascending distances from given test point. A set of n_1 first points, which create a closed figure that contains given test point is found. Since n_1 is the smallest number of points having this property, we know that n_1-th point must be included in final triangle (this is impossible to make smaller triangle, which surrounds this point, and any triangle having this property and not build from n_1-th point will have at least one vertex located further than n_1). If n_1 is equal to three, we already have the best triangle. If not, we have to find

only two points from n_1-1 first points of the previously found figure. To do so, we repeat the same algorithm, but a figure which must surround the test point consists of points having indices 1 to n_2 and point n_1, where $n_2 < n_1$. Again, the point with index n_2 must be included. By repeating the same procedure once more, we find the last vertex and can perform Gouraud shading using these three points. As in the previous Gouraud method (algorithm 4.3), if a test point is located outside the convex hull of all calibrating points, the nearest neighbourhood method is used.

Figure 4.7 shows a Scale Factor map for algorithm 4.4. The dark colour in Fig. 4.4 represents small Scale Factor values and white big *SF* values. Notice, transitions are not smooth as in previous case. This is caused by sudden change of vertices used for the triangles.

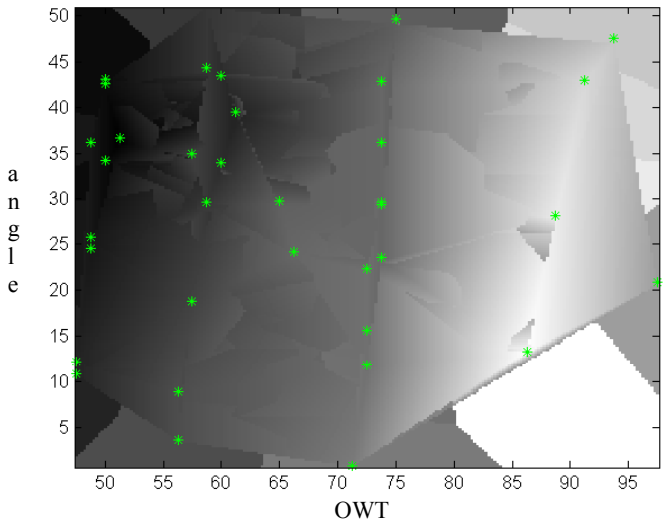


Fig. 4.7 Scale Factor map for nearest triangle Gouraud shading.

4.4. Ice Measurement

After the detailed description of all measurement system methods just given, we now return to the original problem and attempt to measure ice accumulation (thickness) on a transmission line. Consider a sample image from Fig. 4.8. The image in Fig. 4.8 shows a wire covered with thick ice on one end (see chapter 5 for detailed descriptions of

experiments with measurements of ice on a wire). Knowing only wire thickness we try to measure the thickness of ice accumulation. To do this, we use a database of 279 pictures made of the same wire, but from different distances and different angles. Assume, the system is calibrated, which means that a Scale Factor map is ready to use.

The first step is to obtain from the image the Observed Wire Thickness. For this purpose we use only small part of the image containing only a bare wire. We level it and measure the *OWT*. Knowing the angle and *OWT* we approximate the SF value using any of described earlier methods. This makes it possible to convert pixels into real measure units. The last part of the measurement process is to measure ice thickness in pixels and convert pixels into millimetres. When measuring ice thickness precision is not as crucial as when finding *OWT*. In the experiments described in this thesis, wire thickness is roughly three times smaller than ice thickness. This means that any error made when measuring a wire is multiplied by three when measuring ice. Therefore, there was less attention paid to the part of the measurement system responsible for measuring ice. In a more comprehensive ice measure system used by line-maintenance engineers, the ice thickness estimation part of the system being described in this thesis, would take into account a full range of ice thickness possibilities relative to the OWT for a particular transmission line. The intent here is to demonstrate the feasibility of making ice accumulation estimates using wire images.

There are two possible ways to estimate ice thickness on a wire. The first step is rotation of the image of the ice-laden wire to a more or less horizontal position. Then the user has two choices. The user can select measuring points manually using horizontal bars, or the user can use the same algorithm used to find the *OWT* at a point indicated by vertical bar. Using the tool developed as part of the work for this thesis (see section 5 for a description of this tool), a measurement in pixels is automatically converted into millimetres and displayed.

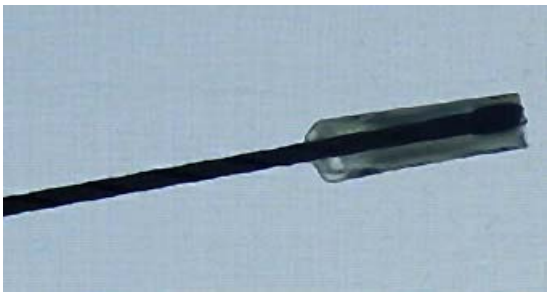


Fig. 4.8 Sample ice on wire image.

Bibliography

[1] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Prentice Hall, Upper Saddle River, New Jersey 07458, 2002.

[2] H. Gouraud "Continuous Shading of Curved Surfaces", IEEE Transactions on Computers, vol. 20, pp. 623-629, June 1971

[3] G. Srikant, L.Wurtz, “A CMOS Parallel Gouraud Shading VLSI Architecture”, IEEE 1992, http://ieeexplore.ieee.org

5. Experiments

Calibration of the transmission line ice measurement (TLIM) system requires a large database containing sample images. In this chapter, we discuss some methods for preparing digital camera images of bare and ice-coated wire, processing the images and validating the information obtained from the images.

5.1. Data preparation

In early stages of the experiments using the TLIM to compute the observed wire thickness (OWT) and observed ice thickness (OIT), a small copper wire (diameter = 1.7 mm) was used (see Fig. 5.1(b)). The small dimensions of a wire made it possible to simulate actual outdoor winter conditions in a common kitchen refrigerator freezer compartment, since the cold chambers in the Chemistry and Microbiology departments were not available. In order to cover the copper wire with ice of required thickness a special procedure was used. A wire was enclosed in a small tube made from aluminum foil. The tube was closed at one end. To make the tube waterproof, aluminum foil was covered with hot wax warmed up with a candle. The process of tightening a tube with a wax is shown in Fig. 5.1(a). After covering the tube with wax, it was filled with water. Then a wire was placed inside the water-filled tin foil tube and then the water-immersed copper wire was put in the freezer until the water was frozen. Cold wax is very fragile after freezing, hence there were no problems with cracking frozen wax covering the ice and removing the covering after removing the ice-clad wire from the freezer. In Fig 5.1(b) is shown a schematically cross-section of a wire inside a tube with ice covered with wax, and the process of covering it with wax and the final result.

Fig. 5.1(a) Preparing Copper Wire.

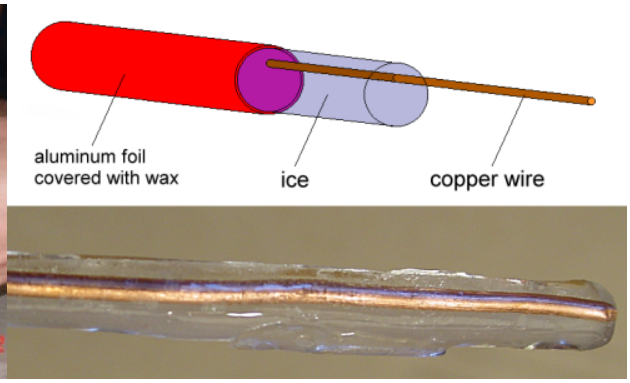


Fig. 5.1(b) Ice-coated Copper Wire

In more advanced stages of experiments, two kinds of power system transmission line wire were used (see Fig. 5.3). The wire samples were provided by the line maintenance department at Manitoba Hydro and are the two most common types of wires used for transmission lines. The thicker wire in Fig. 5.2 had diameter 30.9 millimetres and the thinner one 9.1 millimetres. Samples of the types of wire used are shown in figure 5.2. In order to create ice covering on the smaller diameter wire, the same procedure used as for copper wire in Fig. 5.1 was used again for the Manitoba Hydro wire (see Fig. 5.6). However, the 9.1 mm wire required much more water than the copper wire in Fig. 5.1 to create a thicker ice layer. For this reason, the construction of the aluminum foil tubing used to encase the wire was stronger and required an additional design feature, namely, a wire cage surrounding the transmission line wire to hold the wire in the middle of a foil tube (this wire cage is shown in Fig. 5.6).

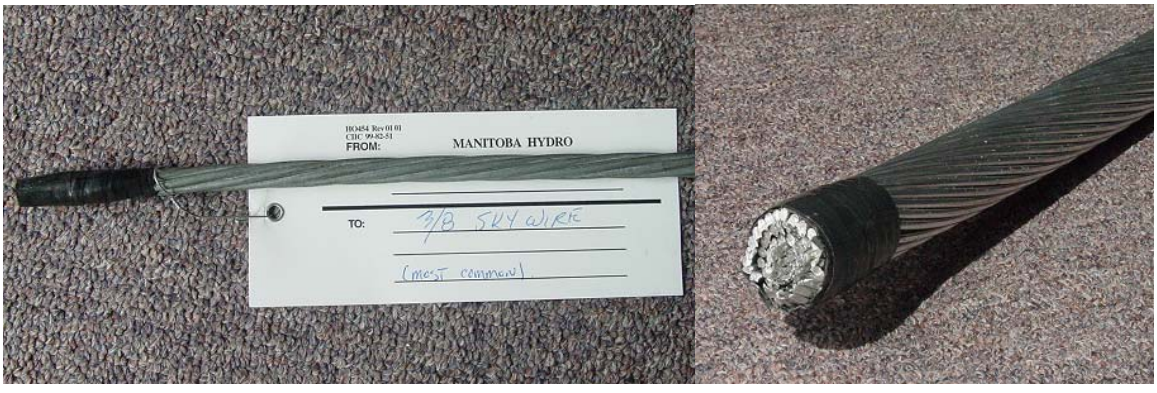


Fig. 5.2 Wires Used in the Project.

Calibration of the transmission line ice-thickness measurement or TLIM system was performed using the thinner, 9.1 mm wire. This means that the system created using wire of this thickness is able to measure ice thickness only relative to 9.1 mm wire. In an industrial version of the TLIM, the system would be calibrated for each of the types of transmission line wire and measurement reference objects such as insulators. For experiments discussed up to this point in this thesis, a wire of known diameter has been used as a frame-of-reference object in computing the Observed Ice Thickness (*OIT*). This frame-of-reference object does not have to be a wire. A frame-of-reference object can be any object (i.e., a transmission line insulator) of known diameter, which is likely to be seen when taking pictures of transmission lines. Transmission line insulators are common and would make excellent frames-of-reference in an industrial version of the TLIM system (see Fig. 5.3 for two examples).

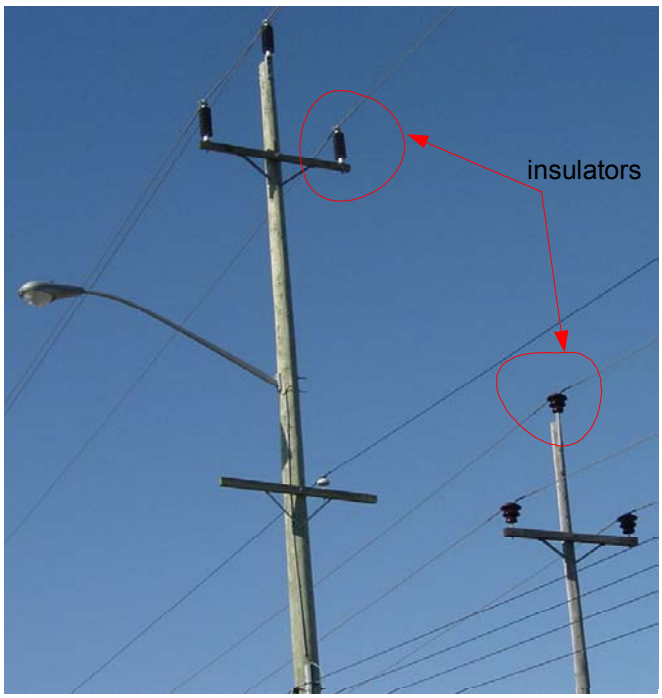


Fig. 5.3 Insulators as Frames-of-reference

For the purpose of creating a calibrating set, 279 pictures were taken, all under laboratory conditions from a distance of approximately 2 metres. In order to simulate real

conditions where access to transmission lines depends on their location and results in various distances between camera and the actual wire, seven different digital camera zoom settings were used. Since all of the pictures of transmission line wire were taken from the same place, zoom change was followed also by a change of light proportions in the scene. Different levels of brightness was compensated by automatic camera aperture mechanism. In addition, for each picture the camera was rotated to reflect a variation in angle, which can be expected when pictures of transmission lines are taken. Therefore, data obtained in the experiments are considered to be sufficiently diverse with respect to angle, distance and camera aperture values. Sample pictures made for this experiment are shown in Fig. 5.4.

Fig. 5.4 Sample Wire Images from Calibrating Set.

In order to create a map of Scale Factor values, additional knowledge about correct wire thickness is required. To verify that it is impossible to find the Scale Factor only from a wire image, consider see what would happen if we do not have any other point of reference. Suppose, for example, we have an image with a wire of known diameter only. By using the methods described earlier, namely, levelling the wire image, we can quite precisely measure its Observed Wire Thickness. However, this is the only information we can learn from the image. It turns out to be impossible to connect *OWT* to some variables, called in this thesis with one word – *magnitude*, describing wire size in the image, namely, distance between wire and the camera, digital camera zoom settings and resolution. Magnitude is somehow included in a *OWT* measurement, but with no additional point of reference it is hidden for us, and we cannot extract it (it is similar to a

case, where we would like to make a boat move, without any interaction with elements of the outer world like water, air, magnetic field, etc.).

The point of reference is another object with known diameter, which must be at the same distance as the wire. In this project, for better accuracy, the point of reference is thirty three times larger than the wire (shown in Fig. 5.5). It is important to note that we can measure the point of reference in an image in pixels. By dividing the known size of a reference point or what we referred to earlier as a frame-of-reference by the number of pixels, we obtain a thirty three times more precise Scale Factor than using the wire only as a reference point. The information gleaned from the point-of-reference is used to calibrate the TLIM system. The real Scale Factor used in the system is calculated by dividing point of reference's *SF* by the wire's *SF*.

Example 5.1. Suppose we have an image showing a wire of diameter 9.1 millimetres and a point of reference of known size 309 millimetres (see, e.g, Fig. 5.5). Assume we use a 4 times scaling to improve resolution. Image measurements show that the wire is 3.75 pixels wide and the point of reference is 79.5 pixels wide. By dividing 79.5 pixels by 309 millimetres we obtain

$SF_{\text{reference}} = 79.5 / 309 = 0.257$ pixels/mm.

Similarly,

$SF_{\text{wire}} = 3.75 / 9.1 = 0.412$ pixels/mm.

Therefore, final scale factor for this aperture, magnitude and angle equals

$SF(aperture, magnitude, angle) = SF_{\text{reference}} / SF_{\text{wire}} = 0.257 / 0.412 = 0.62$

If we apply this information to a picture with a ice-coated wire made with a similar aperture, magnitude and angle and we want to measure ice thickness, all we have to do is measure ice thickness in pixels. Then we compute the wire Scale Factor (found from the

same picture) and after multiplying it by 0.62, we obtain the final Scale Factor which is multiplied by the ice measure in pixels to gives us an ice measure in millimetres. One of the 279 sample images used to calibrate the TLIM system is shown in Fig. 5.5.

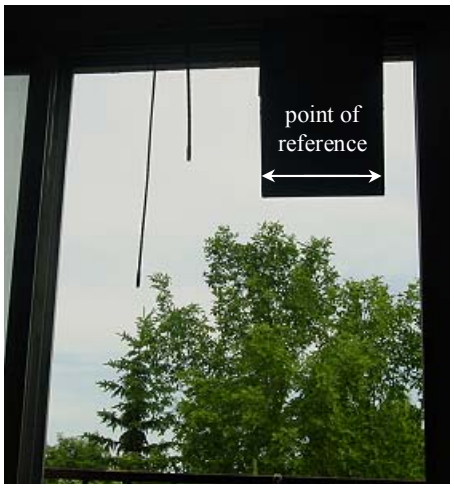


Fig. 5.5 Sample Calibrating Image.

Fig. 5.6 Aluminum Foil Around Wire.

5.2. Implementation

The entire TLIM system has been implemented in Matlab (this can be easily changed to visual C++ or Java SDK for a laptop such as an IBM ThinkPad or Java Micro for a PDA (personal digital assistant) such as a Compac i-pack to be used by line-maintenance engineers). Note that the data acquisition portion of the TLIM system (i.e., image captures) can be carried out by either a stationary or mobile robot as part of robotic inspection system for power transmission equipment (lines, towers, insulators, transformers, and so on). The version of the TLIM system described in this thesis is only a prototype of a full-featured system that can eventually be used in an industrial setting. The Graphical User Interface of the TLIM system presented in this thesis uses some features from version 6 of Matlab, so this version is required for best performance. There are two windows available for the user of the TLIM system. One window is for preparation of data images, and the other window is used to perform measurements. Figures 5.7 and 5.8 show images of these windows with numbers denoting available edit dialogs and buttons.

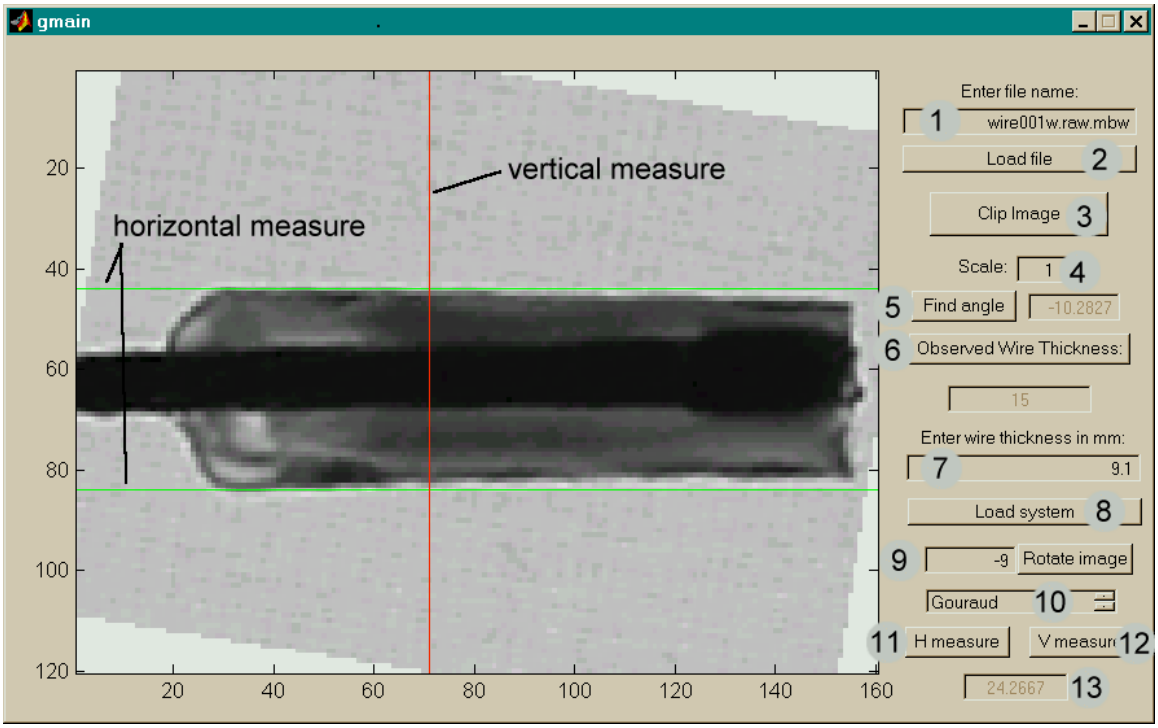


Fig. 5.7 Main Window of Wire Thickness Measuring Software.

First, an image must be loaded into the TLIM system using button (2) in Fig. 5.7. The filename of an image can be typed into the dialog box (1). Two types of files are recognised by the TLIM system: JPEG and internal program files with extension .raw.mbw (more about this format is explained in the part devoted to clipping images). Loaded image can be clipped by pressing button (3) to open the clipping window. After returning from the clipping window, we see a pre-processed image in the main display. The edit dialog button (4) sets the required scale. This scale number must be an integer and controls how many times an image is going to be zoomed during all operations. Higher values of the scale number result in more precise measurements, but also longer computational time.

Button (5) in Fig. 5.7 calls a procedure, which determines the angle between wire and horizon. This angle shows up in the text dialog on right from the button (5) after calculations are done. The next step is to calculate the Observed Wire Thickness (OWT) using the previously calculated angle. An appropriate function is called by clicking

button (6). The result is displayed in the dialog box below button (6). In the dialog box (7), a user is supposed to type the actual wire thickness in millimetres.

Pressing button (8) loads the TLIM system into memory. The TLIM System constructs a table, which contains Scale Factors for each image used in calibration, and corresponding pairs <Observed Wire Thickness, angle>. This information is used to approximate the Scale Factor for new samples. Dialog (9) in Fig. 5.7 defines the angle by which an image is rotated after pressing the button next to it. By default, the wire image rotation angle value is copied from dialog (5) after a wire's angle is found. Nevertheless, because of uneven ice shape a user may wish to type into dialog (9) a different angle to level the ice. Listbox (10) is designed to allow the user to select an algorithm, which is used for approximating Scale Factor values. Possible box (10) alternatives are:

- none (none system, result is given in pixels),
- mean value (*SF* is just average of all *SF*'s in the system),
- nearest neighbourhood,
- Gouraud shading and
- nearest triangle Gouraud shading.

Finally, measurement is taken using buttons (11) or (12). After selecting "Horizontal measure" a big cross controlled by a mouse appears on the main display (this is shown in Fig. 5.7). In this mode, a user uses only the horizontal bar. The user is supposed to place the bar on one edge of measured piece of ice. After clicking the left mouse button, an identical crossed bars show up again. The horizontal bar is supposed to be placed on the second edge of the wire. After clicking the left mouse button again**,** a calculated ice-thickness measure in millimetres is shown in text dialog (13). Both horizontal and vertical lines stay on the main display and changing the approximation algorithm (10) causes immediate update of the ice-measurement results. An alternative measurement method "Vertical measure" requires that the user select only one line. In this mode the user uses the vertical line to make particular ice-thickness measurements. The program detects ice borders on the cross-section created by this line and displays the result in millimetres in dialog box (13).

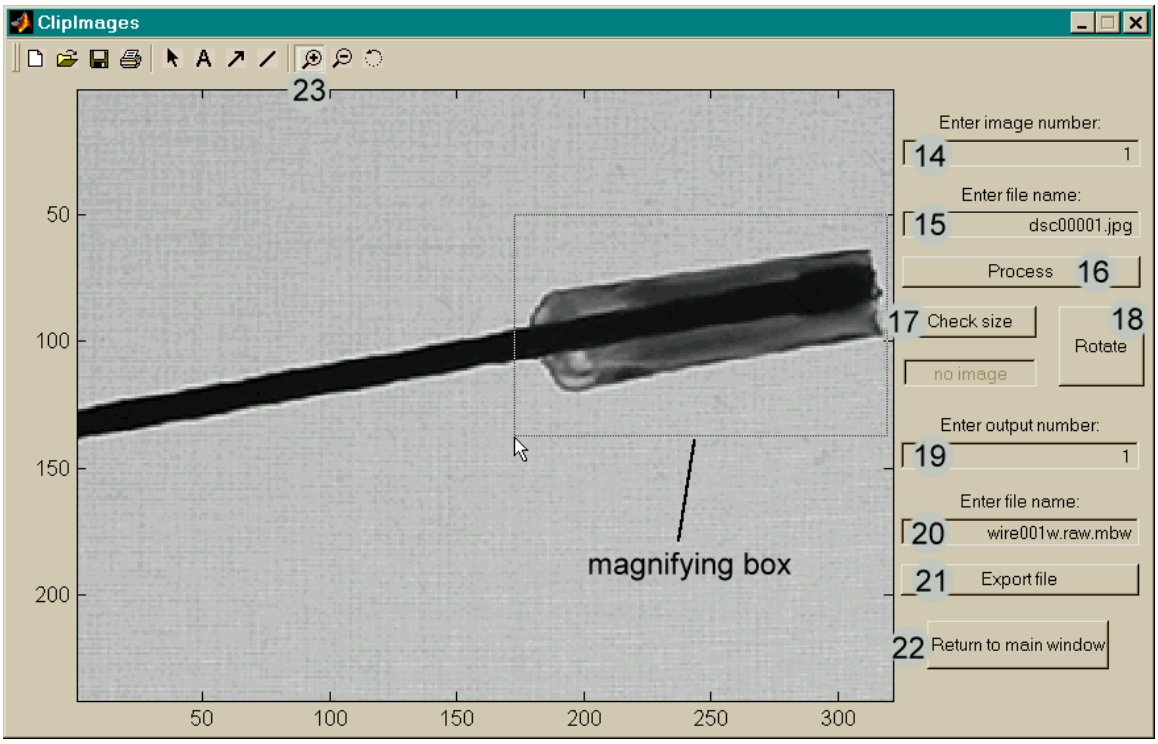


Fig. 5.8 Window for Clipping Images.

Figure 5.8 shows a window designed for image clipping. This window can be used for two separate tasks: preparation of calibration data images or clipping areas of interest found in the main window. Next, a brief description of the functionality of the image clipping window in Fig. 5.8 is given.

Dialog boxes (14) and (15) in Fig. 5.8 are used to type the name of an input file. If in (14) only a number is typed, it is automatically extended to the full filename used by the SONY® CD Mavica Digital Still Camera. Clicking (16) loads the selected file. A number typed in (14) is increased by one and dialog box (15) is automatically updated. The purpose of this procedure is to speed up the process of clipping many images. After an image is loaded, the user is allowed to use any of the standard Matlab "plotedit menu" tools. In particular, using (23) activates the magnifying box, which is an easy way to select an interesting wire image area. A button (17) returns the actual image size in a dialog box below it. In order to comply with the requirement that a wire cannot create a wider angle with the horizon than 45 degrees, a button (18) causes the whole image to be rotated by 90 degrees.

The output file name is controlled by dialogs (19) and (20) in Fig. 5.7. The mechanism of assigning filenames to numbers is similar as with input file names. Saving a file in .raw.mbw format is performed by clicking (21). This format was created specially for this project. It consists of a four bytes header: two bytes for the image width, two for the image height, and the raw data of image ordered row wise. Button (22) is invisible in this mode.

If the clipping window is opened from the main window (using button (3)), then there is a different procedure concerning the handling of input and output files. If there was any image opened in the main window, it is also opened in main display. This frees the user from typing the same filename twice. In addition, button (22) returns to the main window and automatically reopens a clipped file in its current dimensions.

5.3. Practical examples

In this section we will see some more practical applications of TLIM system. We use pictures of power transmission lines covered with ice form ice-storm, which took place in Quebec during January, 1998. These pictures were scanned from illustrations from a book (see [1]). Therefore, we show application of TLIM system when input images were prepared with equipment different than digital camera.

As we said in previous sections, in order to process any images with ice accumulated on wire, we need have access to database of wire images to calibrate the system. For images from ice-storm in Quebec we have only two images of wires with no additional data. Therefore, we make some assumptions, which allow us to perform ice thickness measurements. However, for industrial applications, actual data must be used to calibrate the system.

Fig. 5.9 1998 Quebec Hydro Transmission Lines [3]

Consider the image in the Fig. 5.9 taken from [3]. It shows several transmission lines covered with thick layer of accumulated ice. What makes Fig. 5.9 different from any previously considered images, is that the background is not plain. Instead of the sky we see trees, leafs and other wires. In addition, the compression level is set high resulting in red and green spots. All of these factors make it very difficult to retrieve the information about ice thickness. For the purpose of this experiment a small area (denoted by dashed line on Fig. 5.9) was selected. It contains part of the transmission line on which we want to concentrate. A digital camera was not focused on a selected image fragment, but the wire is the thickest in that part, resulting in more information than in any other location.

The wire from selected part in Fig. 5.9 is assumed to be 9.1 mm thick. Therefore, a database built for calibration purposes on a 9.1 mm transmission line wire obtained from Manitoba Hydro can be used. In Fig. 5.10 an enlarged selected part from Fig. 5.9 is shown. This image cannot be processed by the TLIM system without any modifications for two reasons. First, in the background other wires are visible, whereas input images for the TLIM system are assumed to contain only one wire. Second, the background is darker

than the actual wire. This makes it impossible to distinguish pixels belonging to a wire from pixels belonging to a tree in the background. Therefore, prior to processing by TLIM the image from Fig. 5.10 was pre-processed manually by removing all details from the background and changing the background colour to white. As we can see from Fig. 5.10 neighbouring pixels for the ice right below the wire are brighter than corresponding pixels above the wire. In order to avoid this situation, when pixels belonging to a ice could lead to faulty decisions (for example, darker pixels above the wire could be classified as a part of wire) two different shades of background were used. The result is shown in Fig. 5.11.

Fig. 5.10 Wire Covered With Ice.

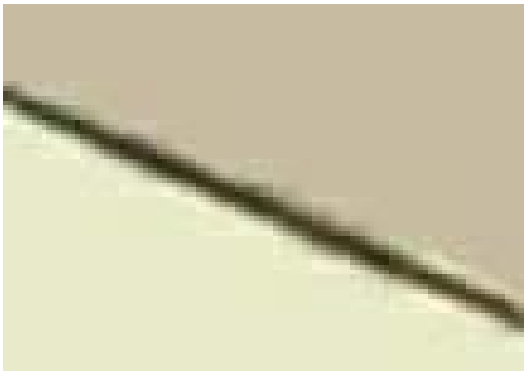


Fig. 5.11 The Same Wire as in 5.10, but With Removed Background.

The image in Fig. 5.11 contains the same information about the wire edges as in Fig. 5.10, but has no additional and useless for details for input to the TLIM system. Indeed, in Fig. 5.12 is shown the wire from Fig. 5.11, but perfectly levelled by the TLIM system.

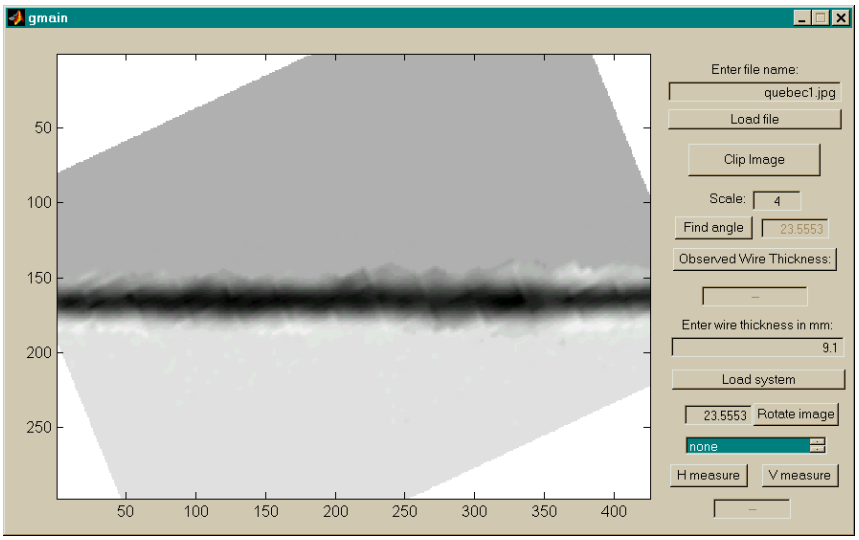


Fig. 5.12 Levelled by TLIM wire from Fig. 5.11

In Fig. 5.13 is shown the wire's cross-section produced by TLIM with denoted points, detected by TLIM as wire edges. Notice, because of different brightness of the background, wire edges are located at different grey levels. This is a very rare situation, since the sky characteristically contains a small frequency of changes. Nevertheless TLIM has no problems with the finding correct points for wire edges.

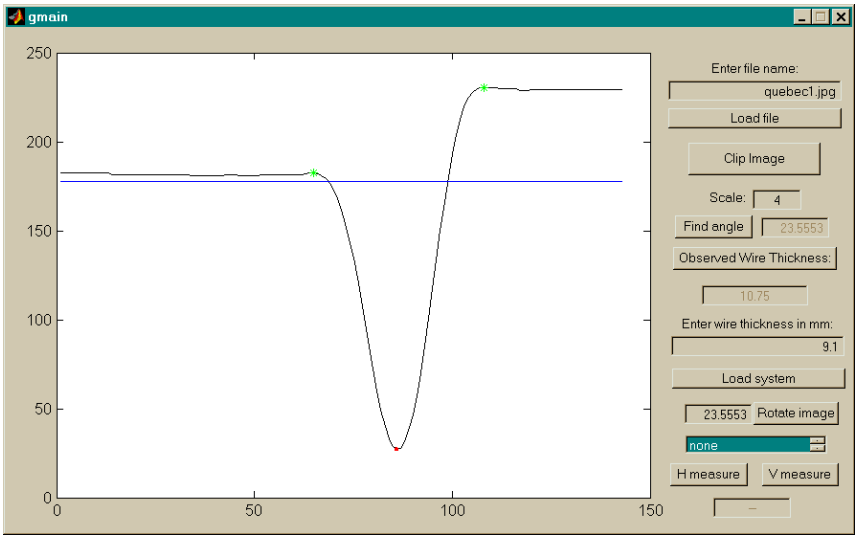


Fig. 5.13 Cross-section of a Wire from Fig. 5.12

After the system is loaded we can start taking measurements. As we can see from Fig. 5.14 it is really hard to determine upper edge of a wire. This problem results from a camera improperly focused on a wire. The final estimated measurement of the ice shown in the 1998 wire image is 25.65 millimetres.

To obtain deeper insight into the problem of validating the calculated thickness let us concentrate on sources of error. There are three main factors, which could cause ice thickness measurement error:

- bad approximation of Scale Factor;
- not precisely locating of upper ice edge;
- not precise locating of lower ice edge.

Let us start from the last two factors. Using the TLIM system a thickness of the boundary region for possible locations of upper and lower ice edges was calculated (see Appendix A for more information from Rough Set Theory). They are 3.1 and 1.5 millimetres long, respectively (see Fig. 5.14, where boundary regions are denoted by horizontal yellow lines). Assuming the worst case scenario, we could measure ice thickness 3.1/2+1.5/2=2.3 mm too small or 2.3 mm to big. Thus, the error made during measuring ice thickness is approximately ±2.3 mm and the range for possible ice widths is 23.35 mm to 27.95 mm.

Now, consider the error caused by estimation of the Scale Factor. From experiments performed for this thesis, we know that the Scale Factor approximation does not introduce error greater than 0.5% of wire thickness. Taking into consideration the range of possible wire widths we have obtained so far and the error caused by bad approximation of the Scale Factor (assuming the worst case scenario), we get the new range 23.23 mm to 28.09 mm. (we get it from 23.35*99.5% = 23.23, and 27.95*100.5% = 28.09)

Therefore, the measurement of 25.65 mm for the 1998 ice-covered transmission line can be 2.42 mm too long or 2.44 too short, which gives an error rate of ±9.5%. Considering that picture in the Fig. 5.9 was not taken for purpose of this measurement, this is a quite good result. For images dedicated for purpose of measuring ice, namely,

pictures focused on ice with better resolution and the sky in the background only, an even smaller error rate is expected.

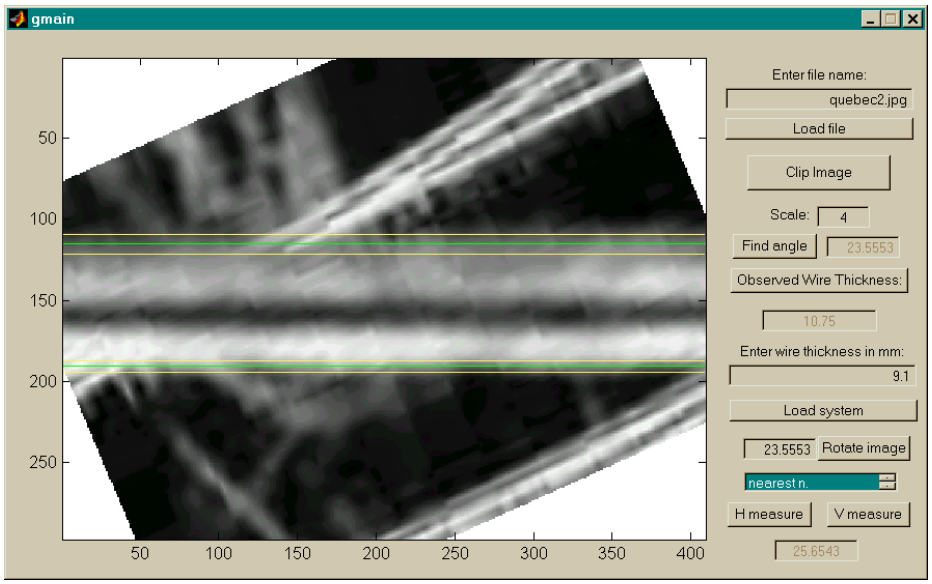


Fig. 5.14 Measurement bars superimposed on wire image

Bibliography

[1] I. MacAlpine, Bell Canada lines coated with ice on Counter Street in Kingston, Ontario on January 9, 1998. In: M. Abley, *The Ice Storm: An Historic Record in Photographs of January 1998*. CA: The Montreal Gazette, 1998, p. 126.

[2] I. MacAlpine, Bell Canada lines coated with ice on Counter Street in Kingston, Ontario on January 9, 1998. In: M. Abley, *The Ice Storm: An Historic Record in Photographs of January 1998*. CA: The Montreal Gazette, 1998, p. 47.

6. Performance

In this section we discuss some results showing TLIM system performance. Most experiments were performed on bare wire only, since this value is the base of all calculations and has the most influence on the whole system. System performance for estimating ice thickness was also tested. For validating ice measurements, actual ice thickness was measured with a micrometer and compared with the value returned by the TLIM system.

For all measurements a database consisting of 279 images containing 9.1 mm transmission wire was used as a calibrating set. In order to improve precision, a scale factor equal to 4 was used. For test purposes, wires of diameters 9.1 and 7.3 mm were used. To make the validating process more reliable a cross-validation technique was used. For each test, a database set was divided into calibrating and testing set. To measure system accuracy with respect to the size of the calibrating set, nine different calibrating set sizes were used, namely, containing 23, 56, ..., 251 images, which corresponds to 10%, 20%, ..., 90% of entire database. Each test was performed several times, each time a calibrating set was randomly generated from the database, and the results were averaged. The testing set was always created from all remaining images from the database after selecting a calibrating set. To sum up, a string c1000(28/251) denotes the following testing procedure:

1. A calibrating set consisting of 28 images from 279 images is chosen.
2. A testing set consisting from all 251 remaining images is chosen.
3. The error using the calibrating and testing set is calculated.
4. The testing procedure is repeated 1000 times from point 1.
5. All partial results from point 4 are averaged.

6.1. Scale Factor Approximation

We begin from tests performed on a transmission line wire only. For this test, we were trying to calculate the correct Scale Factor from an image of a wire. Since all

pictures contain a frame of reference, we can calculate the Scale Factor much more precisely. The error is the absolute value of the percentage difference between Scale Factor calculated from a wire based on TLIM system and calibrating set, and the optimal one found using a frame of reference as in (6.1)

$$\text{Error} = |\, 100*(SF_{\text{TLIM}} / SF_{\text{frame of ref.}} - 1)|\ \% \qquad (6.1)$$

The percentage in (6.1) can be directly converted into millimetres, provided we know the wire diameter.

First, consider the influence of the aperture parameter on the system error. The purpose of aperture is to maintain a balance between *OWT* and the angle of a wire relative to the horizontal during the process of approximating the Scale Factor value. It is supposed that there is an optimum value, which reflects a correlation between these two factors resulting in the best performance. In the following discussion, the aperture denotes the relative proportions between *OWT* and angle. For example, for aperture equal to 1, there is no scaling of *OWT* values. Both, *OWT* and angle are used in any of the three implemented algorithms, namely, nearest neighbourhood, Gouraud shading and nearest triangle Gouraud shading, as they were calculated from the image. For higher apertures, *OWT* is more important. For example, if aperture equals 10, all *OWT* values are multiplied by 10 before any calculations.

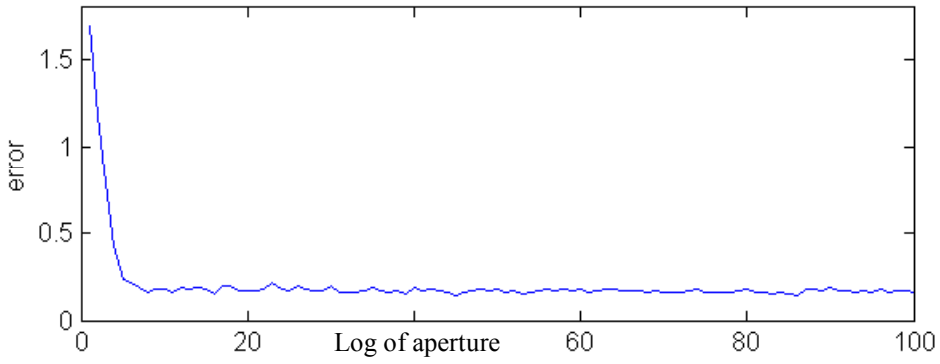


Fig. 6.1 Percentage Error vs. Aperture Value.

In Fig. 6.1 a plot showing the dependency of the approximating Scale Factor error on aperture is shown. Aperture values are shown in a logarithmic scale, the calibrating set consisted of 223 images, which is 80% of all available images. Each value comes from an average of 500 trials — c500(223/56) (cross validation 500 times, calibrating set 223 images, testing set 56 images). Error is given as a percentage. Fig. 6.1 shows the plot obtained for the nearest neighbourhood algorithm, but is representative for all three algorithms and will be discussed in general and conclusions drawn from it will be generalised into the Gouraud shading and the nearest triangle Gouraud shading algorithms.

From Fig. 6.1 we can see a strong correlation between aperture value and the system error, namely, the bigger aperture the smaller the error. This means that the more we trust *OWT* values and neglect the angle, the better the result we obtain. This is an unexpected result, however very optimistic. Independence from the angle means that we have only one variable on which Scale Factor depends. The search space simplifies to one dimension only, making the system simpler and faster. In other words, TLIM is angle invariant! The user does not have to be concerned about the angle of a camera nor the angle of the transmission line wire when taking the pictures. This is a very important finding, since it greatly simplifies the future use of the TLIM system by transmission line maintenance engineers using the same kind of camera used for this research.

I would rather not generalise angle independence to systems where pictures come from different sources (i.e., different than the digital camera used in this project). As we have mentioned before, aperture depends on digital device properties and can be different for other devices. One can imagine a situation where there is a different number of pixels in the horizontal and vertical directions and this is not properly corrected by camera (or computer) software. In this case, the same wire will have different thickness depending on the angle relative to horizontal picture edges. In order to correct this problem (if it arises with a different camera), the angle must be included in calculations of the Scale Factor. Therefore, the TLIM model includes angle as a parameter that the Scale Factor depends on. This makes the TLIM universal and flexible to ensure its independence from used hardware.

Let us now compare the result from Fig. 6.1 with a Scale Factor map for the nearest neighbourhood method. Consider Fig. 6.2, where the horizontal axis denotes *OWT* values, the vertical axis denotes absolute values of the angle and the Scale Factor is denoted by different colours of points (dark denotes small *SF* values and white denotes big *SF* values). The fact that the approximation of Scale Factor is independent of angle means that on the plot in Fig. 6.2, colour will not change along the vertical axis. Indeed, plot shows some regularities. All points in the same column seem to have the same colour. The fact that the colour borders are not aligned vertically comes rather from the fact that we use a small number of data than from some dependency on angle.

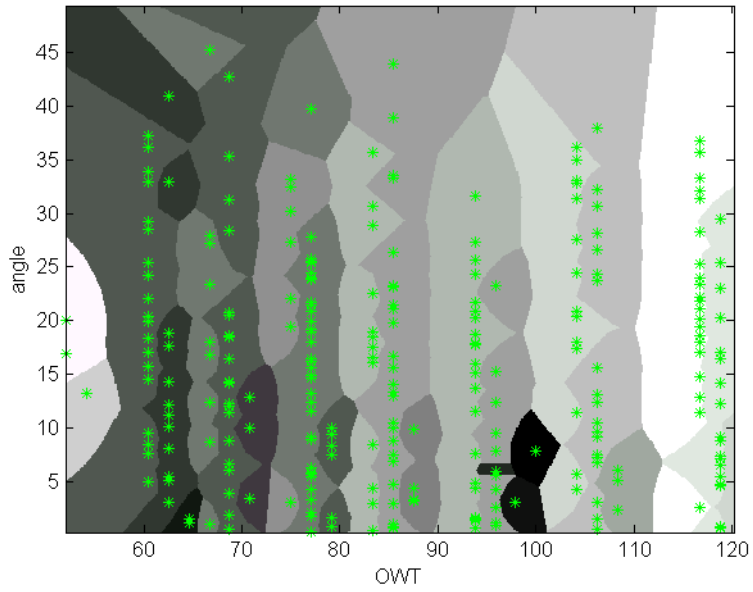


Fig. 6.2 Scale Factor Map for Nearest Neighbourhood Alg. (aperture 8.3333)

Fig. 6.2 shows a Scale Factor map for aperture equal to 8.3333. In Fig. 6.3 the Scale Factor for aperture equal to 1000000 is plotted. Colour edges have vertical direction resulting in better classification than in previous case.

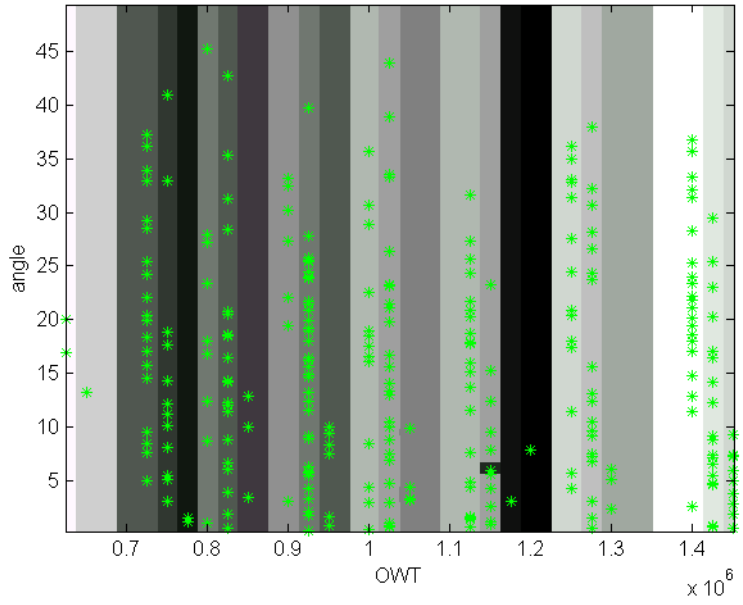


Fig. 6.3 Scale Factor Map for Nearest Neighbourhood Alg. (aperture 1000000)

A dependency between aperture and calibrating set size is shown in Fig. 6.4. As expected, the bigger calibrating set size the smaller the error. Also the bigger aperture the better accuracy (as discussed earlier). This scheme is identical for Gouraud shading and nearest triangle Gouraud shading algorithms.

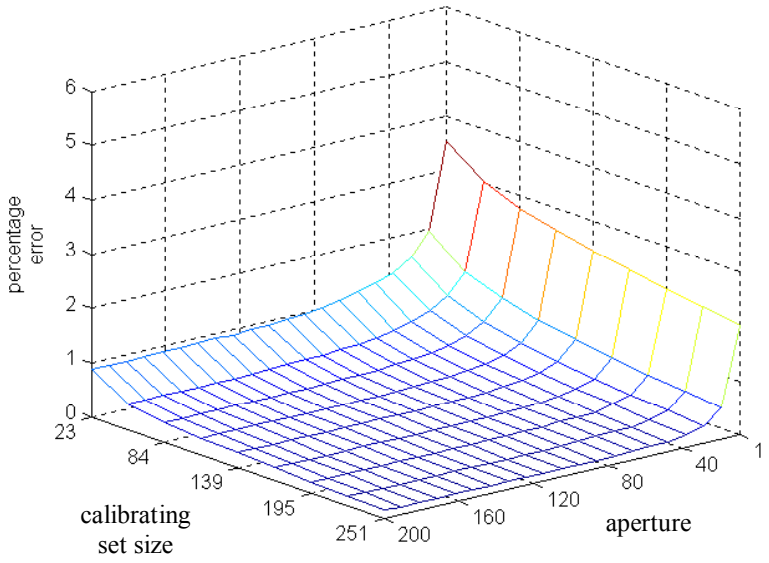


Fig. 6.4 Calibrating Set Size vs. Aperture (c1000)

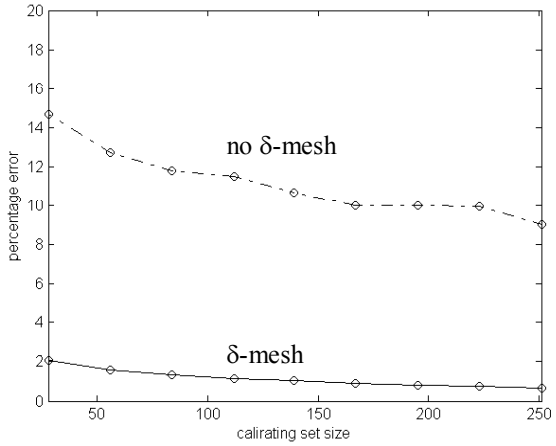


Fig. 6.5 Error vs. Calibrating Set Size for Nearest Neighbourhood

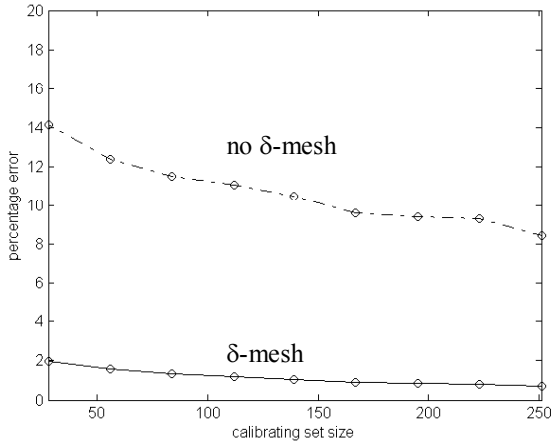


Fig. 6.6 Error vs. Calibrating Set Size for Gouraud Shading

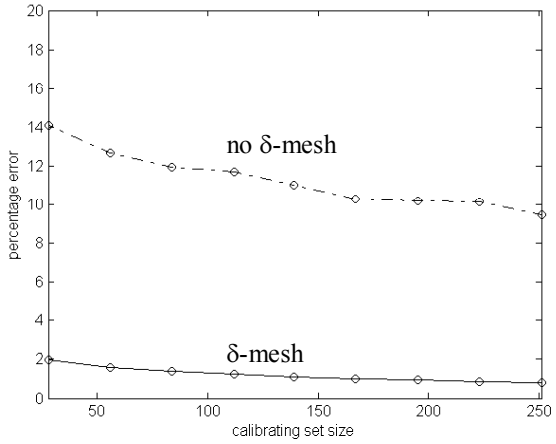


Fig. 6.7 Error vs. calibrating set size for nearest triangle Gouraud shading

In Figs. 6.5, 6.6 and 6.7, plots of errors calculated with and without δ-mesh are shown. The aperture was set to 8.3333 in all these figures and the results come from 1000 trials on average. As expected, error is many times greater (approximately 7 times) for the TLIM system where δ-mesh was not used. This is caused by the fact that the noise is not controlled in the non-application of the δ-mesh resulting in very poor angle estimation. There is also no significant difference between the results for different Scale Factor approximation algorithms. This fact confirms the observation that this huge error was introduced in a stage proceeding approximation phase, namely, in the image processing stage.

Let us now turn to a comparison between the three approximating algorithms, namely nearest neighbourhood, Gouraud shading and nearest triangle Gouraud shading. The image showing all three plots is shown in Fig. 6.8. In order to get the best results, tests for Fig. 6.8 were performed with 50 cross validation cases, with the aperture equal to 10^6. The error was measured for Scale Factor approximation. For 9.1 mm wire, 1 % is roughly one tenth of a millimetre.

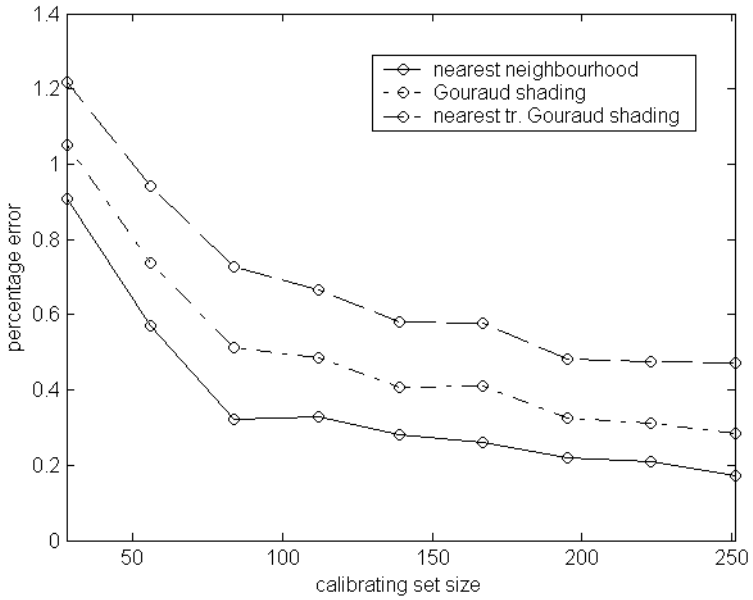


Fig. 6.8 Comparison of Three Approximating Algorithms

As we can see the nearest neighbourhood method gives the best result regardless of calibrating set size. This is an unexpected result. It suggests, that the approximating Scale Factor values between points results in worse accuracy than just returning the Scale Factor value of the closest point. A dependency of the Scale Factor only on *OWT* can also indicate that constructing triangles, which spread in the *angle* direction, takes into account points, which are too far from the point of interest. A hypothesis drawn is that for a system where hardware used for taking pictures is rotation invariant (as like in this case), then the nearest neighbourhood method outperforms the Gouraud shading methods only if there are enough calibrating points. In order to verify this hypothesis, additional tests with very small calibrating set sizes were performed. In Fig. 6.9 the results of these tests are presented. The performance order is now reversed. For small sets, the nearest neighbourhood method returns the worst error. This means that when calibrating points are spread, "shading" approximation is much more efficient than taking the *SF* value from the nearest point. It is also important to mention**,** that in industrial applications, the calibrating set will cover only a small percentage of all possible testing samples. We can only take a small number of pictures when calibrating the system and real life situations can lead to a huge number of possible transmission line wires relative to camera locations. Thus, the Gouraud shading model has potential possibilities for great performance in real life applications.

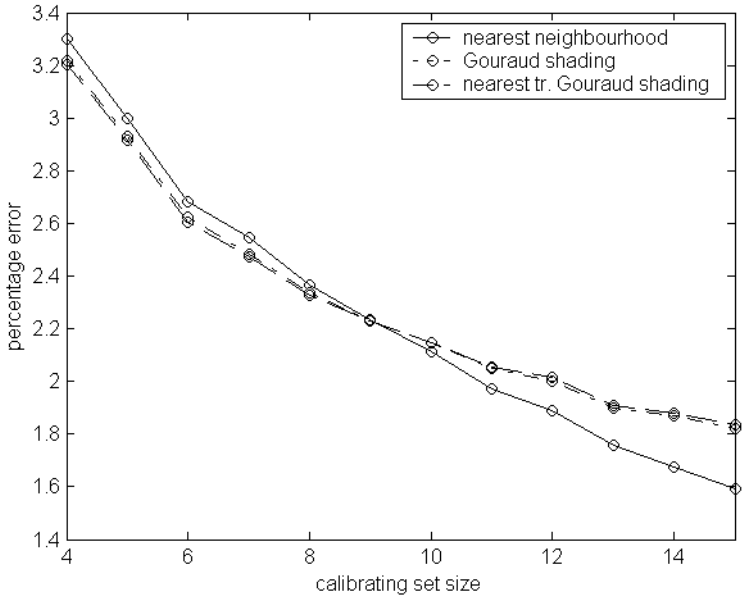


Fig. 6.9 *SF* Approximation Error for Very Small Calibrating Set Sizes

A table 6.1 contains numeric values plotted in Fig. 6.8.

calibr./testing	SF average	nearest n.	Gouraud sh.	n.tr. G. shading
251/28	4.6616	0.1709	0.2841	0.4721
223/56	4.6436	0.2081	0.3129	0.4735
195/84	4.5598	0.2182	0.3238	0.4804
167/112	4.6924	0.2609	0.4113	0.5770
139/140	4.6742	0.2807	0.4061	0.5817
112/167	4.6349	0.3300	0.4845	0.6671
84/195	4.6415	0.3235	0.5128	0.7263
56/223	4.6668	0.5703	0.7358	0.9404
28/251	4.7268	0.9080	1.0493	1.2189

Table 6.1 Percentage Error for TLIM System

6.2. Ice Measurement

In the remaining part of this chapter, we show one more example of ice measurement. For this purpose, a wire of diameter of 7.3 millimetres was used. The wire was covered with ice using method described in section 5. An image of a wire coated

with ice was taken in the same laboratory conditions as images for the calibrating set, e.g. from approximately 2 metres. The system was built based on 279 images showing 9.1 mm transmission line wire. Measurements were taken from three different points on the ice covering the wire (see Fig. 6.11). Simultaneously, the real ice thickness was measured using specialised ruler (see Fig. 6.10). Ice thicknesses estimates returned by the TLIM system were compared with actual wire thicknesses. The results are shown in Table 6.2. As we see, the error is very small, i.e., the error in TLIM system ice thickness estimates does not exceed 1 millimetre.

measurement location	measured thickness	actual thickness	error
1	27.45	26.85	3 %
2	31.3	30.33	3.2 %
3	29.3	28.34	3.4 %

Table 6.2 Ice Measurement Error for Image from Fig. 6.11

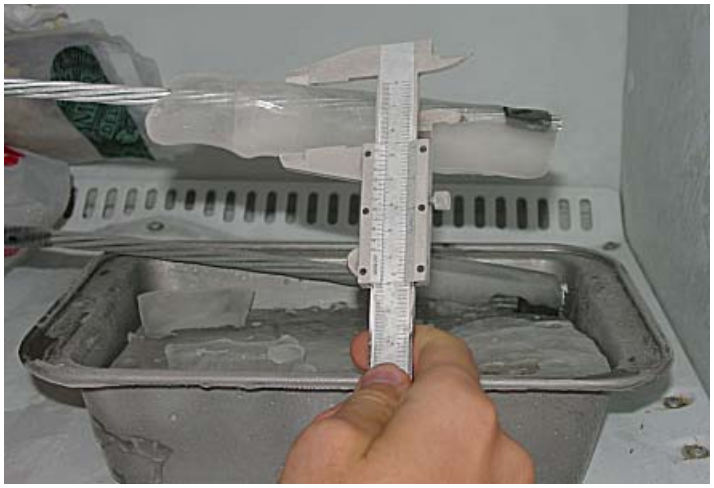


Fig. 6.10 Measurement of Ice Thickness

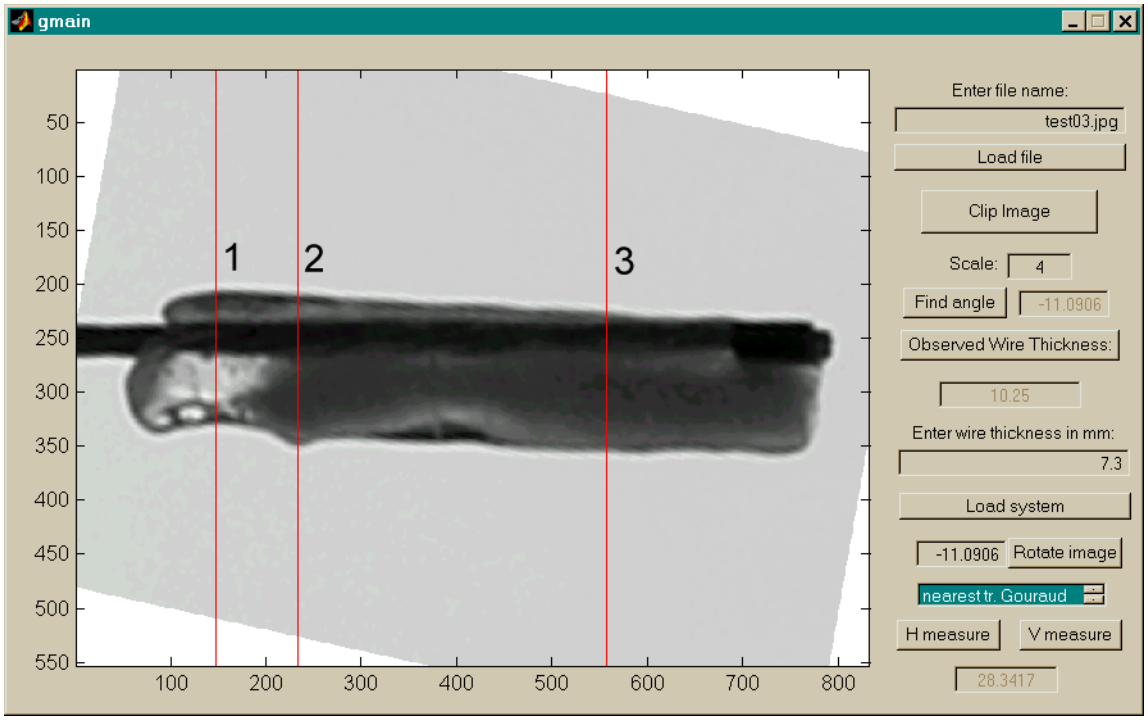


Fig. 6.11 Wire Covered With Ice with Denoted Measurement Points

7. Conclusion

In this thesis, a system for measuring ice accumulated on power transmission lines is presented. The principal goal of this research has been to accomplish the task using information contained only in digital images of power transmission wires covered with ice. This new approach combines rough set theory with classical methods from digital image processing.

The design of the measurement system is divided into two stages. First, a frame of reference is processed, which is in this case a bare wire, to find the appropriate scale factor to convert pixels from an image to millimetres. This stage's main part is based on the use of a δ-mesh tool. The δ-mesh is an extension of Rough Set theory. In the context of solving the ice-accumulation measurement problem, the δ-mesh is used to combat the noise contained in a digital image. Practical experiments proved the usefulness of the δ-mesh as a tool for filtering digital images.

The main objective of the first stage in measuring ice-thickness was to extract from a wire image information about the exact location of wire edges. The goal was to obtain better quality than one pixel. Therefore, techniques utilising pixel colour in addition to pixel location have been used. The δ-mesh has been combined with a linear interpolation method to detect the exact angle of the wire with respect to the horizontal image axis. Then the wire was levelled in preparation for the next stage in the measurement process.

In the second stage in measuring ice-thickness, wire edges are detected. The δ-mesh, min-max search and derivative processing are applied to make sure that for each image the same point representing an edge is detected.

The exact wire angle and proportion of actual wire edges extracted from the image wire edges are used as parameters for calibrating the ice-measurment system. This makes it possible to extend knowledge from a relatively small calibrating set into any case that can happen in industrial applications. The discrete to continuous case was extended with the use of the Gouraud shading algorithm. Non-conventional usage of this classical image processing algorithm in a form of pattern recognition is made possible by

introducing two additional algorithms for partitioning the parameter space into set of triangles. As comparison of these efficient, but complex methods with a simple nearest neighbourhood algorithm is also presented.

The implemented system has been tested on set of 279 digital images taken by a digital camera, as well as on images scanned from photographs of ice-covered transmission lines during the 1998 Eastern Canada ice storm. The error in estimating the scale factor to convert from pixels into millimetres is very low, not exceeding 0.5% for calibrating set sizes bigger than 50% of all available data. Measuring of ice resulted in a 3% error for images taken by a digital camera and a 10% error for scanned photographs. This is considered to be a good estimation and a big step forward in the design of a robotic approach to inspecting and maintaining power transmission lines systems.

Future work should be concentrated on developing the automation of the pre-processing step of digital images as well as detecting edges of ice. In the present version of the ice-measurement system, input images can contain only a transmission line wire with uniform sky in the background. Therefore, any image containing other details must be manually processed. Usage of common digital image processing methods like thresholding and the Hough transform can make this process entirely automatic. In addition, using a mathematical model for a wire hanging under the influence of gravity and image warping can extend even more the class of images suitable for the system. For example, these new methods can help make the system entirely invariant of the location of the digital camera and the transmission line wire.

A second problem that requires more future improvement is detecting the ice edges. The principal objective of this thesis has been to build the core of the system for image based distance measurements. Therefore, the focus was set on frame-of-reference objects (e.g., wire, insulators) to facilitate measurements. Nevertheless, ice is partially transparent (in a degree depending on temperature) and identifying its edges is totally a separate problem. This problem needs more study and can be a good starting point for future work in making this system even more accurate.

Bibliography

[1] J.J. Alpigini Measures of closeness of performance map information granules: A rough set approach. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 293-299.

[2] M. Borkowski, Signal analysis using rough integrals. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 218-225.

[3] M. Borkowski, J.F. Peters, Approximating sensor signals: A rough set approach. In W. Kinsner, A. Sebak, K. Ferens (Eds.), Proc. of the IEEE Canadian Conf. on Electrical and Computer Engineering 2002 (CCECE'02), 12-15 May 2002, Winnipeg, Manitoba, Canada, 66-72.

[4] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Prentice Hall, Upper Saddle River, New Jersey 07458, 2002.

[5] H. Gouraud "Continuous Shading of Curved Surfaces", IEEE Transactions on Computers, vol. 20, pp. 623-629, June 1971

[6] P.R. Halmos, Measure Theory. London: D. Van Nostrand Co., Inc., 1950.

[7] P.V.C. Hough, Methods and means of recognizing complex patterns, U.S. Patent 3,069,654, 1962.

[8] I. MacAlpine, Bell Canada lines coated with ice on Counter Street in Kingston, Ontario on January 9, 1998. In: M. Abley, *The Ice Storm: An Historic Record in Photographs of January 1998*. CA: The Montreal Gazette, 1998, p. 126.

[9] S.K. Pal, L. Polkowski, A. Skowron (Eds.), Rough-Neuro Computing: Techniques for Computing with Words. Berlin: Springer-Verlag, 2002.

[10] Z. Pawlak, Rough sets, International Journal of Computer and Information Sciences, vol. 11, 1982, 341-356.

[11] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data. Boston, MA, Kluwer Academic Publishers, 1991.

[12] Z. Pawlak, J.F. Peters, A. Skowron, Z. Suraj, S. Ramanna, M. Borkowski, Rough measures: Theory and Applications. In: S. Hirano, M. Inuiguchi, S. Tsumoto

(Eds.), Rough Set Theory and Granular Computing, Bulletin of the International Rough Set Society, vol. 5, no. 1 / 2, 2001, 177-184.

[13] Z. Pawlak, A. Skowron, Rough membership functions. In: R. Yager, M. Fedrizzi, J. Kacprzyk (Eds.), Advances in the Dempster-Shafer Theory of Evidence, NY, John Wiley & Sons, 1994, 251-271.

[14] J.F. Peters, T.C. Ahn, M. Borkowski, Obstacle Classification by a Line-Crawling Robot: A Rough Neurocomputing Approach. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 606-613.

[15] J.F. Peters, T.C. Ahn, M. Borkowski, V. Degtyaryov, S. Ramanna, Line-crawling robot navigation: A rough neurocomputing approach. In: D. Maravall, D. Zhou (Eds.), Fusion of Soft Computing and Hard Computing Techniques for Autonomous Robotic Systems. Studies in Fuzziness and Soft Computing, J. Kacprzyk (Ed.). Berlin: Physica-Verlag, 2002 [to appear].

[16] J.F. Peters, S. Ramanna, M. Borkowski, A. Skowron, Z. Suraj, Sensor, filter and fusion models with rough Petri nets, *Fundamenta Informaticae*, vol. 34, 2001, 1-19.

[17] J.F. Peters, S. Ramanna, Z. Suraj, M. Borkowski, Rough neurons: Petri net models and Applications. In: L. Polkowski, A. Skowron (Eds.), *Rough-Neuro Computing*. Berlin: Springer, 2002, 472-491.

[18] J.F. Peters, A. Skowron, Z. Suraj, M. Borkowski, W.Rzasa, Measures of Inclusion and Closeness of Information Granules: A Rough Set Approach. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 304-311.

[19] J.F. Peters, A. Skowron, Z. Suraj, W. Rzasa, M. Borkowski, Clustering: A Rough Set Approach to Constructing Information Granules. In: Z. Suraj (Ed.), Soft Computing and Distributed Processing, 6th Int. Conf., Rzeszow, Poland, 24-25 June 2002, 57-62.

[20] S. Ramanna, Rough neural network for software change prediction. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Rough Sets and Soft

Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 614-621.

[21] A. Skowron, C. Rauszer, The Discernibility Matrices and Functions in Information Systems. In: Slowinski, R. (Ed.), Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic Publishers, Dordrecht, 1992, 331-362.

[22] A. Skowron, J. Stepaniuk, J.F. Peters, Hierarchy of information granules. In: L. Czala (Ed.), *Proc. of the Workshop on Concurrency, Specification and Programming*, Oct. 2001, Warsaw, Poland, 254-268.

[23] A. Skowron, R.W. Swiniarski, R.W., Information granulation and pattern recognition. In: S. Pal, L. Polkowski, A. Skowron (Eds.), Rough-Neuro Computing. Berlin: Physica-Verlag, 636-670.

[24] G. Srikant, L.Wurtz, "A CMOS Parallel Gouraud Shading VLSI Architecture", IEEE 1992, http://ieeexplore.ieee.org

[25] http://www.du.edu/~jcalvert/math/catenary.htm

Appendix A (Brief Introduction to Rough Sets)

This Appendix gives a brief overview of some fundamental concepts and features of rough set theory that are important to an understanding of the underlying concepts in the approach to pre-processing digital images for input to the TLIM ice-measurement system. Rough set theory provides a suite of methodologies useful in the numerical characterisation of imprecise data [1]-[2].

I Set Approximation

For computational reasons, a syntactic representation of knowledge is provided by rough sets in the form of data tables. Informally, a data table is represented as a collection of rows each labeled with some form of input, and each column is labeled with the name of an attribute that computes a value using the row input. Consider, for example, a small table reflecting measurements of complexity of two software modules m1 and m2 using LOC (Lines-Of-Code) and McCabe's cyclomatic complexity metric V(G). If a decision column is added to the table, then the data table is called a decision table (see Fig. A.1).

input \ A	LOC	V(G)	changes
m1	276	43	10
m2	210	12	2

Fig. A.1 Sample Decision Table

In Fig. A.1, for example, module m1 has 276 lines of executable and commented lines of code and V(G) is equal to 43. In this case, the recorded number of changes is 10. In effect, this table encodes knowledge about metrics data for two software modules m1 and m2 and the corresponding changes made to these modules. Formally, a data (information) table is represented by a pair (U, A), where U is a non-empty, *finite* set of objects and A is a non-empty, finite set of attributes, where $a: U \rightarrow V_a$ for every $a \in A$. For each $B \subseteq A$, there is associated an equivalence relation $\mathrm{Ind}_A(B)$ such that

$$\mathrm{Ind}_A(B) = \{(x, x') \in U^2 \mid \forall a \in B.\ \mathrm{a}(x) = \mathrm{a}(x')\}$$

If $(x, x') \in \mathrm{Ind}_A(B)$, we say that objects x and x' are indiscernible from each other relative to attributes from B. This is a fundamental concept in rough sets. The notation $[x]_B$ denotes equivalence classes of $\mathrm{Ind}_A(B)$. Further, partition $\mathrm{U}/\mathrm{Ind}_A(B)$ denotes the family of all equivalence classes of relation $\mathrm{Ind}_A(B)$ on U. For $X \subseteq U$, the set X can be approximated only from information contained in B by constructing a B-lower and B-upper approximation denoted by $\underline{B}X$ and $\overline{B}X$ respectively, where $\underline{B}X = \{x \mid [x]_B \subseteq X\}$ and $\overline{B}X = \{x \mid [x]_B \cap X \neq \varnothing\}$. A lower approximation $\underline{B}X$ of a set X is a collection of objects that can be classified with full certainty as members of X using the knowledge represented by attributes in B. By contrast, an upper approximation $\overline{B}X$ of a set X is a collection of objects representing both certain and possible uncertain knowledge. Whenever $\underline{B}X = \overline{B}X$, the collection of objects can be classified perfectly, and forms what is known as a crisp set. In the case $\underline{B}X$ is a proper subset of $\overline{B}X$, then the collection contains objects that cannot be classified with certainty, and the pair $(\underline{B}X, \overline{B}X)$ is called a rough set.

II Rough Membership Set Function

In this section, a set function form of the traditional rough membership function was introduced in [3].

Definition A.1 Let $S = (U, A)$ be an information system, $B \subseteq A$, and let $[u]_B$ be an equivalence class of an object $u \in U$ of $\mathrm{Ind}_A(B)$. A set function $\mu_u^B: \wp(U) \rightarrow [0, 1]$ defined by (A.1).

$$\mu_u^B(X) = \frac{card(X \cap [u]_B)}{card([u]_B)} \qquad \textbf{(A.1)}$$

for any $X \in \wp(Y)$, Y ⊆ U, is called a *rough membership function*. A rough membership function provides a classification measure inasmuch as it tests the degree of overlap between the set X and an equivalence class $[\mathrm{u}]_\mathrm{B}$. The form of rough membership

function in Def. A.1 is slightly different from the classical definition [3] where the argument of the rough membership function is an object u and the set X is fixed. For example, let $X_{B_{approx}} \in \{\bar{B}X, \underline{B}X\}$ denote a set approximation. Then we compute the degree of overlap between $X_{B_{approx}}$ and $[u]_B$ in (A.2).

$$\mu_u^B(X_{B_{approx}}) = \frac{\text{Card}([u]_B \cap X_{B_{approx}})}{\text{Card}([u]_B)} \quad \textbf{(A.2)}$$

III Attribute Reduction and Decision Rules

Knowledge representation in the form of a discernibility matrix was first proposed by Skowron and Rauzer [4]. Informally, a discernibility matrix is an n × n table that distinguishes one input object from another based on attribute values. This representation has many advantages, since it enables efficient computation of reduced attribute sets called reducts and decision rules. An approach to finding a subset of attributes (reduct) with the same classificatory power as the entire set of attributes in an information system is briefly described in this section. This also leads to a brief discussion about the derivation of decision rules with minimal descriptions in their left-hand sides. In deriving decision system rules, the discernibility matrix and discernibility function are essential. Given a decision table DT = (U, A ∪ {d}), the n x n matrix (c_{ij}) called the discernibility matrix M of S (denoted M(DT)) is defined in (A.3).

$c_{ij} = \{a \in A \mid a(x_i) \neq a(x_j)\}$, for i, j = 1, ..., n. **(A.3)**

A discernibility function f_{DT} relative to discernibility matrix M for a decision table DT is a boolean function of m boolean variables $a_1^*, ..., a_m^*$ corresponding to attributes a_1, ..., a_m respectively, and defined in (A.4).

$$f_{DT}(a_1^*, ..., a_m^*) =_{df} \wedge\{\vee_{ij}^* \mid 1 \leq j \leq i \leq n,\ c_{ij} \neq \varnothing\},\ \text{where } c_{ij}^* \{a^* \mid a \in c_{ij}\} \quad \textbf{(A.4)}$$

The set of all prime implicants of f_S determines the set of all reducts of S. A reduct is a minimal set of attributes $B \subseteq A$ that can be used to discern all objects obtainable by all of the attributes of an information system. The reducts of a decision system $S_d = (U, A \cup \{d\})$ correspond to the prime implicants of the discernibility function f_{DT}. That is, $Ind_S(B) = Ind_S(A)$. In effect, a reduct is a subset B of attributes A of information system S that preserves the partitioning of the universe U. Hence, a reduct can be used to perform the same classifications as the whole attribute set A of the information system. The set of all reducts of S is denoted by RED(S). Let $B \subseteq A$. The set of all reducts in IS with attribute set B is denoted by RED(B). A method used to find a proper subset of attributes of A with the classificatory power as the entire set A has been termed *attribute reduction* [5]. Let f_M^d be a decision-relative discernibility function with respect to discernibility matrix M and decision table DT. This boolean function can be constructed from the discernibility matrix for DT. The set of all prime implicants of f_M^d defines the set of all decision-relative reducts of the decision system S_d. In other words, precise conditions for decision rules can be extracted from f_M^d derived from a discernibility matrix M as in [3].

Bibliography

[1] Z. Pawlak, Rough sets, International Journal of Computer and Information Sciences, vol. 11, 1982, 341-356.

[2] Z. Pawlak, Rough Sets: Theoretical Aspects of Reasoning About Data. Boston, MA, Kluwer Academic Publishers, 1991.

[3] Z. Pawlak, A. Skowron, Rough membership functions. In: R. Yager, M. Fedrizzi, J. Kacprzyk (Eds.), Advances in the Dempster-Shafer Theory of Evidence, NY, John Wiley & Sons, 1994, 251-271.

[4] A. Skowron, C. Rauszer, The Discernibility Matrices and Functions in Information Systems. In: Slowinski, R. (Ed.), Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic Publishers, Dordrecht, 1992, 331-362.

[5] A. Skowron, R.W. Swiniarski, R.W., Information granulation and pattern recognition. In: S. Pal, L. Polkowski, A. Skowron (Eds.), Rough-Neuro Computing. Berlin: Physica-Verlag, 636-670.

Appendix B (Mathematics Related to the Design of the TLIM System)

I Metrics

The nearest neighbourhood method results are dependent on the metric used when the evaluating distance between data points. Usually, finding the right distance measure is itself a separate question and the measure is the main factor influencing the resulting accuracy. Let us begin by recalling the metric definition for the two dimensional real ($\mathbb{R}$) plane.

Definition E.2 A **metric M** is a function $\mathbb{R}\times\mathbb{R}\rightarrow\mathbb{R}$ which for all vectors $a,b,c\in\mathbb{R}^2$ satisfies the following conditions:

- nonnegativity: $\mathbf{M}(a,b) \geqslant 0$
- reflexivity: $\mathbf{M}(a,b) = 0$ if and only if $a = b$
- symmetry: $\mathbf{M}(a,b) = \mathbf{M}(b,a)$
- triangle inequality: $\mathbf{M}(a,b) + \mathbf{M}(b,c) \geqslant \mathbf{M}(a,c)$

The most known family of metrics parameterised by value k is the Minkowski metric (called also the L_k norm):

$$L_k(a,b) = \left(\sum_{i=1}^{d} |a_i - b_i|^k \right)^{\frac{1}{k}}$$

where d is the dimension of the space.

Here, I describe briefly only three members of this family, namely, L_1, L_2 and L_∞.

- L_1 known also as the Manhattan or city block distance. A distance is measured only along directions that are parallel to coordinate axis

- L_2 is just the Euclidean distance, for the two dimensional real plane Euclidean distance is given by a formula $L_2(a,b) = \sqrt{(a_1 + b_1)^2 + (a_2 + b_2)^2}$.

- L_∞ is the maximum of projections of a and b onto each of the d coordinate axes.

In order to the visualise properties of each metric, in Fig. E.1 a plot of a circle of diameter 1 in each metric is shown.

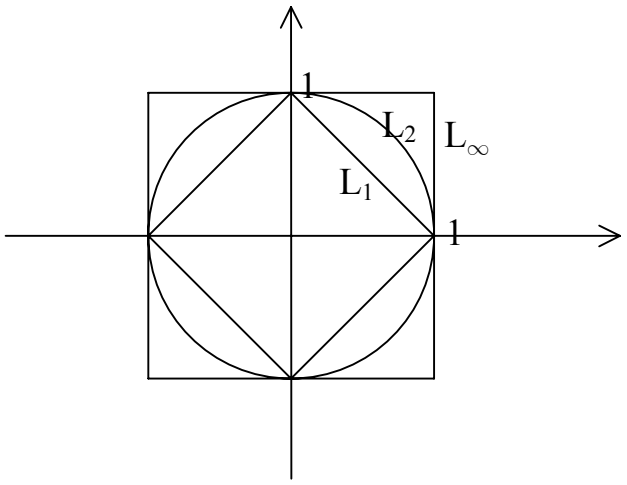


Fig. E.1 A Circle In Three Different Metrics.

II Convex Sets

The following definitions are cited from [5]. Convex set idea is used for description of area of parameter space covered by particular approximation algorithms, namely, outside convex hull of all calibrating points nearest neighbourhood method is always used.

Definition E.3 A set **A** is said to be ***convex*** if the straight line segment joining any two points in **A** lies entirely within **A**.

Definition E.4 The convex hull **H** of an arbitrary set **S** is the smallest convex set containing **S**.

III Chain Equation

The following theory may be useful in defining geometrical transformations to straighten the wire on processed image. This can extend the range of images capable for TLIM to process.

A mathematical formula describing a uniform and flexible chain hanging under the influence of gravity is called the ***catenary***. In order to describe the shape of the chain, which could be in our case a wire, we have to define few parameters characteristic for this wire. Let w denote the unit length of a wire, S the total length of a wire, L the span and h the sag (or deflection). If by x we denote the horizontal axis and x=0 denote the middle point of the wire, then for $x \in (-\frac{L}{2}, \frac{L}{2})$ a velocity of a wire is given by Eqn. E.1.

$$y = \frac{H}{w}(\cosh\frac{wx}{H} - 1) \qquad \text{(E.1)}$$

where H is constant. If not all of the variables are given, they can be found from the following additional equations:

- $S = (2H/w) \sinh (wL/2H)$,
- $L = (2H/w) \sinh^{-1} (Sw/2H)$,
- $H = (w/8h)(S^2 - 4h^2)$.

In Fig. E.2 a plot showing a shape of a wire stretched between two points is shown.

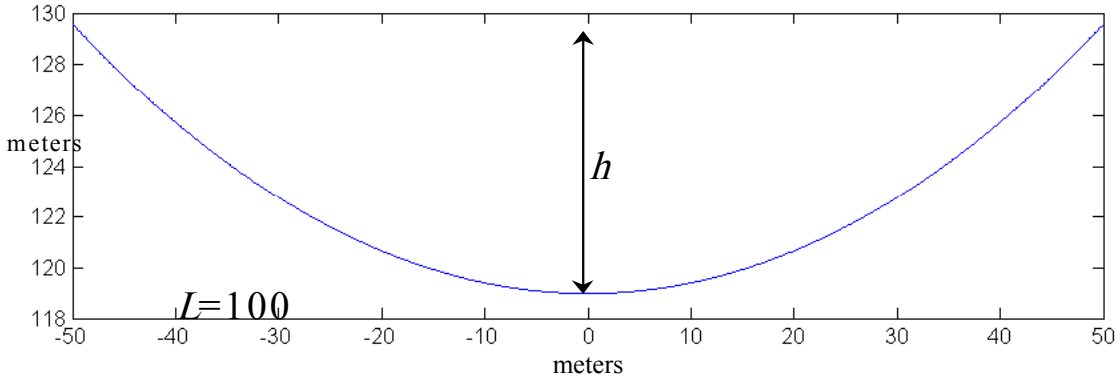


Fig. E.2 A Wire Shape Obtained From Catenary Equation.

IV Image Warping

Image warping is an algorithm for applying geometrical transformations to digital images. After finding the tiepoints using algorithm described in E.III we can use them to straighten the wire.

Geometrical transformations are used to remove (or add) from an image unwanted geometrical distortions (see, e.g., Fig. A.3). They are defined by selection a mesh of points, called tiepoints. One set of tiepoints is located in the input (distorted) image and the second set in the output (corrected) image. A mapping between input and output images is going to be exact at specified tiepoints. If by x and y we denote the coordinates of the corrected image, and by $\hat{x}$ and $\hat{y}$ the coordinates of the input image, then the First Order Warp equation is defined by Eqn. (E.2.

$$\begin{cases} \hat{x} = c_1 x + c_2 y + c_3 xy + c_4 \\ \hat{y} = d_1 x + d_2 y + d_3 xy + d_4 \end{cases} \qquad \text{(E.2)}$$

To find the eight unknown coefficients c_1, c_2, c_3, c_4 and d_1, d_2, d_3, d_4 we need to define four tiepoints. The coefficients can be calculated from simple equations by reversing the matrix A. Let

$$A = \begin{bmatrix} x_1 & y_1 & x_1 y_1 & 1 \\ x_2 & y_2 & x_2 y_2 & 1 \\ x_3 & y_3 & x_3 y_3 & 1 \\ x_4 & y_4 & x_4 y_4 & 1 \end{bmatrix} \quad \underline{\hat{x}} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{bmatrix} \quad \underline{\hat{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \hat{y}_4 \end{bmatrix} \quad \underline{c} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \quad \underline{d} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

then we can find coefficients $\underline{c}$ and $\underline{d}$ from equations A.3:

$$\underline{\hat{x}} = A\underline{c} \;\Rightarrow\; \underline{c} = A^{-1}\underline{\hat{x}} \quad \text{and} \quad \underline{\hat{y}} = A\underline{d} \;\Rightarrow\; \underline{d} = A^{-1}\underline{\hat{y}} \qquad \text{(E.3)}$$

It is also possible to select more than four tiepoints. In such a case, it is impossible to satisfy all the equations. A pseudo-inverse matrix is used, which minimises the mean squared error at all tiepoints. A pseudo-inverse matrix for a given matrix A is given by

$$\text{Pseudo-inverse}(A) = (A^T A)^{-1} A^T \qquad \text{(E.4)}$$

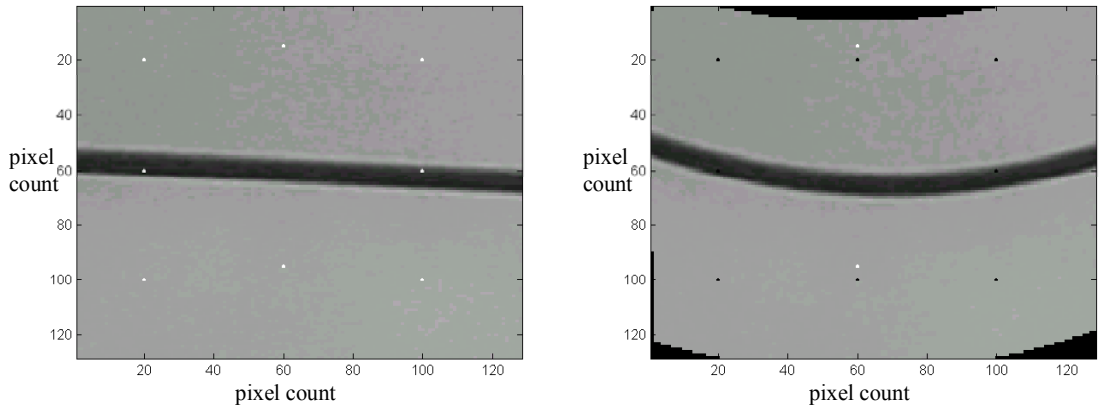


Fig. E.3 A Transmission Line Wire Bent By Means Of Image Warping.

V Hough Transform

In this section we shortly describe a concept useful in preprocessing of digital images before they can be processed by TLIM system. By using Hough transform, a transmission line wire can be identified on a picture containing more objects than just a wire. This makes possible to exclude the user from manual preprocessing of images.

The Hough transform is a mapping, which makes it possible to identify points lying on the straight line [**Error! Reference source not found.**]. Prior to applying the Hough transform an image must be preprocessed to make it consist of two lines. In other words, the Hough transform can be applied only to black and white images.

The Hough transform main idea is to convert equations of all lines in the image from Cartesian coordinates into polar coordinates. Then for each point from an input image we construct a curve given by the following formula:

$$\rho = x\cos(\theta) + y\sin(\theta) \qquad \text{(E.5)}$$

Points from the Cartesian coordinate system correspond to sinusoidal curves in the Hough space (ρ,θ). Furthermore, a line passing through any two points in Cartesian coordinates is the crossing point for the corresponding two curves in the Hough space (see, e.g., Fig. E.4). This makes it easy to identify points lying on a straight line in an input image. In the Hough space it is enough to identify points, where sinusoidal curves intersect and find the corresponding to these crossing curves points from an input image.

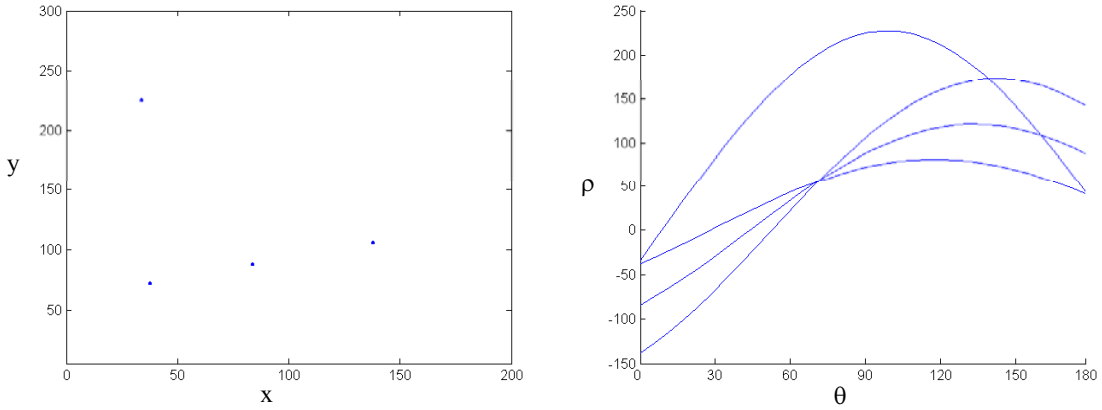


Fig. E.4 Points In Cartesian Space And Corresponding Curves In Hough Space.

VI Rough Measure

In this section, we discuss more some properties and examples of application of rough measures introduced briefly in chapter 3.

Example E.1 The ρ in a rough membership function (rmf) can be interpreted as in (E.6).

$$\mu_y^{B,\delta}(X)=\frac{\rho\left(X\cap[y]_B^{\delta}\right)}{\rho\left([y]_B^{\delta}\right)}=\frac{\int_{X\cap[y]_B^{\delta}} 1\ dx}{\int_{[y]_B^{\delta}} 1\ dx} \qquad \text{(E.6)}$$

Proposition E.2 The rmf $\mu_y^{B,\delta}$ in (E.6) is additive on U.

Proof. Let $X,\ Y \subset U$, and $X \cap Y = \varnothing$.

$$\mu_y^{B,\delta}(X \cup Y) = \frac{\int_{(X \cup Y) \cap [y]_B^\delta} 1\, dx}{\int_{[y]_B^\delta} 1\, dx} = \frac{\int_{(X \cap [y]_B^\delta) \cup (Y \cap [y]_B^\delta)} 1\, dx}{\int_{[y]_B^\delta} 1\, dx} = \frac{\int_{(X \cap [y]_B^\delta)} 1\, dx + \int_{(Y \cap [y]_B^\delta)} 1\, dx}{\int_{[y]_B^\delta} 1\, dx} = \frac{\int_{(X \cap [y]_B^\delta)} 1\, dx}{\int_{[y]_B^\delta} 1\, dx} + \frac{\int_{(Y \cap [y]_B^\delta)} 1\, dx}{\int_{[y]_B^\delta} 1\, dx}$$

$$= \mu_y^{B,\delta}(X) + \mu_y^{B,\delta}(Y) \qquad \blacksquare$$

It is also the case that $\mu_y^{B,\delta}$ is a non-negative set function. Hence, by definition of a measure (i.e., a measure is a non-negative, additive set function [4]), $\mu_y^{B,\delta}$ is a measure. This fact has significance for more advanced work on measurement systems used to inspect power system equipment such as transmission lines, wood towers, and other structures. That is, $\mu_y^{B,\delta}$ provides the basis for an extension of the Lebesgue integral (called rough integral), which has been shown to provide a form of ordered weighted average [5] useful in a variety of applications. The rough integral has been used in combination with the d-mesh and convex hulls in classifying sensor signals [6] and in signal analysis [7].

Example E.2 Sample Sensor Signal Measurements

The integral in ((E.6) is evaluated relative to subintervals over which sensors $a \in B$ are defined (i.e., subintervals wherein sensor signal measurements have been recorded). Consider, for example, a sampling of a real-valued signal in Fig. E.5 from a sensor.

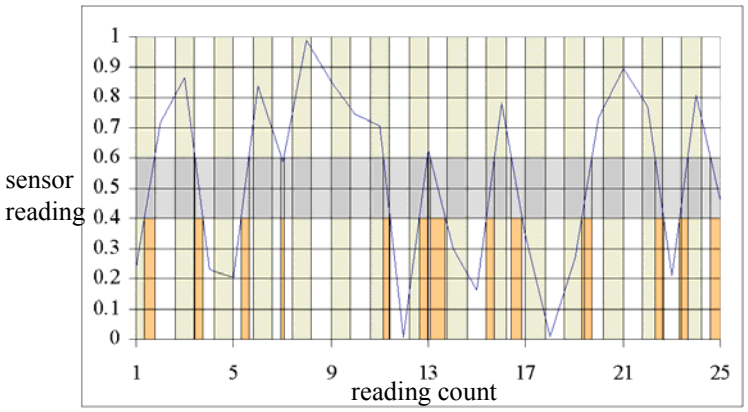


Fig. E.5 Sampling Of Real-Valued Signal.

From a 24 second signal we are able to measure every odd 0.8 sec. period. Our target value is y=0.5 with tolerance δ=0.1. Therefore,

$$X = \bigcup_{k=0}^{14} [1 + 1.6k, 1.8 + 1.6k] =$$

[1, 1.8] ∪ [2.6, 3.4] ∪ [4.2, 5.0] ∪ [5.8, 6.6] ∪ [7.4, 8.2] ∪ [9, 9.8] ∪
[10.6, 11.4] ∪ [12.2, 13] ∪ [13.8, 14.6] ∪ [15.4, 16.2] ∪ [17.0, 17.8] ∪
[18.6, 19.4] ∪ [20.2, 21] ∪ [21.8, 22.6] ∪ [23.4, 24.2]

and then construct $[y]_B^{0.1}\big|_{y=0.5}$ as follows.

$[y]_B^{0.1}\big|_{y=0.5}$ = [1.32, 1.8] ∪ [3.42, 3.72] ∪ [5.33, 5.68] ∪ [6.96, 7.01] ∪
[11.22,11.36] ∪ [12.66, 12.97] ∪ [13.11, 13.69] ∪ [15.4, 15.73] ∪
[16.44,16.84] ∪ [19.31,19.65] ∪ [22.33,22.62] ∪ [23.32,23.74] ∪ [24.56, 25]

Hence,

$X \cap [y]_B^{0.1}\big|_{y=0.5}$ = [1.32, 1.8] ∪ [11.22,11.36] ∪ [12.66, 12.97] ∪
[15.4, 15.73] ∪ [19.31, 19.4] ∪ [22.33, 22.6] ∪ [23.4, 23.74]

Then the rough membership function value can be computed as follows.

$$\mu_{0.5}^{B,0.1}(X) = \frac{\int_{X \cap [y]_B^{0.1}\big|_{y=0.5}} 1\, dx}{\int_{[y]_B^{0.1}\big|_{y=0.5}} 1\, dx} =$$

$$\frac{1.8 - 1.32 + 11.36 - 11.22 + \ldots + 23.74 - 23.4}{1.8 - 1.32 + 3.72 - 3.42 + \ldots + 25 - 24.56} = \frac{1.96}{4.43} = 0.44$$

A rough membership set function provides a classification measure inasmuch as it tests the degree of overlap between the set X in $\wp(U)$ and equivalence class $[y]_B^\delta$. A form of rough membership set function for non-empty, finite sets was introduced in [3].

Definition E.5 Let $u \in U$. A non-negative and additive set function $\rho_u \ \wp(X) \to [0, \infty)$ defined by $\rho_y(Y) = \rho'(Y \cap [y]_B^\delta)$ for $Y \in \wp(X)$, and $y = a(u)$, where $\rho'. \ \wp(X) \to [0, \infty)$ is called a *rough measure* relative to $U/\mathrm{Ing}_{A,\delta}(B)$ and u on the δ-indistinguishability space $(X, \wp(X), U/\mathrm{Ing}_{A,\delta}(B))$.

Proposition E.1 The rmf $\mu_y^{B,\delta}$ is a measure of X in U relative to $U/\mathrm{Ing}_{A,\delta}(B)$.

Bibliography

[1] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Prentice Hall, Upper Saddle River, New Jersey 07458, 2002.

[2] http://www.du.edu/~jcalvert/math/catenary.htm

[3] J.F. Peters, S. Ramanna, M. Borkowski, A. Skowron, Z. Suraj, Sensor, filter and fusion models with rough Petri nets, *Fundamenta Informaticae*, vol. 34, 2001, 1-19.

[4] P.R. Halmos, Measure Theory. London: D. Van Nostrand Co., Inc., 1950.

[5] Z. Pawlak, J.F. Peters, A. Skowron, Z. Suraj, S. Ramanna, M. Borkowski, Rough measures: Theory and Applications. In: S. Hirano, M. Inuiguchi, S. Tsumoto

(Eds.), Bulletin of the International Rough Set Society, vol. 5, no. 1 / 2, 2001, 177-184.

[6] J.F. Peters, T.C. Ahn, M. Borkowski, V. Degtyaryov, S. Ramanna, Line-crawling robot navigation: A rough neurocomputing approach. In: D. Maravall, D. Zhou (Eds.), Fusion of Soft Computing and Hard Computing Techniques for Autonomous Robotic Systems. Studies in Fuzziness and Soft Computing, J. Kacprzyk (Ed.). Berlin: Physica-Verlag, 2002 [to appear].

[7] M. Borkowski, Signal analysis using rough integrals. In: J. Alpigini, J.F. Peters, A. Skowron, N. Zhong (Eds.), Advances in Rough Sets and Soft Computing, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag, 2002, 218-225.

[8] P.V.C. Hough, Methods and means of recognizing complex patterns, U.S. Patent 3,069,654, 1962.

Appendix C (TLIM System Algorithms in C++)

This appendix contains C++ codes for algorithms used in the project. In order to simplify code, some intuitive sub-function's names were used:

allocMatrix(a,b) - allocates memory for matrix of dimensions a by b;

sortMatrix(m,c) - sorts matrix m, by its c column;

isIn(p,STR,t3,t2=null,t1=null) – checks if point p is enclosed by a figure made from points: STR[0] to STR[t3] and STR[t2] and STR[t1]; straightforward implementation checks the sum of angles fro point p to given points

```
class point2D{
public:
        double OWT;
        double angle;};
class point3D : public point2D{
public:
        double SF;};
```

I Nearest Neighbourhood

```
double distance(point3D q, point2D p)
{       return (q.OWT-p.OWT)^2+(q.angle-p.angle)^2;}

point2D nearestNeighbourhood(point3D *TR, int lenTR, point2D p)
{
double minDist = distance(TR[0],p);
int minPoint = 0;
for (int i=1; i<lenTR; i++)
{       double d = distance(TR[i],p);
        if (d < minDist)
```

```
        {       minDist = d;minPoint =  i;}
}

return TR[minPoint];
}
```

II Gouraud shading

```
double Gouraud(point3D TR[3], point2D p)
{
// find the first line coefficents
double A1 = TR[0].angle-p.angle;
double B1 = -p.OWT-TR[0].OWT;
double C1 = -p.OWT*A1-p.angle*B1;

// find the second line coefficients
double A2 = TR[2].angle-TR[1].angle;
double B2 = TR[1].OWT-TR[2].OWT;
double C2 = -TR[1].OWT*A2-TR[1].angle*B2;

// find crossing of these two lines
double D = A1*B2-A2*B1;
if (!D)
        // Outside triangle! Return Nearest Neighbourhood value
else
   double xc = (B1*C2-B2*C1)/D;
   double yc = (C1*A2-C2*A1)/D;
end

// find value for the crossing point
double I = sqrt(pow(TR[1].OWT-TR[2].OWT,2)+pow(TR[1].angle-TR[2].angle),2));
```

```
double alpha = sqrt(pow(TR[1].OWT-xc,2)+pow(TR[1].angle-yc,2));
double beta = sqrt(pow(TR[2].OWT-xc,2)+pow(TR[2].angle-yc,2));

if fabs(I-alpha-beta) > 1e-10
        // Outside triangle! Return Nearest Neighbourhood value

double C1 = (alpha*TR[2].SF+beta*TR[1].SF)/I;

// find value for point p
I1 = sqrt(pow(TR[0].OWT-xc,2)+pow(TR[0].angle-yc),2));
alpha = sqrt(pow(TR[0].OWT-p.OWT,2)+pow(TR[0].angle-p.angle,2));
beta = sqrt(pow(p.OWT-xc,2)+pow(p.angle-yc,2));

if fabs(I-alpha-beta) > 1e-10
        // Outside triangle! Return Nearest Neighbourhood value

return (alpha*C1+beta*TR[0].SF)/I;
}
```

III Partitioning Into Triangles

```
double *****partitionIntoTriangles(point3D *TR, int lenTR)
{
// make a table of all lines
int ll = lenTR*(lenTR-1)/2;
double ***tofl = allocMatrix(3,ll);
int k = 1;
for (int i=0; i<LenTR-1; i++)
{       double tr1 = TR[i].OWT, tr2 = TR[i].angle;
        for (int j=i+1;  j<ltr;  j++)
        {       tofl[0,k] = i;  tofl[1,k] = j;
```

```
                tofl[2,k] = (tr1-TR[j].OWT)^2+(tr2-TR[j].angle)^2];k=k+1;}
}

// sort them in ascending order
tofl = sortMatrix(tofl,3);

double *****ntr = allocMatrix(5,ll);
int lntr = 0;
for (i=0; i<ll; i++)
{       double x1 = TR[tofl[0,i]].OWT, y1 = TR[tofl[0,i]].angle;
        double x2 = TR[tofl[1,i]].OWT; y2 = TR[tofl[1,i]].angle;
        double A = y2-y1, B = x1-x2, C = -x1*A-y1*B;

        for (int j=0; j<lntr; j++)
        {       double tr11=TR[ntr[0,j]].OWT, tr21=TR[ntr[0,j]].angle;
                double tr12=TR[ntr[1,j]].OWT, tr22=TR[ntr[1,j]].angle;

                if ((x1 == tr11 && y1 == tr21)  ||  (x1 == tr12 && y1 == tr22) ||
                    (x2 == tr11 && y2 == tr21)  ||  (x2 == tr12 && y2 == tr22))
                        continue;

                // find crossing of lines
                double D = A*ntr[3,j]-ntr[2,j]*B, xc, yc;
                if (!D)
                        continue;    // parallel
                else
                {       xc = (B*ntr[4,j] - ntr[3,j]*C) / D;
                        yc = (C*ntr[2,j] - ntr[4,j]*A) / D;}

                // check one vector
                double I = sqrt((x1-x2)^2 + (y1-y2)^2);
```

```
                double alpha = sqrt((x1-xc)^2 + (y1-yc)^2);
                double beta = sqrt((x2-xc)^2 + (y2-yc)^2);

                if (I < alpha+beta-1e-12)
                        continue;

                // check the other
                I = sqrt((tr11-tr12)^2 + (tr21-tr22)^2);
                alpha = sqrt((tr11-xc)^2 + (tr21-yc)^2);
                beta = sqrt((tr12-xc)^2 + (tr22-yc)^2);

                if (I < alpha+beta-1e-12)
                        continue;

                A = 0;B = 0;break;
        }

        if (A!=0 || B!=0)
        {       ntr[0,++lntr] = tofl[0,i];
                ntr[1,lntr] =  tofl[1,i];
                ntr[2,lntr] =  A; ntr[3,lntr] = B; ntr[4,lntr] = C; }
}
return ntr;
}
```

IV Find Nearest Triangle

```
point3D[3] findNearestTriangle(point3D *TR, int lenTR, point2D p)
{
double *** allocMatrix(3,lenTR);
```

```
for (int i=0; i<LenTR; i++)
{       STR[0,i] = TR[i].OWT;  STR[1,i] = TR[i].angle;
        STR[2,k] = (p.x-TR[i].OWT)^2+(p.y-TR[i].angle)^2];}

// sort them in ascending order
STR = sortMatrix(STR,3);

int t1 = 2, t2 = 1, t3 = 0;
while (!isIn(p,STR,t1))
        if (++t1 == lenTR)
                return null; // outside any triangle!

if (t1 != 3)
        while (!isIn(p,STR,t2,t1))
                if (++t2 == t1-1)
                        break; // while

if (t2 != 2)
        while (!isIn(p,STR,t3,t2,t1))
                if (++t3 == t2-1)
                        break; // while

point3D R[3];
R[0] = TR[t1]; R[1] = TR[t2]; R[2] = TR[t3];

return R;
}
```

Appendix D (TLIM System Matlab Code)

This appendix contains all Matlab source code used for this project.

File ClipImages.m:

```
function varargout = ClipImages(varargin)
% ClipImages Application M-file for ClipImages.fig
%    FIG = ClipImages launch ClipImages GUI.
%    ClipImages('callback_name', ...) invoke the named callback.

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Use system color scheme for figure:
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

   % additional initialization
   colormap(gray(256));
   plotedit(handles.ClipImages,'on');

   if nargout > 0
                varargout{1} = fig;
        end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end

end

% --------------------------------------------------------------------
function varargout = edit1_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.edit1.

number=get(h,'String');

if length(number)==1
```

```
    filename='dsc0000';
elseif length(number)==2
    filename='dsc000';
elseif length(number)==3
    filename='dsc00';
elseif length(number)==4
    filename='dsc0';
else
    filename='dsc';end
filename=strcat(filename,number,'.jpg');
set(handles.edit2,'String',filename)

% --------------------------------------------------------------------
function varargout = process_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.process.

filename=get(handles.edit2,'String');

if strncmp(fliplr(filename),'wbm.',4)==1
    disp('reading .mbw file...')
    % attempt to load a file
    [img,vsize,hsize]=preprocessImage1(filename);

else
    disp('reading .jpg file...')

    % attempt to load a file
    global img;
    img=imread(filename,'jpg');

    % convert to black and white
    img=double(img(:,:,1))+double(img(:,:,2))+double(img(:,:,3))/3;

    % rescale colours to full range
    mn=min(min(img));mx=max(max(img));
    img=(img-mn)*256/(mx-mn);

    % increase input file number
    number=get(handles.edit1,'String');
    number=str2num(number)+1;
    set(handles.edit1,'String',int2str(number))
    ClipImages('edit1_Callback',handles.edit1, eventdata, handles, varargin);

end
image(img,'Parent',handles.mainWindow);
```

```
[ys,xs]=size(img);
set(handles.edit4,'String',strcat(int2str(xs),'x',strcat(int2str(ys))));

% --------------------------------------------------------------------
function varargout = check_size_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton2.

global img;

p=get(handles.mainWindow,'CameraPosition');
s=get(handles.mainWindow,'DataAspectRatio');
set(handles.edit4,'String',strcat(int2str(floor(p(1)+s(1)/2)-round(p(1)-
s(1)/2)+1),'x',int2str(floor(p(2)+s(2)/2)-round(p(2)-s(2)/2)+1)));

% --------------------------------------------------------------------
function varargout = edit5_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.edit5.

number=get(h,'String');

if length(number)==1
   filename='wire00';
elseif length(number)==2
   filename='wire0';
else
   filename='wire';end
filename=strcat(filename,number,'w.raw.mbw');

set(handles.edit6,'String',filename)

% --------------------------------------------------------------------
function varargout = export_file_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.export_file.
disp('attempting to export file...')

global img;

p=get(handles.mainWindow,'CameraPosition');
s=get(handles.mainWindow,'DataAspectRatio');
xs=floor(p(1)+s(1)/2)-round(p(1)-s(1)/2)+1;
ys=floor(p(2)+s(2)/2)-round(p(2)-s(2)/2)+1;
set(handles.edit4,'String',strcat(int2str(xs),'x',int2str(ys)));

filename=get(handles.edit6,'String');
f=fopen(filename,'w');
```

```
% save header
fwrite(f,[xs ys],'uint16');

% save file
fwrite(f,img(round(p(2)-s(2)/2):floor(p(2)+s(2)/2),round(p(1)-
s(1)/2):floor(p(1)+s(1)/2))','uint8');

fclose(f);

% increase output file number
number=get(handles.edit5,'String');
number=str2num(number)+1;
set(handles.edit5,'String',int2str(number))
ClipImages('edit5_Callback',handles.edit5, eventdata, handles, varargin);

% --------------------------------------------------------------------
function varargout = rotate_image_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.rotate_image.

global img;
img=img';

image(img,'Parent',handles.mainWindow);

% --------------------------------------------------------------------
function                                    varargout                                =
openImage(gimg,gfilename,gscale,oBorder,angle1,angle2,rThickness)

global img;
global rfilename;
global ggscale;
global goBorder;
global gangle1;
global gangle2;
global grThickness;

img=gimg;
rfilename=gfilename;

image(img);
set(findobj(gcf,'Tag','edit2'),'String',rfilename);
set(findobj(gcf,'Tag','returnToMW'),'Visible','on');

ggscale=gscale;goBorder=oBorder;gangle1=angle1;gangle2=angle2;grThickness=rThick
ness;
% --------------------------------------------------------------------
```

```
function varargout = returnToMW_Callback(h, eventdata, handles, varargin)

global img;
global rfilename;
global ggscale;
global goBorder;
global gangle1;
global gangle2;
global grThickness;

p=get(handles.mainWindow,'CameraPosition');
s=get(handles.mainWindow,'DataAspectRatio');
xs=floor(p(1)+s(1)/2)-round(p(1)-s(1)/2)+1;
ys=floor(p(2)+s(2)/2)-round(p(2)-s(2)/2)+1;

% open window
gmain
% open file
gmain('openImage',img(round(p(2)-s(2)/2):floor(p(2)+s(2)/2),round(p(1)-
s(1)/2):floor(p(1)+s(1)/2)),rfilename,ggscale,goBorder,gangle1,gangle2,grThickness);
% close this window
close ClipImages
```

File gmain.m:

```
function varargout = gmain(varargin)
% GMAIN Application M-file for gmain.fig
%    FIG = GMAIN launch gmain GUI.
%    GMAIN('callback_name', ...) invoke the named callback.

if nargin == 0  % LAUNCH GUI

        fig = openfig(mfilename,'reuse');

        % Use system color scheme for figure:
        set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

   % additional initialization
   colormap(gray(256));
   global gscale;
   gscale=1;

   % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);

   if nargout > 0
```

```
        varargout{1} = fig;
    end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

    try
        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
    catch
        disp(lasterr);
    end

end

% --------------------------------------------------------------------
function varargout = loadFile_Callback(h, eventdata, handles, varargin)

global gimg;
global rimg;
global gvsize;
global ghsize;
global gfilename;

gfilename=get(handles.edit2,'String');

if strncmp(fliplr(gfilename),'wbm.',4)==1
   disp('reading .mbw file...')
   % attempt to load a file
   [gimg,gvsize,ghsize]=preprocessImage1(gfilename);

else
   disp('reading .jpg file...')

   % attempt to load a file
   gimg=imread(gfilename,'jpg');

   % convert to black and white
   gimg=double(gimg(:,:,1))+double(gimg(:,:,2))+double(gimg(:,:,3))/3;

   % rescale colours to full range
   mn=min(min(gimg));mx=max(max(gimg));
   gimg=(gimg-mn)*256/(mx-mn);

   [gvsize,ghsize]=size(gimg);
end

image(gimg,'Parent',handles.mainWindow);
```

```
rimg=gimg;

% --------------------------------------------------------------------
function varargout = findAngle_Callback(h, eventdata, handles, varargin)

global gscale;
global gimg;
global gvsize;
global ghsize;

set(handles.editAngle,'String','working...');drawnow
angle=gdrst1(gimg,gvsize,ghsize,gscale);
set(handles.editAngle,'String',num2str(angle));
set(handles.rotateAngle,'String',num2str(angle));

[rgimg,vsize,hsize]=rotateImg1(gimg,angle,gscale,gvsize,ghsize);
image(rgimg,'Parent',handles.mainWindow);

% --------------------------------------------------------------------
function varargout = scale_Callback(h, eventdata, handles, varargin)

global gscale;
gscale=str2num(get(h,'String'));

% --------------------------------------------------------------------
function varargout = outerBorder_Callback(h, eventdata, handles, varargin)

global gscale;
global gimg;

set(handles.showOuterBorder,'String','working...');drawnow
[gw,angle,gf]=ggetBorders(gimg,str2num(get(handles.editAngle,'String')),gscale,1,handle
s);
set(handles.showOuterBorder,'String',num2str(gw));

% --------------------------------------------------------------------
function varargout = clipImage_Callback(h, eventdata, handles, varargin)

global gimg;
global gfilename;
global gscale;

% open window
ClipImages
% open file in a window
```

```
ClipImages('openImage',gimg,gfilename,num2str(gscale),get(handles.showOuterBorder,'
String'),get(handles.editAngle,'String'),get(handles.rotateAngle,'String'),get(handles.wire
Th,'String'));
% close this window
close gmain

% --------------------------------------------------------------------
function varargout = openImage(img,filename,scale,oBorder,angle1,angle2,rThickness)

global gimg;
global gfilename;
global gvsize;
global ghsize;
global gscale;

gimg=img;
gfilename=filename;
[gvsize,ghsize]=size(gimg);
gscale=str2num(scale);

image(gimg);
set(findobj(gcf,'Tag','edit2'),'String',gfilename)
set(findobj(gcf,'Tag','scale'),'String',scale);
set(findobj(gcf,'Tag','showOuterBorder'),'String',oBorder);
set(findobj(gcf,'Tag','editAngle'),'String',angle1);
set(findobj(gcf,'Tag','rotateAngle'),'String',angle2);
set(findobj(gcf,'Tag','wireTh'),'String',rThickness);

% --------------------------------------------------------------------
function varargout = wireTh_Callback(h, eventdata, handles, varargin)

global gwireThickness;

gwireThickness=str2num(get(handles.wireTh,'String'));

% --------------------------------------------------------------------
function varargout = loadSystem_Callback(h, eventdata, handles, varargin)

global gscale;
global gtr;
global gcorrect;
global gangles;
global gntr;
global gnsamples;

% build system
```

```
load(strcat('tr',int2str(gscale),'.mat'),'tr');
gnsamples=length(tr);
gtr=tr(:,1:gnsamples);

%gntr=trnet(gtr(1:2,:));
load(strcat('ntr',int2str(gscale),'.mat'),'ntr');
gntr=ntr;

% load correct values
load 'correct.mat' correct
gcorrect=correct;

% load angles
load(strcat('angs',int2str(gscale),'.mat'),'angs');
gangs=angs;

disp('system loaded...')
% --------------------------------------------------------------------
function varargout = rotateImage_Callback(h, eventdata, handles, varargin)

global gimg;
global gscale;
global gvsize;
global ghsize;
global rimg;
global rvsize;
global rhsize;

angle=get(handles.rotateAngle,'String');

[rimg,rvsize,rhsize]=rotateImg1(gimg,str2num(angle),gscale,gvsize,ghsize);
image(rimg,'Parent',handles.mainWindow);

% --------------------------------------------------------------------
function varargout = drawLines_Callback(h, eventdata, handles, varargin)

global gtr;
global gntr;
global gscale;

l=findobj(gcf,'Tag','lineOne');
if ~isempty(l)
   delete(l);end
l=findobj(gcf,'Tag','lineTwo');
if ~isempty(l)
   delete(l);end
```

```
l=findobj(gcf,'Tag','lineThree');
if ~isempty(l)
   delete(l);end

xl=get(handles.mainWindow,'Xlim');
[x,y1]=ginput(1);
line([xl(1) xl(2)],[y1 y1],'Tag','lineOne','Color','g');

[x,y2]=ginput(1);
line([xl(1) xl(2)],[y2 y2],'Tag','lineTwo','Color','g');

set(handles.realTh,'String','working...');drawnow
d=abs(y1-y2);
width=str2num(get(handles.showOuterBorder,'String'));
alpha=str2num(get(handles.wireTh,'String'))/width;
angle=abs(str2num(get(handles.editAngle,'String')))*0.6;

switch get(handles.listbox,'ListboxTop')
case 1, l=d;
case 2, l=d*alpha/mean(gtr(3,:))/gscale;
case 3, l=d*alpha/nn(8.3333*width,angle,gtr)/gscale;
case 4, l=d*alpha/appg(8.3333*width,angle,gtr,gntr)/gscale;
case 5, l=d*alpha/appgnt(8.3333*width,angle,gtr)/gscale;
end

set(handles.realTh,'String',num2str(l));

% --------------------------------------------------------------------
function varargout = listbox_Callback(h, eventdata, handles, varargin)

global gtr;
global gntr;
global gscale;
global gimg;
global gvsize;

l1=findobj(gcf,'Tag','lineOne');
l2=findobj(gcf,'Tag','lineTwo');
l3=findobj(gcf,'Tag','lineThree');

if (~isempty(l1) & ~isempty(l2))% | ~isempty(l3)
   set(handles.realTh,'String','working...');drawnow

   if ~isempty(l3)
      gvsize
      x=get(l3,'XData');
```

```
      size(gimg(:,floor(x(1)/gscale)))
      [le,re]=gdete1(gimg(:,floor(x(1)/gscale)),gvsize,gscale);
      d=re-le;
   else
      y1=get(l1,'YData');
      y2=get(l2,'YData');
      d=abs(y1-y2);
   end

   width=str2num(get(handles.showOuterBorder,'String'));
   alpha=str2num(get(handles.wireTh,'String'))/width;
   angle=abs(str2num(get(handles.editAngle,'String')))*0.6;

   switch get(handles.listbox,'ListboxTop')
   case 1, l=d;
   case 2, l=d*alpha/mean(gtr(3,:))/gscale;
   case 3, l=d*alpha/nn(8.3333*width,angle,gtr)/gscale;
   case 4, l=d*alpha/appg(8.3333*width,angle,gtr,gntr)/gscale;
   case 5, l=d*alpha/appgnt(8.3333*width,angle,gtr)/gscale;
   end

   set(handles.realTh,'String',num2str(l));
else
   set(handles.realTh,'String','--');end


% --------------------------------------------------------------------
function varargout = drawLine_Callback(h, eventdata, handles, varargin)

global gtr;
global gntr;
global gscale;
global rimg;
global rvsize;

l=findobj(gcf,'Tag','lineOne');
if ~isempty(l)
   delete(l);end
l=findobj(gcf,'Tag','lineTwo');
if ~isempty(l)
   delete(l);end
l=findobj(gcf,'Tag','lineThree');
if ~isempty(l)
   delete(l);end

y=get(handles.mainWindow,'Ylim');
```

```
[x,yy]=ginput(1);
line([x x],[y(1) y(2)],'Tag','lineThree','Color','r');

[le,re]=gdete1(rimg(:,floor(x/gscale)),rvsize,gscale);
d=re-le;
width=str2num(get(handles.showOuterBorder,'String'));
alpha=str2num(get(handles.wireTh,'String'))/width;
angle=abs(str2num(get(handles.editAngle,'String')))*0.6;

switch get(handles.listbox,'ListboxTop')
case 1, l=d;
case 2, l=d*alpha/mean(gtr(3,:))/gscale;
case 3, l=d*alpha/nn(8.3333*width,angle,gtr)/gscale;
case 4, l=d*alpha/appg(8.3333*width,angle,gtr,gntr)/gscale;
case 5, l=d*alpha/appgnt(8.3333*width,angle,gtr)/gscale;
end

set(handles.realTh,'String',num2str(l));
```

File gdete1.m:

```
% Maciej Borkowski 2002
% detect edges

function [le,re]=gdete(slice,svsize,scale)

% edge threshold
eth=1*scale;

% find maximas
[i,m]=min(slice);mn=mean(slice);

% go left
for i=m:-1:2
   if slice(i)-slice(i-1)>=0 & slice(i)>mn
      break;end
end
% check if this is real edge
if slice(i)-slice(i+scale)>eth
   le=i;
else
   der1=slice(i+1:m)-slice(i:m-1);
   for j=1:length(der1)
      if abs(der1(j))>1/scale
         break;end
   end
   le=i+j-1;
```

```
end

% go right
for i=m:svsize-1
   if slice(i+1)-slice(i)<=0 & slice(i)>mn
      break;end
end
% check if this is real edge
if slice(i)-slice(i-scale)>eth
   re=i;
else
   der1=slice(i:-1:m+1)-slice(i-1:-1:m);
   for j=1:length(der1)
      if abs(der1(j))>1/scale
         break;end
   end
   re=i-j+1;
end
```

File ggetBorders.m:

```
% Maciej Borkowski 2002
% finds borders (third edition)
% usage [w,angle,f]=getBorders(img,angle,scale,width,handles)

function [w,angle,f]=ggetBorders(img,angle,scale,width,handles)

[vsize,hsize]=size(img);

% rescale & rotate image (use linear interpolation)
[rsimg,shsize,svsize]=rotateImg1max(img,angle,scale,vsize,hsize);

aslice=mean(rsimg');

% get rid of borders
rsimg=rsimg(:,2:(end-1));shsize=shsize-2;
s=0;ls=0;
for ys=1:svsize
   if rsimg(ys,1)~=255 | rsimg(ys,shsize)~=255
      break;
   else
      i=find(rsimg(ys,:)~=255);
      s=s+sum(rsimg(ys,i));ls=ls+length(i);end
end
for ye=svsize:-1:1
   if rsimg(ye,1)~=255 | rsimg(ye,shsize)~=255
      break;
```

```
   else
      i=find(rsimg(ye,:)~=255);
      s=s+sum(rsimg(ye,i));ls=ls+length(i);end
end
rsimg=rsimg(ys:ye,:);svsize=ye-ys+1;
if ls~=0
   s=s/ls;[i,j]=find(rsimg==255);
   rsimg(i,j)=s;end

slice=mean(rsimg');

[le,re]=gdete(slice,svsize,scale,handles);

f=scale*width/(re-le);
w=(re-le)/scale;
```

File gdrst1.m:

```
% Maciej Borkowski 2002
% main file (third edition)
% usage angle=gdrst1(img,vsize,hsize,scale)

function angle=gdrst1(img,vsize,hsize,scale)

% find initial angle
angle=iniAngle(img,scale,vsize,hsize);
a1=angle;

flmc=FLMcriterion1(img,angle,scale,vsize,hsize);
hflmc=flmc;oangle=angle;

if flmc>0   angle=angle-1;
else        angle=angle+1;end

while abs(flmc)>0.01

   oflmc=flmc;
   flmc=FLMcriterion1(img,angle,scale,vsize,hsize);

   for i=1:length(hflmc)
      if abs(hflmc-flmc)<0.00001
         angle=(angle+oangle)/2;break;end
   end
   if i==length(hflmc)
      hflmc=[hflmc flmc];
   else
      continue;end
```

```
   if flmc*oflmc<0
      a=angle+flmc/(-flmc+oflmc)*(angle-oangle);
      oangle=angle;angle=a;
   else
      a=flmc/(oflmc-flmc)*(angle-oangle)+angle;
      oangle=angle;angle=a;
   end

   disp([angle flmc]);
end

disp([angle angle-a1])
```

File iniAngle.m:

```
% Maciej Borkowski 2002
% finds coarse angle

function angle=iniAngle(img,scale,vsize,hsize);

% rescale & rotate image (use linear interpolation)
[rsimg,shsize,svsize]=rotateImg1(img,0,scale,vsize,hsize);

% apply delta-mesh
[dmesh,dmx,dmy]=f_dMesh1(rsimg,32,32,shsize,svsize);

% find angle
[c,i]=min(dmesh);

angle=getAngle(1:dmx,i*vsize/hsize);
```

File getAngle.m:

```
% Maciej Borkowski 2002
% finds angle for points using linear least squares fit

function ang=getAngle(x,y);

c=cov(x,y);

ang=c(1,2)/c(1,1);
b=mean(y)-ang*mean(x);

ang=atan(ang)*180/pi;
```

File FLMCriterion1.m:

```
% Maciej Borkowski 2002
```

```
% finds first-last-match crtiterion (second edition)

function flmc=FLMcriterion1(img,angle,scale,vsize,hsize);

% rescale & rotate image (use linear interpolation)
[rsimg,shsize,svsize]=rotateImg1(img,angle,scale,vsize,hsize);

% apply delta-mesh
[dmesh,dmx,dmy]=f_dMesh1(rsimg,24,12,shsize,svsize);

% select twice below average
me=mean(reshape(dmesh,1,dmx*dmy));s=0;l=0;
for x=1:dmx
   i=find(dmesh(:,x)<me);
   % remove outsiders
   if length(i)>1
      ro=[];
      if i(2)-i(1)~=1
         ro=[ro 1];end
      if i(length(i))-i(length(i)-1)~=1
         ro=[ro length(i)];end
      for j=2:length(i)-1
         if i(j)-i(j-1)~=1 & i(j+1)-i(j)~=1
            ro=[ro j];end
      end
      i(ro)=[];
   end

   s=s+sum(dmesh(i,x));
   l=l+length(i);
end

s=s/l;
dm=zeros(dmy,dmx);
for x=1:dmx
   i=find(dmesh(:,x)>s);
   dm(i,x)=255;
end
%add more
for y=1:(dmy-1)
   i=find(dm(y+1,:)==0);
   dm(y,i)=0;
   i=find(dm(dmy-y,:)==0);
   dm(dmy-y+1,i)=0;
end
```

```
% find flmc
fl=floor(dmx/2-1);
rt=zeros(dmy,fl);
for x=2:fl
   for y=1:dmy
      if dm(y,x)~=255 | dm(y,dmx-x+1)~=255
         rt(y,x-1)=dmesh(y,dmx-x+1)-dmesh(y,x);end
   end
end

% average flmc
avrt=sum(rt')/fl;

il=1;ir=dmy;
for i=1:dmy
   if avrt(i)~=0
      il=i;
      break;end
end
for i=dmy:-1:1
   if avrt(i)~=0
      ir=i;
      break;end
end

im=il+(ir-il)/2;
if floor(im)==im
   adp=1;
else adp=0;end;
ls=sum(avrt(il:floor(im)));
rs=sum(avrt(ceil(im):ir));

flmc=rs-ls;
```

File rotateImage1.m:

```
% Maciej Borkowski 2002
% rotate & rescale image using linear interpolation (second edition)

function [rsimg,shsize,svsize]=rotateImg1(img,angle,scale,vsize,hsize)

cosa=cos(-pi*angle/180);sina=sin(-pi*angle/180);
svsize=scale*vsize;shsize=scale*hsize;
rsimg=max(max(img))*ones(svsize,shsize);
hsize2=hsize/2;vsize2=vsize/2;
for y=scale:svsize-scale
        for x=scale:shsize-scale
```

```
                xxx=x/scale;yyy=y/scale;
                xx=cosa*(xxx-hsize2)+sina*(yyy-vsize2)+hsize2;
                yy=cosa*(yyy-vsize2)-sina*(xxx-hsize2)+vsize2;
                xs=floor(xx);ys=floor(yy);
                if xs>0 & xs<hsize & ys>0 & ys<vsize
                        xf=xx-xs;yf=yy-ys;
                        rsimg(y,x)=img(ys,xs)+(img(ys,xs+1)-
                        img(ys,xs))*xf+(img(ys+1,xs)-
                        img(ys,xs))*yf+(img(ys,xs)+img(ys+1,xs+1)-img(ys+1,xs)-
                        img(ys,xs+1))*xf*yf;
                end
        end
end

rsimg=rsimg(scale:svsize-scale,scale:shsize-scale);
shsize=shsize-2*scale+1;svsize=svsize-2*scale+1;
```

File f_dMesh1.m:

```
% Maciej Borkowski 2002
% calculates d-mesh (second edition)

function [dmesh,dmx,dmy]=f_dMesh(rsimg,dx,dy,shsize,svsize)

deltax=shsize/dx;if deltax<1 deltax=1;end
deltay=svsize/dy;if deltay<1 deltay=1;end
dmx=ceil(shsize/deltax);dmy=ceil(svsize/deltay);
dmesh=zeros(dmy,dmx);
dmc=zeros(dmy,dmx);

for y=1:svsize
        for x=1:shsize
                xs=ceil(x/deltax);ys=ceil(y/deltay);
                dmesh(ys,xs)=dmesh(ys,xs)+rsimg(y,x);
                dmc(ys,xs)=dmc(ys,xs)+1;
        end
end
dmesh=dmesh./dmc;
```

File preprocessImage1.m:

```
% Maciej Borkowski 2002
% read file and detect wire (second edition)

function [img,vsize,hsize]=preprocessImage1(filename)

f = fopen(filename,'r');
hsize=fread(f,1)+256*fread(f,1);vsize=fread(f,1)+256*fread(f,1);
```

```
[img,l]=fread(f,[hsize,vsize]);fclose(f);
if l~=vsize*hsize
        'Unexpected end of file!'
        return
end
img=img';
```

Appendix E (Research Project Digital Camera Specifications)

SONY CD Mavica MVC-CD300 Digital Still Camera Specifications:

System

Image device

8.93 mm (1/1.8 type) color CCD

Lens

f=7 - 21mm (9/32 - 27/32 inches)

F=2.0 - 2.5

Exposure control

Automatic exposure, Shutter speed priority, Aperture priority, Manual exposure

White balance

Automatic, Indoor, Outdoor, One-push

Data system

Movie	MPEG1
Still	JPEG, GIF (in TEXT mode, Clip Motion), TIFF Audio with still image: MPEG1 (Monaural)

Recording medium

8 cm CD-R/CD-RW

Recommended flash recording distance (ISO is set to AUDIO)

0.3 to 3 m (11 7/8 inches to 8 1/3 feet)

Drive

Read	Maximum x8
Write	x4
Readout	Noncontact optical readout (using semiconductor laser)

Laser

Wavelength	777 to 787 nm

NA	0.5
Maximum output	23 mW
Emission duration	600ns

Input and Output connector

A/V OUT (MONO) (Monaural)

Minijack Video	1 Vp-p, 75 Ω unbalanced, sync negative
Audio	327 mV (at a 47 kΩ load)
Output impedance	2.2 kΩ

ACC jack

Mini-minijack (Ø 2.5 mm)

USB jack

mini-B

LCD screen

LCD panel

TFT (Thin Film Transistor active matrix) drive

LCD size

2.5 type

Total number of dots

123 200 (560x220) dots

General

Application

Sony battery pack NP-FM50 (supplied)

Power requirements

7.2 V

Power consumption (During shooting with the LCD backlight turn on)

3.5 W

Operating temperature

0°C to 40°C (32°F to 104°F)

Storage temperature

-20°C to +60°C (-4°F to =140°F)

Dimensions (Approx.)

143x92x94 mm (5 ¾ x 3 5/8 x 3 ¾ inches) (w/h/d)

Mass (Approx.)

650 g (1 lb 7 oz) (including NP-FM50 battery pack, disc and lens cap, etc.)

Built-in microphone

Electret condenser microphone

Built-in speaker

Dynamic speaker

NP-FM50 battery pack

Battery type

Lithium ion

Maximum output voltage

DC 8.4 V

Mean output voltage

DC 7.2 V

Capacity

8.5 Wh (1180 mAh)

Operating temperature

0°C to 40°C (32°F to 104°F)

Dimensions (Approx.)

38.2x20.5x55.6 mm (1 9/16 x13/16 x 2 ¼ inches) (w/h/d)

Mass (Approx.)

76 g (3 oz)