工學碩士 學位論文

Estimation Method for Container Surface Damage Using the Parameters of Capsize-Gaussian Function

Supervisor Hwan-Seong Kim

2006 年 2 月

韓國海洋大學校 大學院

物流시스템 工學科

Graduate School of Korea Maritime University

Department of Logistics

Tran Ngoc Hoang Son

工學碩士 學位論文

Estimation Method for Container Surface Damage Using the Parameters of Capsize-Gaussian Function

Supervisor

Prof. Hwan-Seong Kim

By

Tran Ngoc Hoang Son

A thesis submitted in partial fulfillment of the requirements

for the degree of

Master of Engineering

In the Department of Logistics

Graduate School of Korea Maritime University

February, 2006

Acknowledgement

I would like to express my deep gratitude to my supervisor Professor Kim Hwan-Seong, for his invaluable help, encouragement, and patience during the master course in Engineering of Logistics. His insight, vision, suggestions, and special behavior of brisk style in work has impressed to me and helped me to archive success in research and study.

I am grateful to Professor Nam Ki-Chan, Logistics System and Transportation Planning; Professor Kwak Kyu-Seok, Port Logistics and Social Engineering; Professor Shin Jae-Yeong, Management Science and Logistics Information System.. They all helped me immeasurably in detailing comments and improving the presentation of material in different parts of the thesis.

Special thanks to all the members of Logistic Automation Lab (LAL) and Intelligent Robot Lab (IRL) for giving me a helpful, active and comfortable environment which conducts my scientific pursuits: Dr. Nguyen Tuong Long (LAL), MS. Bae Min-Ju (LAL), Mr. Nguyen Duy Anh (LAL), Dr. Kim Heon-Hui (IRL), Mr. Oh Ju-Heon (IRL), Mr. Ok Dong-Ju (IRL).

I closely express my thanks to Korean and Vietnamese friends for their friendliness, sharing, and confidence: Dr. Tran Thien Phuc, Dr. Nguyen Tan Tien, Dr. Chung Tan Lam, Mr Tran Thanh Ngon, MS. Nguyen Thi Thuy Duong, MS. Du My Le, Mr Nguyen Hai Linh, MS. Tran Thi Thanh Dao.

Last but by no means least, it is warmly that I would like to thank my parents and my younger brother for their constant support and encouragement.

Korea Maritime University, Pusan, Korea

December 28th, 2005

Tran Ngoc Hoang Son

Estimation Method for Container Surface Damage Using the Parameters of Capsize-Gaussian Function

Tran Ngoc Hoang Son

Department of Logistics

Graduate School of Korea Maritime University, 2006

Abstract

In this thesis, an estimating damage on external surface of container using Capsize-Gaussian-Function (be called CGF) is presented. Estimation of the damage size can be get directly from two parameters of CGF, these are the depth and the flexure, also the direction of damage. The performance of the present method has been illustrated using an image of damage container, which had been taken from Hanjin Port in Pusan, Korea, after using image processing techniques to do preprocessing of the image, especially, the main used technique is Canny edge detecting that is widely used in computer vision to locate sharp intensity and to find object boundaries in the image, then correlation between the edge image

from the preprocessing step and the CGF with three parameters (direction, depth, flexure), as a result, we get an image that perform damage information, and these parameters is an estimator directly to the damage.

Contents

List of Figures

Chapter 1: Introduction

1.1 Thesis overview

Estimating quality of containers is an important problem in a Port System. Automation of this process is one of needs of improvement and modernization Port Management Technology. This thesis presents a research about recognition and estimating the size of damage on external surface of container. The damage is alteration in the shape of dimensions of container as a result of the application or accident of stress to it. Information of the damage could be obtained by using camera or capture devices. Some of automatic applications are used to recognize and estimate the rate of damage. In order to detect and estimate the damage shape, first of all, the image could be preprocessed and detected edges using Canny edge detector or another edge detection methods, and second step, we need to extract interesting features which could describe the characteristics of problem. In chapter two, we have briefly described some of image processing procedures used in the preprocessing data, and addressed detail to Canny edge detector which is used to extract the boundary of the image. In chapter three, an feature extracting method

by using Capsize-Gaussian-Function in the directly estimating the damage will be introduced. Chapter four will introduce the correlation operators which be used to match between original image and sets of capsize-gaussian function, and the simple threshold method which be used to recognize deformation points. In the following chapter five, we will show the implementation and results of our research. In chapter six will be discussion and conclusion.

1.2 Introduction to image processing-base detection damage-container system

Figure 1.2.1 is a illustration an automatic checking container system. There are three cameras mounted on three side of container-checking gate to get image data around three sides of container when a container carrying truck goes in to the gate. These data will be sent to Central Monitoring, Diagnosing, and Controlling System. In there, the images will be processed and analyzed to make quantitative measurements about size and direction of deformation on container and use these to make a decision, such as controlling the switch gate to choose path (or accept or reject) for the truck going out.

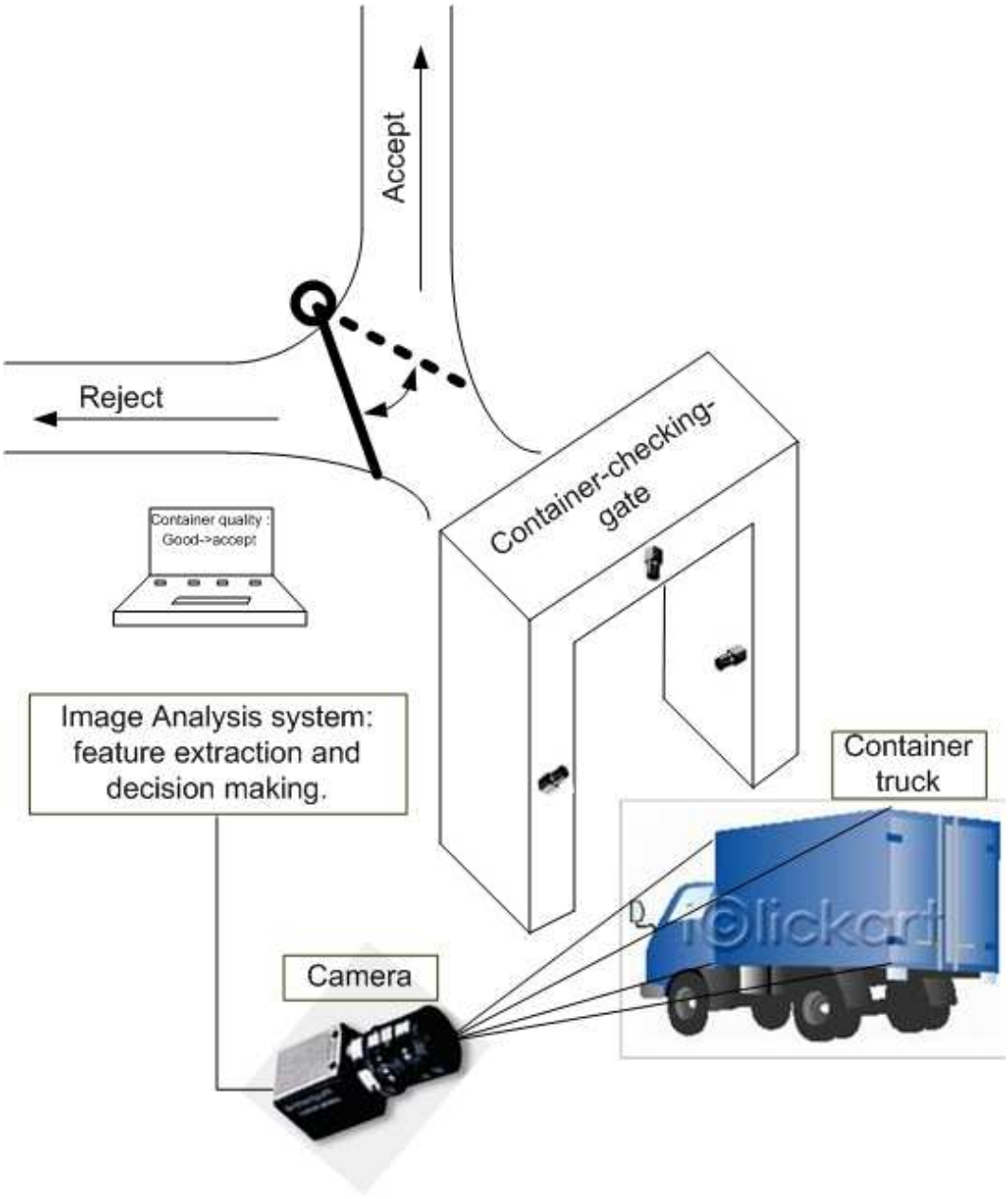


Fig. 1.2.1 Container checking and sorting at gate.

1.3 Implementation and limit of thesis

This thesis have researched about a method of detecting and estimating deformation or damage along the edges of container, that is a let of Capsize-Gaussian Function (CGF) in approximating of function of deformation at a point on the edge. From three parameters of CGF: direction, depth and flexure, we can get an image that performing damage information, and these parameters are an estimator directly to the damage.

Chapter 2: Image Processing

2.1 Introduction

The important of images in the development of human civilization is evident. Even in the ancient times images have played a significant role. Today, image technologies develop rapidly, and we may expect that in future the important of images will increase progressively. Apart from spiritual or artistic connotations, the significance of images is especially found in information systems. Images are used to improve or assist human visual perception. Thermograph, for instance, helps us to perceive heat sources that are hard to observe with the naked eye. Magnetic resonance imaging (MRI) devices can sense materials in the interior of 3D objects which may be used for patient screening and monitoring or for detection of tumors or other disease in patients. Images acquired by satellites are useful in tracking of earth resources, geographic mapping, prediction of agricultural crops, urban growth, and weather, flood and fire control, and many other environmental applications. In visual information systems images are the physical carriers of information. Images can easily be

brought in a form suitable for further processing. Therefore, another application of images is in automated information systems , i.e. image based measurement systems. A rudimentary example is an infrared detector equipped with only one sensor. These detectors are used, for instance, to open doors automatically as soon as a person wants to enter a building. Its output is a symbol with two possible states: one indicating that there is a heat source in its vicinity; the other that there isn’t. An example of a more complex image based measurement system that reads the images on three sides of container and measures parameters describing the qualities of container. Here, the output consists of command to accept or reject a container.

Observed image, which is included of informative edge of deformation shapes and another unwanted edges, should be decreasing uninteresting variations or suppressing noise and detecting the edges. These processes are called conditioning. There are many kinds of techniques used in this step, refer [8][9], such as noise cleaning, sharpening, edge detection, line detection. In section two, we have just presented Canny edge detector which is widely used in computer vision to locate sharp intensity changes and to find object boundaries in an image [1].

2.2 Image Processing, Computer Vision, and Applications of Image Based Measurement Systems

Image processing and computer vision are all processes that deal with images. The difference between these processes is in the representation of the input and output data. The inputs of image processing are images and the outputs of it are images. The inputs of computer vision are images and the outputs of it are descriptions of scene.

Our processing objects are digital images. A digital image $I(m,n)$ is described in a tow-dimensional discrete space which is derived from an analog image $I(x, y)$ in a tow-dimensional continuous space through a sampling process.

Digital image processing has a broad spectrum of applications, such as remote sensing via satellites and other spacecraft, image transmission and storage for business applications, medical processing, radar, sonar, and acoustic image processing, robotics, and automated inspection of industrial parts, or automated control bases on extracted characteristic information in an image.

Computer vision is the science that develops the theoretical and algorithmic basis by which useful information about the world can be automatically extracted and analyzed from an observed image, image set, or image sequence from computations made by special-purpose or general-purpose computers. Such information can be related to the recognition of a generic object, the there –dimensional description of an unknown object, the position and orientation of the observed object, or the measurement of any spatial property of

an object, such as the distance between two of its distinguished points or the diameter of a circular section. Applications of the technology range from vision-guided robot assembly to inspection tasks involving mensuration, verification that all parts are present, or determination that surfaces have no defects (example : container surface).

Image based measurement is a process that assigns meaningful numbers and symbols to objects in an imaged scene. This information may concern: a simple detection of an object (Is there a certain object in the scene), classification of objects (To what class does an object belong), parameter estimation (What is the size, position and orientation of an object), shape analysis (What is the shape and geometry of the objects), scene description (What are the objects in the scene, and how are they related to each other).

Image based measurement systems find wide applications in various kinds of scientific areas. But also in various industrial, medical and agricultural environments the applications are rapidly growing. The first scientific discipline that discovered the power of computer vision was astronomy. In the sixties, pictures of the moon were processed by computers. Initially, these techniques were developed to improve the quality of the images, i.e. restoration. Later, scientists found out that similar techniques could be applied to take over the manual, time consuming procedures needed to extract the desired information from these images, other scientific disciplines that use computer vision nowadays include: photogrammetry, particle physics, biology, medical science, geology and oceanology, science of material, industry, security. In the industrial and

agricultural area, the main application of image based measurement is quality control. In many production lines, for instance in the food industry, the quality of products must be guaranteed. Manual inspection is laborious an unreliable, and thus expensive. However, the quality control often involves a visual task. Therefore, in these branches, the application of computer vision is fruitful.

2.3 Recognition Methodology

Computer recognition and inspection of objects is, in general, a complex procedure requiring a variety of steps that successively transform the iconic data to recognition information. Handling unconstrained environments is often difficult for today's computer vision and recognition technology because the existing algorithms are specialized and do not develop one or more of the necessary transformation steps to a high enough degree. This thesis is that there are no shortcuts. A recognition methodology must pay substantial attention to each of the following of six steps: image formation, conditioning, labeling, grouping, extracting, and matching.

Image formation occurs when a sensor resisters radiation that has interacted with physical objects. The mathematical model of imaging has several different components: an image function is the fundamental abstraction of an image; An geometrical model describes how three dimensions are projected into two; A radio-metrical model shows how the imaging geometry, light sources, and reflectance properties of objects affect the light measurement at the sensor; A spatial frequency model describes how different spectral measurements are related

to image colors; A digitizing model describes the process of obtaining discrete samples.

Conditioning is based on a model that suggests that the observed image is composed of an information pattern modified by uninteresting variations that typically add to or multiply the informative pattern. Conditioning estimates the informative pattern on the basic of the observed image. Thus conditioning suppresses noise, which can be thought of as random un-patterned variations affecting all measurements. Conditioning can also perform background normalization by suppressing uninteresting systematic or patterned variations. Conditioning is typically applied uniformly and is context independent.

Labeling is based on a model that suggests that the informative pattern has structure as a spatial arrangement of events, each spatial event being a set of connected pixels. Labeling determines in what kinds of spatial events each pixel participates. For example, if the interesting spatial events of the informative pattern are events only of high-valued and low-valued pixels, then the thresholding operations include edge detection, corner finding, an identification of pixels that participate in various shape primitives.

The labeling operation labels pixels with the kinds of primitive spatial events in which the pixel participates. The grouping operation identifies the events by collecting together or identifying maximal connected sets of pixels participating in the same kind of event. If the labels are symbolic, then the grouping is really a connected components operation. If the labels are gray levels, then the grouping operation is what the vision literature call a segmentation. If the

labels are step edges, then the grouping operation constitutes edge linking, and so on. The grouping operation involves a change of logical data structure. The observed image, the conditioned image, and the labeled image are all digital image data structures. Depending on the implementation, the grouping operation can produce either an image data structure in which each pixel is given an index associated which the spatial event to which it belongs or a data structure that is a collection of sets. Each set corresponds to a spatial event and contains the pairs of (row, column) positions that participate in the event. In either case a change occurs in the logical data structure. The entities of interest before the grouping step are pixels. The entities of interest after the grouping step are sets of pixels.

The grouping operation determines the new set of entities. But after the grouping step the new entities are naked. The only thing they possess is their identity. The extracting operation computes for each group of pixels a list of its properties. Example properties might include its centroid, its area, its orientation, its spatial moments, its gray tone moments, its spatial-gray tone moments, it circumscribing circle, it inscribing circle, and so on. Other properties might depend on whether the group is considered a region or an arc. If the group is a region then number of holes might be a useful property. If the group is an arc, then average curvature might be a useful property. Extracting also can measure topological or spatial relationships between two or more groupings. For example, an extracting operation may make explicit that two groupings touch or are spatially close or that one grouping is above another.

After the completion of the extracting operation, the events occurring on the image have been identified and measured. But the events in and of themselves have no meaning. The meaning of the observed spatial events emerges when a perceptual organization has occurred such that a specific set of spatial events in the observed spatial organization clearly constitutes an imaged instance of some previously known object, such as a chair or the letter A. Once an object or set of objects parts has been recognized, then measurements such as the distance between two parts, the angle between two lines, or the area of an object part can be made and related to the allowed tolerance, for instance, in an inspection scenario. It is the matching operation that determines the interpretation of some related set of image events, associating these events with some given three-dimensional object or two-dimensional shape. The association determined by matching establishes a correspondence between each spatial event in the related set of events on the image with some spatial event on the three-dimensional object or two-dimensional shape. The association is one that in some sense best matches both the character of the spatial events and the stand in some spatial relationships. Thus, after matching, two primitive image events that stand in some spatial relationship will have associated with them two object events that stand in a similar relationship.

A wide variety of image operations are matching operations, the classic one is template matching, which is effective only if the variety of instances expected to be encountered is limited. For example, rotation and size variations

must be very small, the background must be nearly uniform. Random shape deformations must be minimal.

Simple shapes will correspond to primitive spatial event, and the property measurement form the primitive spatial event will often be adequate to permit recognition of the shape. In this case the matching operation amounts to matching the vector of propertied measured from the image spatial event with the vector of properties of a prototype representative. Such matching constitutes statistical pattern recognition. Complex shapes will correspond to a set of primitive spatial event as well as the spatial relationships between the events. In this case the matching amounts to determining a relational homomorphism with unary constraints established by the required matching of the property vectors of the observed image events with the property vectors of the prototype primitives. Such a matching is what constitutes structural pattern recognition.

2.4 Canny Edge Detector

Canny edge detector determined edges by an optimization process, refer to [1][12], to ensure three criteria is that low error rate, the edge points be well localized (minimize the actual edge), and have a only one response to a single edge. Based on these criteria, a typical implementation of the Canny edge detector follows steps below.

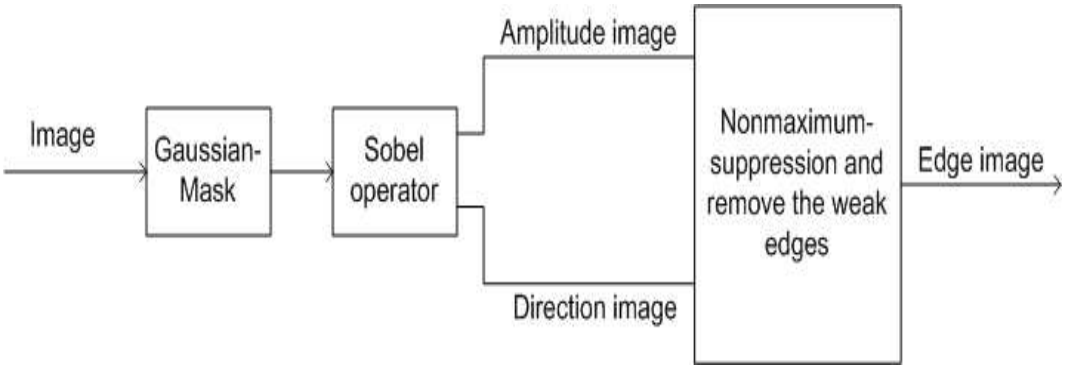


Fig. 2 Illustrating Canny edge detector.

1. Smooth the image to eliminate the noise and reduce desired image details by using an appropriate Gaussian mask (filter) convolute with the origin image. (Refer to **Fiter_Gaussian** function in the appendixes)

2. Determine gradient magnitude and gradient direction at each pixel from the smoothed image. (Refer to **Gradient_Sobel** function in the appendixes)

3. Non-maximum suppression is used to be applied to trace along the edge in the gradient direction and suppress any pixel value that is not considered to be an edge. If the gradient magnitude at a pixel is larger than those at its two neighbors in the gradient direction, mark the pixel as an edge. Otherwise, mark the pixel as the background. (Refer to **NonMaximum_Supression** function in the appendixes)

4. Remove the weak edges by hysterisis thresholding.

Chapter 3: Capsize-Gaussian Function

3.1 Define types of damage or deformation

Damage or deformation on container is caused by collision, so the shape of deformation can be figured in Fig. 3.1.1.

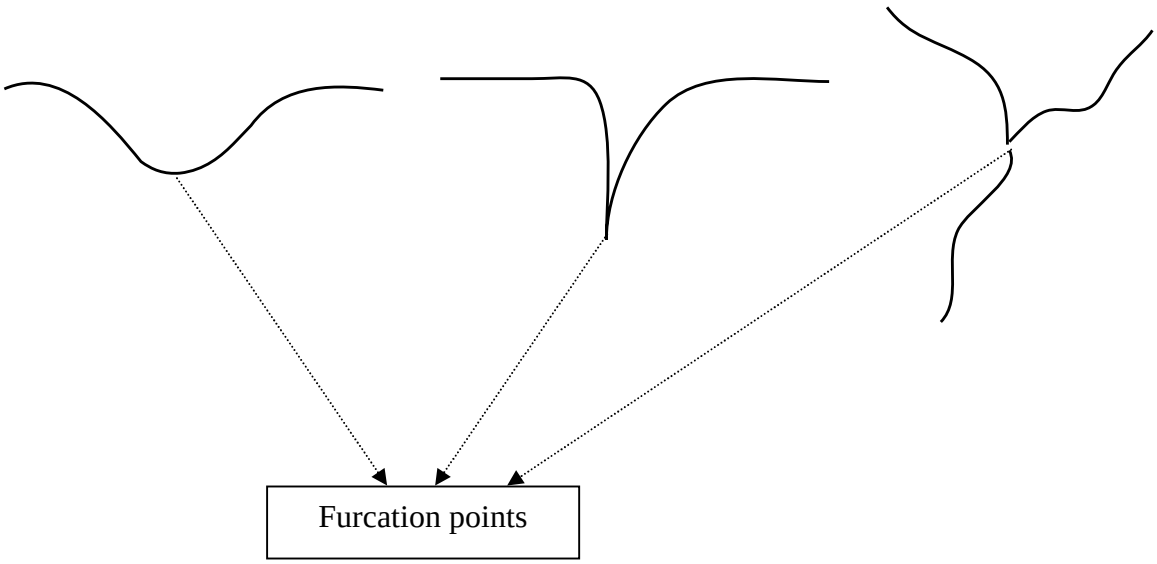


Fig. 3.1.1 Types of deformation shape

These deformation shapes have a general feature that is furcation point from which there are several branch starting.

From the shape of these, a ideal of using capsized Gaussian function to match with the deformation shape at furcation point for estimation the deformation was founded. Shape of the Capsize-Gaussian Function (CGF) will be introduced below.

3.2 Shape of CGF in recognition curves of deformation

First of all, a quote from beginning of chapter twelve in [11] makes me interesting and pleasure. That is: “One of the most interesting of the word is that it can be considered to be made up of patterns. A pattern is essentially an arrangement. It is characterized by the order of the elements of which it is made, rather than the intrinsic nature of these elements”, Norbert Wiener. So a deformation can be recognized by a function or a class of function which can describe the deformation shape. An ideal uses Capsize-Gaussian Function (CGF) (1) to match with embossment-lines from the deformation image. Fig. 3.2.1 illustrates the shape of CGF.

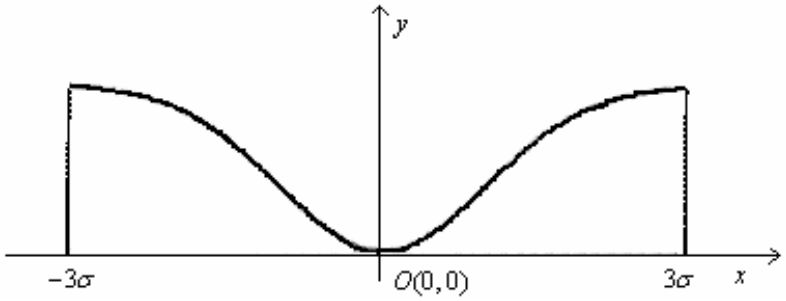


Fig. 3.2.1 Capsize-Gaussian Function

$$y(x) = a\left[1 - e^{\left(-\frac{x^2}{2\sigma^2}\right)}\right] \qquad (1)$$

$$|x| \le 3\sigma$$

The difference of depth and flexure level of deformation at a point can be measured through changing of two parameters a (depth) and σ (flexure), respectively, Fig. 3.2.2.

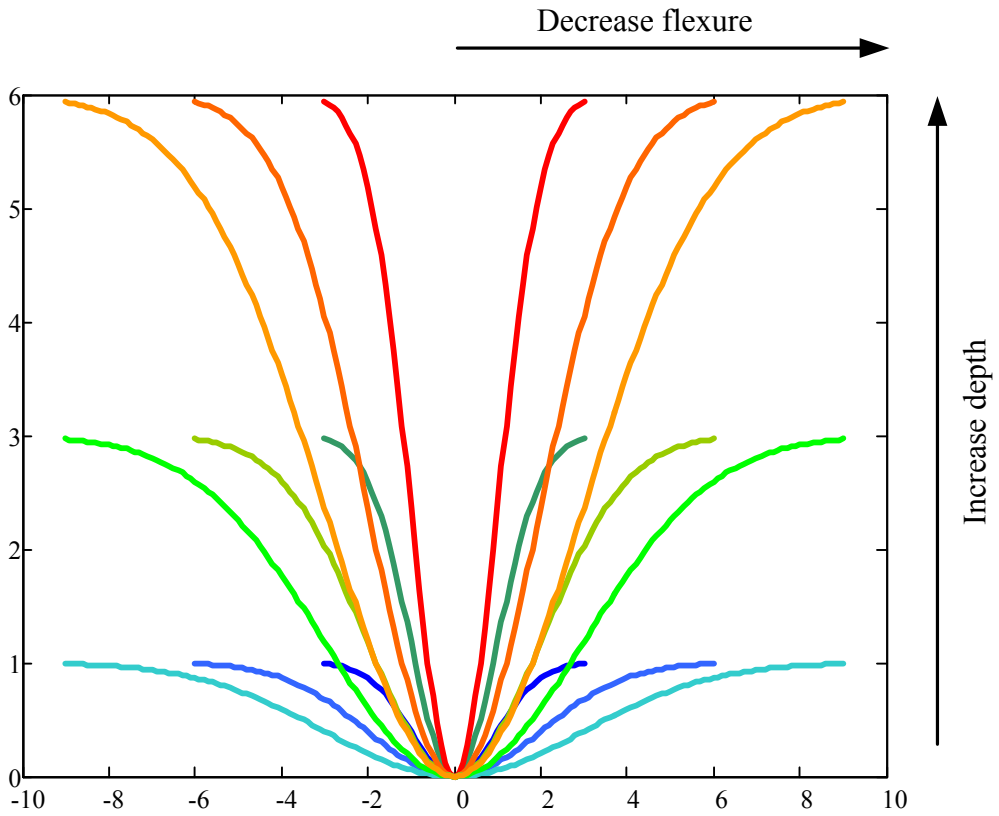


Fig. 3.2.2 Changing of depth and flexure levels of CGF

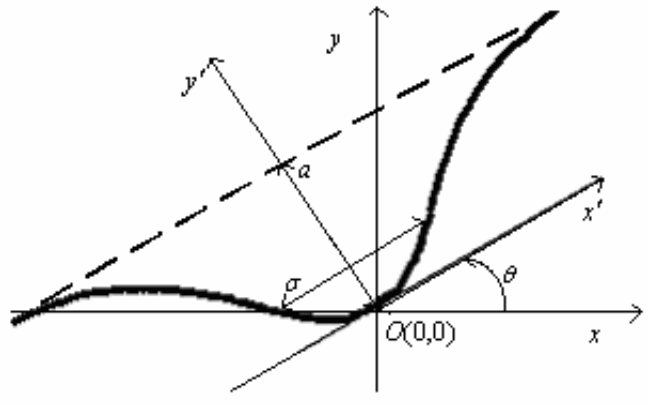


Fig. 3.2.3 Rotate CGF an angle θ about $O(0,0)$

But deformation has arbitrary shapes, so we need one more parameter θ for rotating CGF like (2). Fig.3.2.3 shows the CGF of rotating an angle θ about the origin O(0,0).

$$\Re(\theta,a,\sigma)=\left\{(x,y)\left|\begin{array}{l} y'=a\left[1-e^{\left(-\frac{x'^2}{2\sigma^2}\right)}\right] \\ \begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}x'\\y'\end{bmatrix} \end{array}\right.\right\} \tag{2}$$

$$W(x,y,\theta,a,\sigma)=\begin{cases}1, & \text{if } (x,y)\in \Re(\theta,a,\sigma)\\ 0, & \text{otherwise}\end{cases}$$

where $W(x,y,\theta,a,\sigma)$ is a window containing CGF at angle θ.

A function $f(x,y)$ of deformation shape can be analyzed as a linear combination of expansion functions of CGF windows

$$f(x,y)=\sum_i\sum_j\sum_k d_{i,j,k}W\left(x,y,\theta_i,a_j,\sigma_k\right) \tag{3}$$

where i,j,k are integer indexes of the finite sum, the $d_{i,j,k}$ are real-valued expansion coefficients or called correlation coefficients which those values will be estimated in chapter four. Fig. 3.2.4 illustrates vector integrated information of deformation at point (x,y).

The code listing in the appendixes of **Gauu_Draw** functions is used to determine a CGF with its three parameters of depth, flexure and direction.

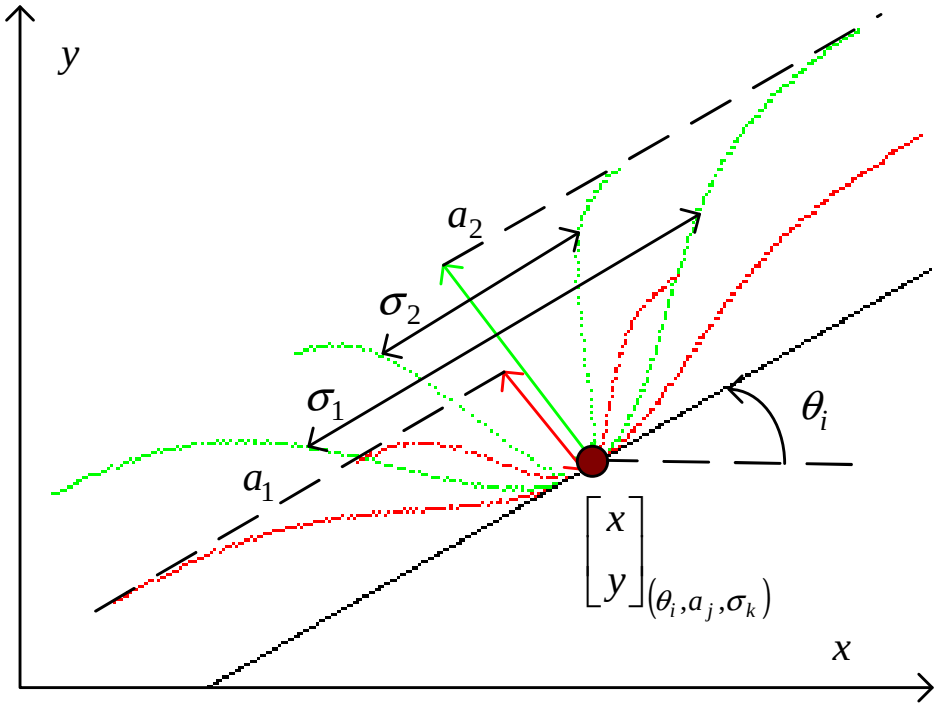


Fig. 3.2.4 Vector Integrated information of deformation at a point (x,y)

Chapter 4: Analysis tools

4.1 Matching by correlation

Recognition techniques based on matching represent each class by a prototype pattern vector. An unknown pattern is assigned to the class to which it is closest in terms of a predefined metric. The simplest approach is the minimum-distance classifier, which, as its name implied, computes the Euclidean distance between the unknown and each of the prototype vectors. It chooses the smallest distance to make a decision. Here. We just discuss an approach based on correlation, which can be formulated directly in terms of images and is quite intuitive.

Correlation coefficient of two functions $I(x, y)$, deformation image, and $W(x, y, \theta, a, \sigma)$, CGF window mask, is defined as both continuous and discrete case as :

$$d(m,n,\theta,a,\sigma)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}W(x+m,y+n,\theta,a,\sigma)I(x,y)dxdy \tag{4}$$

$$d(m,n,\theta,a,\sigma)=\sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty}W(x+m,y+n,\theta,a,\sigma)I(x,y) \tag{5}$$

Correlation function measures the relative matching of two functions for different shifts given by m, n. In matching, $I(x,y)$ is an image containing shapes of deformation. If we want to determine whether $I(x,y)$ contains a shape of deformation, we just do correlate between $I(x,y)$ and CGF window mask. Then, if there is a match, the correlation of the two functions will be maximum at the location where $W(x,y,\theta,a,\sigma)$ finds a correspondence in $I(x,y)$.

An example to illustrate the effect of correlation operation is shown in Fig. 4.1.1. An edge image with array of line form the Capsize-Gaussian-Function after rotate -90 degree will be correlate with a mask $W\left(x,y,-\frac{\pi}{2},a,\sigma\right)$, and the result will appear white points (called centers of shape deformation) in the horizontal line. As expected, we can see the highest value (white points) of the correlation function occurs at the point where the image is exactly on the top of the mask image.

The correlation function given in equation (5) has the disadvantage of being sensitive to changes in the amplitude of $W(x,y,\theta,a,\sigma)$ and $I(x,y)$. For example, doubling all values of $I(x,y)$ doubles the value of $d(x,y,\theta,a,\sigma)$. An

approach frequently used to overcome this difficulty is to perform matching via the correlation coefficient, which is defined as

$$\delta(m,n,\theta,a,\sigma)=\frac{\sum\sum\left[I(x,y)-\bar{I}(x,y)\right]\left[W(x+m,y+n,\theta,a,\sigma)-\overline{W}\right]}{\sqrt{\sum\sum\left[I(x,y)-\bar{I}(x,y)\right]^2\sum\sum\left[W(x+m,y+n,\theta,a,\sigma)-\overline{W}\right]^2}} \quad (6)$$

Where $\overline{W}$ is the average value of the pixels in W, $\bar{I}(x,y)$ is the average value of $I(x,y)$ in the region coincident with the current location of W, and the

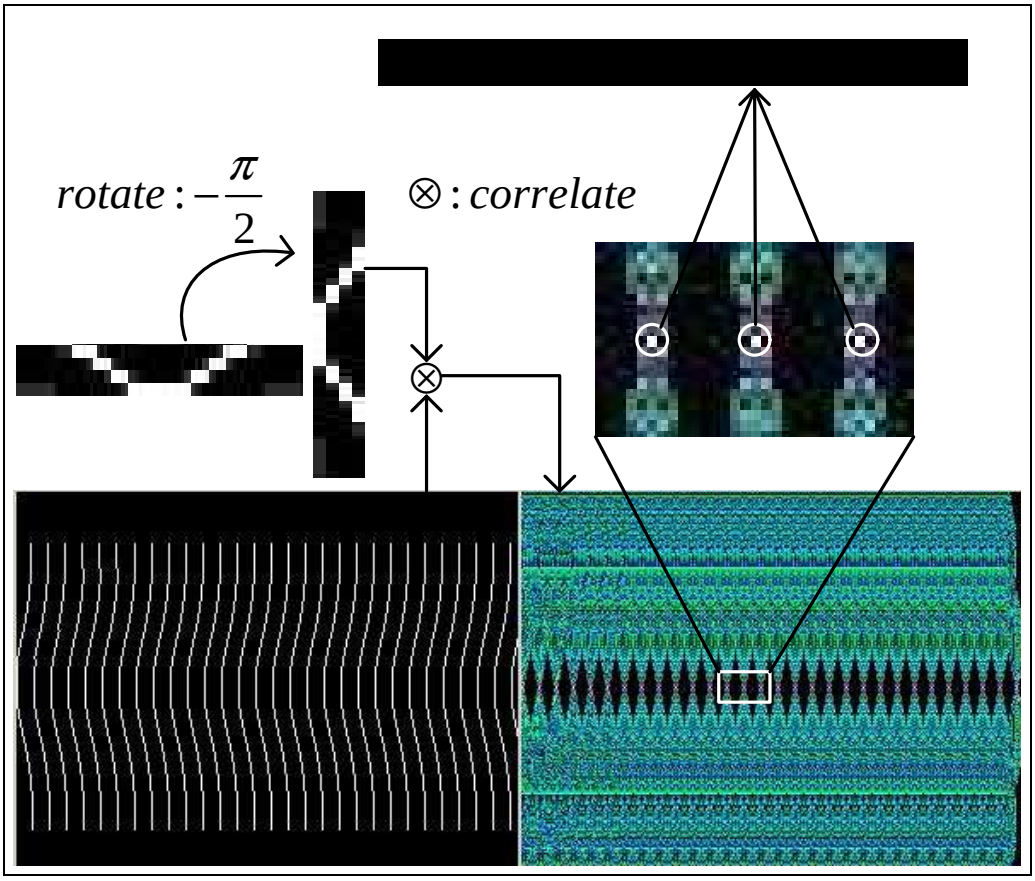


Fig. 4.1.1 Illustrate the method to detect points of high deformation

summations are taken over the coordinates common to both $I(x, y)$ and W the correlation coefficient $\delta(m, n, \theta, a, \sigma)$ is scaled in the range -1 to 1, independent of scale changes in the amplitude of $I(x, y)$ and W.

4.2 Recognition based on threshold method

After correlation two the functions, we want to remove the weak points where the matching is not clear, so we can reject these point by a threshold, like Fig.4.2.1.

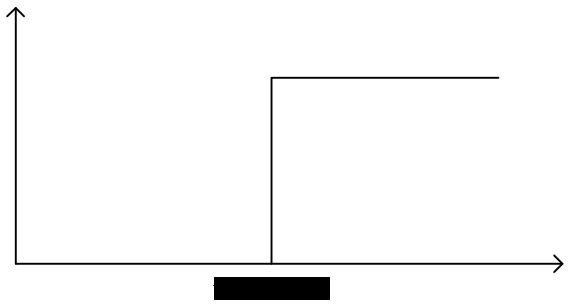


Fig. 4.2.1 Thresholding transformation

Chapter 5: Implementation and Results

5.1 Assign example problem and implementing tools

To reduce the time of computation of the integral correlation calculation, for the 294x220 origin image of Fig. 5.1.1, we have just chosen a few of elements of each parameters, but it has not lost the signification of our method. Values for each parameter are given, such as:

$$\theta = \left[-\frac{\pi}{2} \quad 0 \quad \frac{\pi}{2} \quad \pi \right]$$

$$a = [5 \quad 10 \quad 20]$$

$$\sigma = [7 \quad 15 \quad 28]$$

Fig. 5.1.1 Origin image

All the calculations and results have been programmed using Microsoft Visual C++ 6.0 compiler.

5.2 Results

First of all is result in edge image: Sobel amplitude image in Fig. 5.2.1, Sobel direction image in Fig. 5.2.2, Canny edge image in Fig. 5.2.3.

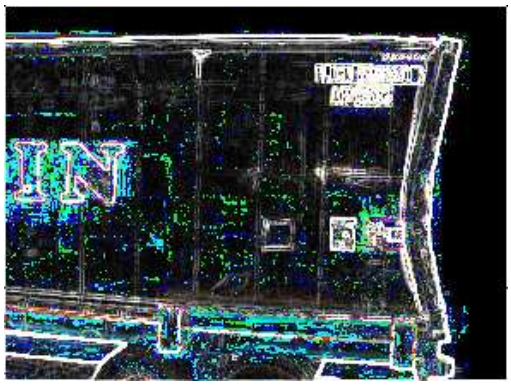


Fig. 5.2.1 Sobel amplitude image

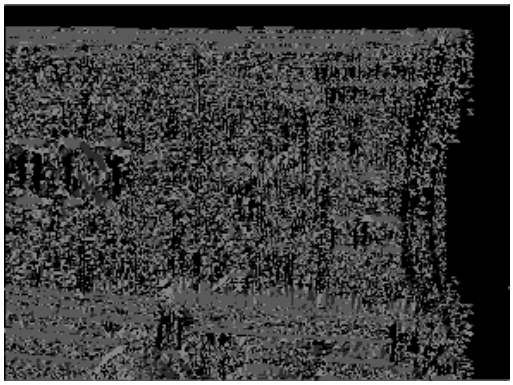


Fig. 5.2.2 Sobel direction Image

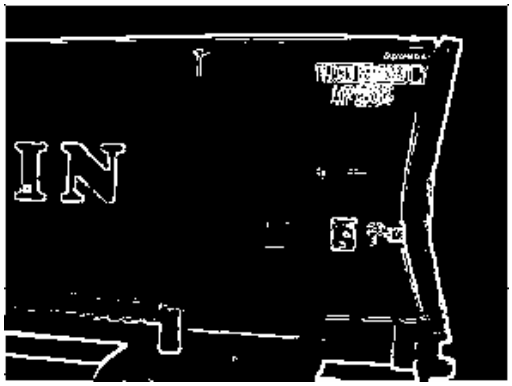
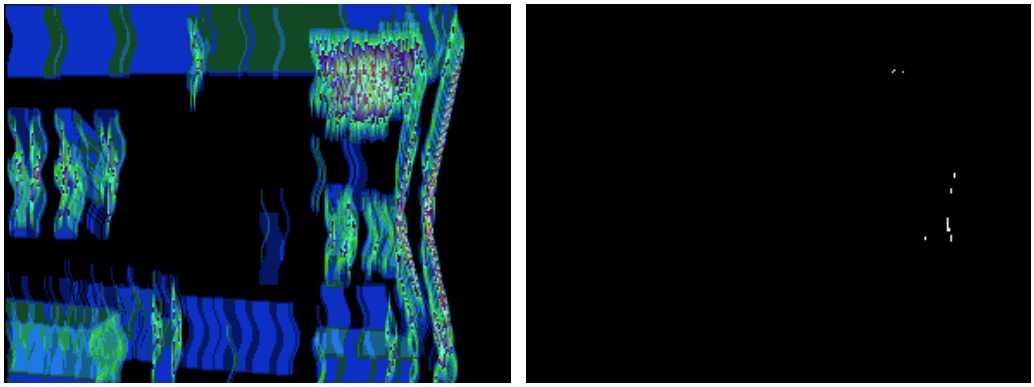


Fig 5.2.3 Canny edge image

Second results are list of images from correlating between origin image and CGF with given values of parameters above.

-pi/2

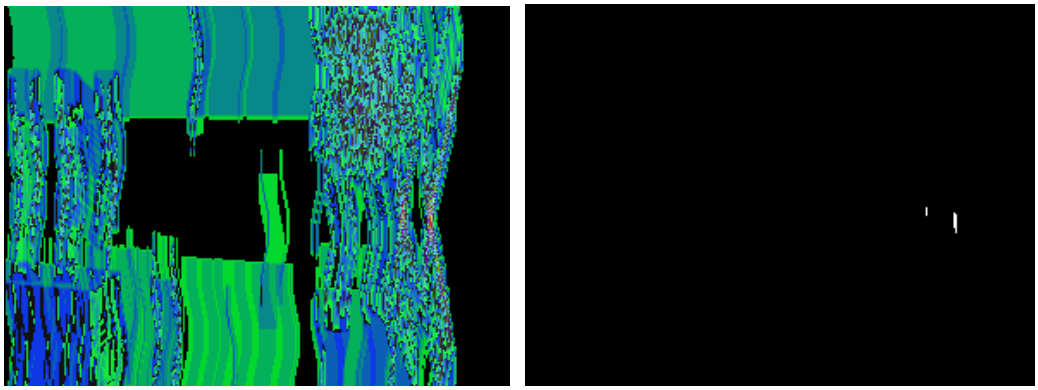


a=5, s=7 threshold image

Fig. 5.2.4.a Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 5, 7\right)$

-pi/2

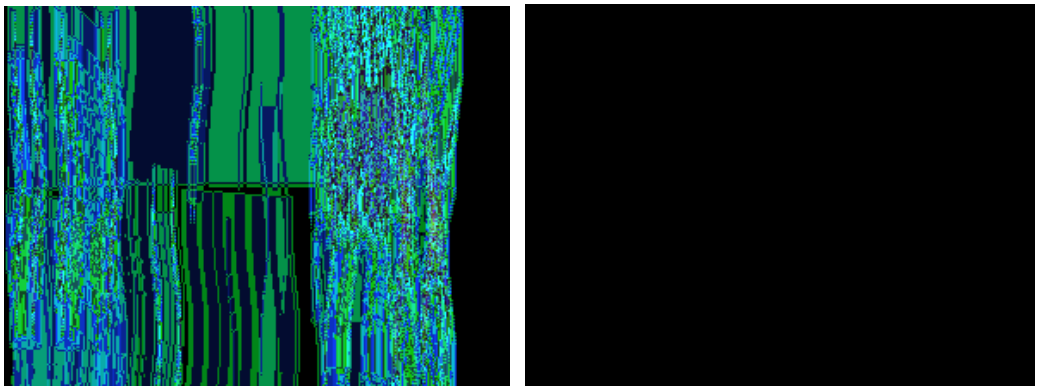


a=5, s=15 threshold image

Fig. 5.2.4.b Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 5, 15\right)$

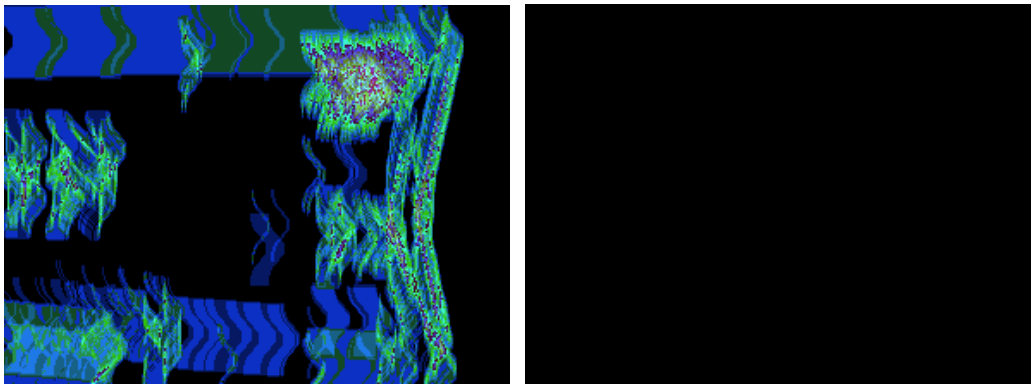
-pi/2



a=5, s=28 threshold image

Fig. 5.2.4.c Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 5, 28\right)$

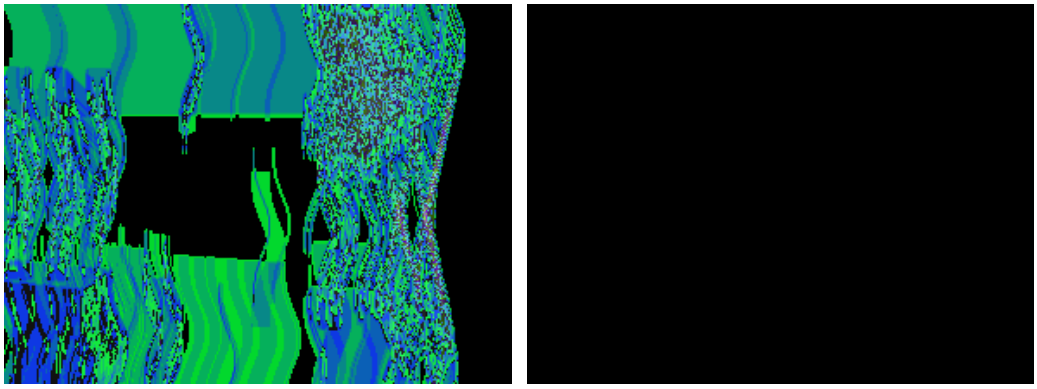
-pi/2



a=10, s=7 threshold image

Fig. 5.2.4.d Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 10, 7\right)$

-pi/2

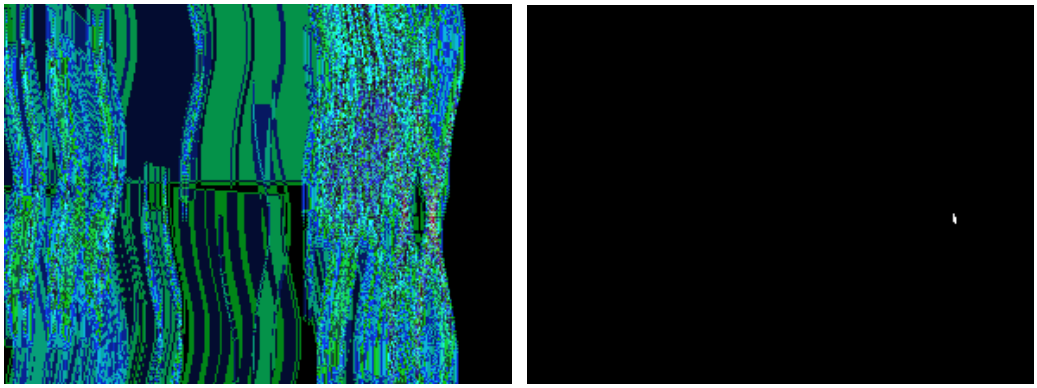


a=10, s=15 threshold image

Fig. 5.2.4.e Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 10, 15\right)$

-pi/2

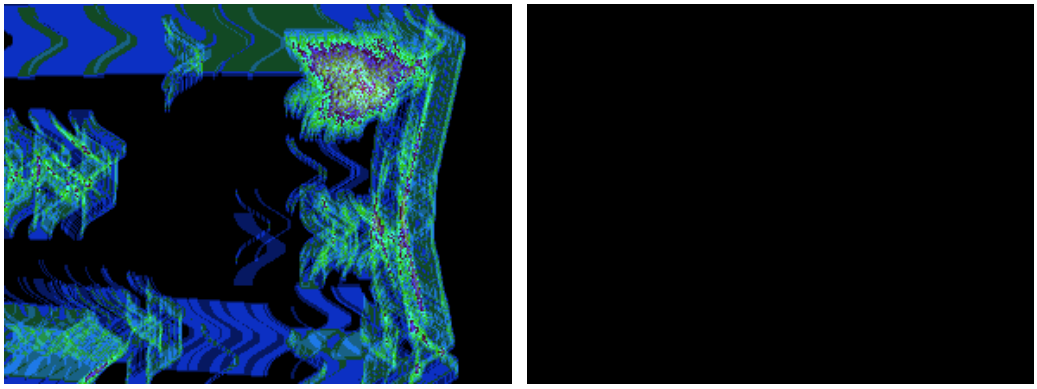


a=10, s=28 threshold image

Fig. 5.2.4.f Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 10, 28\right)$

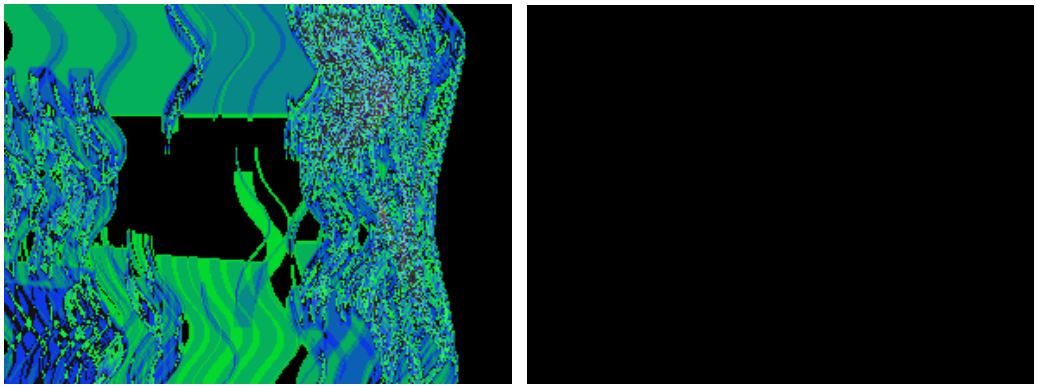
-pi/2



a=20, s=7 threshold image

Fig. 5.2.4.g Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 20, 7\right)$

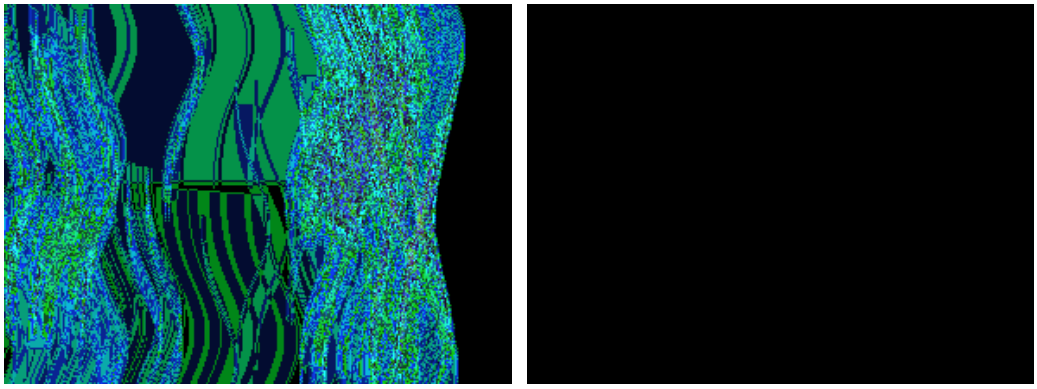
-pi/2



a=20, s=15 threshold image

Fig. 5.2.4.h Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 20, 15\right)$

-pi/2

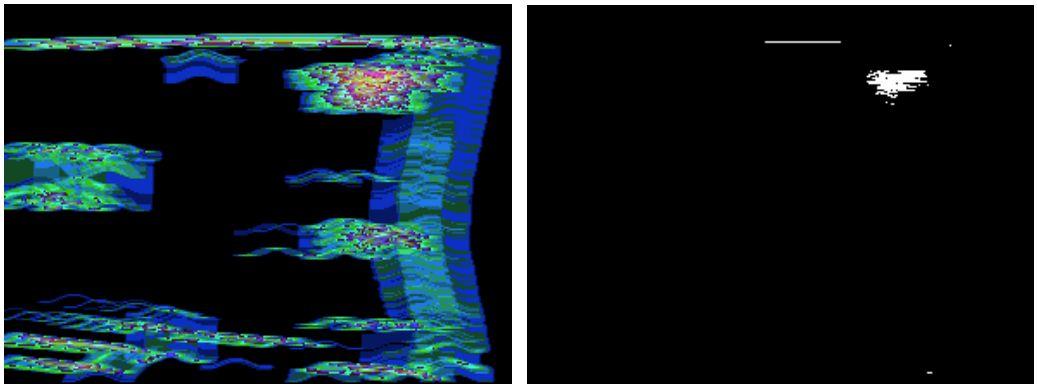


a=20, s=28 threshold image

Fig. 5.2.4.i Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 20, 28\right)$

Zero pi

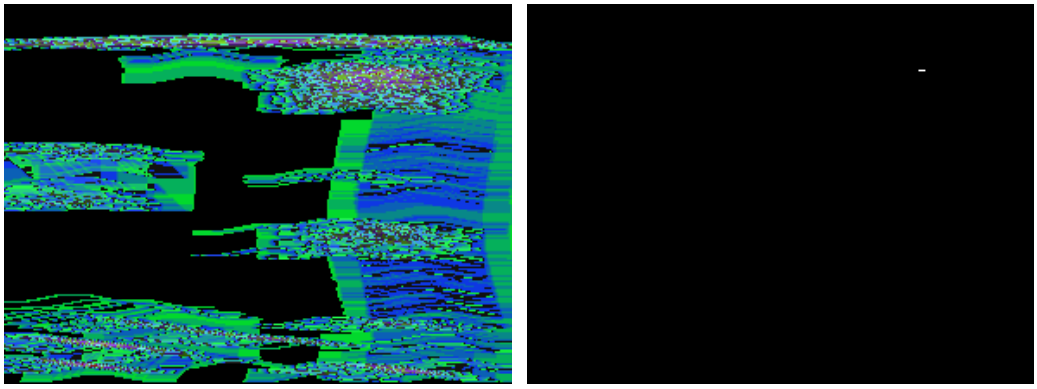


a=5, s=7 threshold image

Fig. 5.2.5.a Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = (0, 5, 7)$

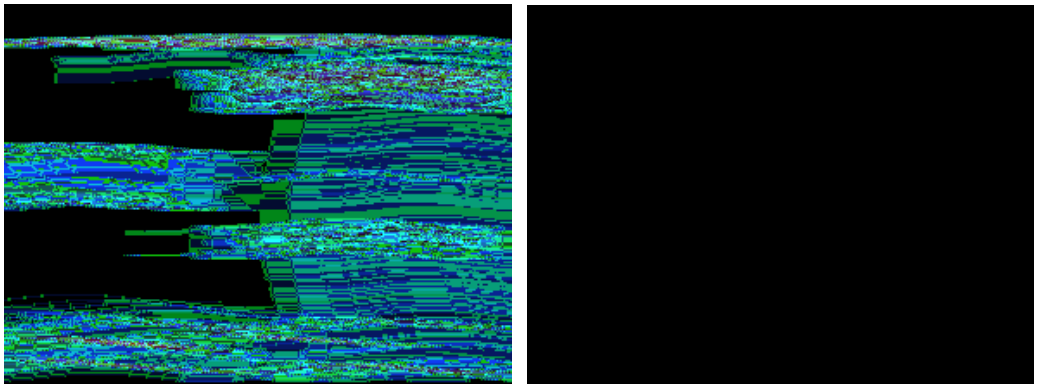
Zero pi



a=5, s=15 threshold image

Fig. 5.2.5.b Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (0,5,15)$

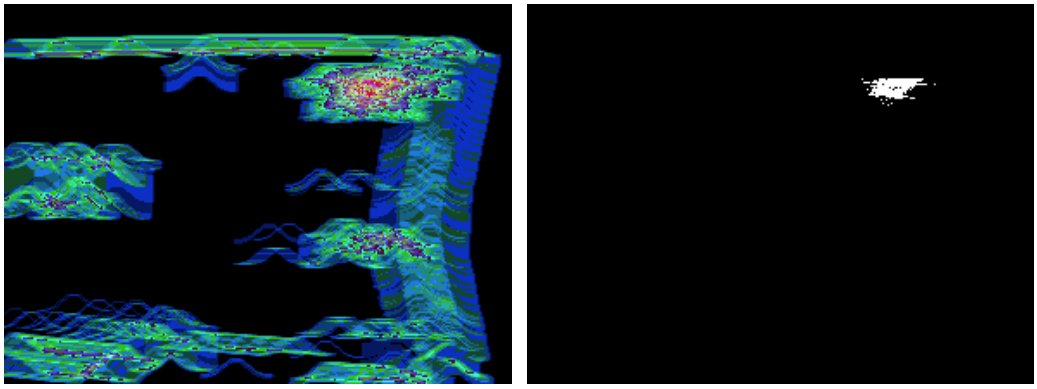
Zero pi



a=5, s=28 threshold image

Fig. 5.2.5.c Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (0,5,28)$

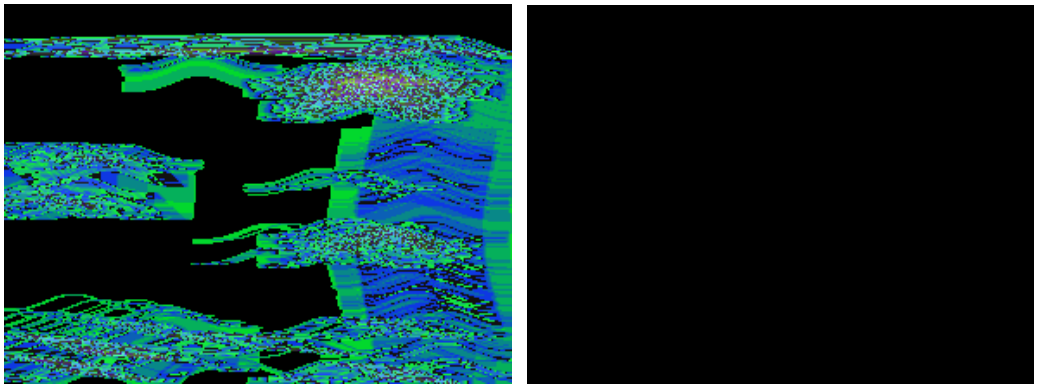
Zero pi



a=10, s=7 threshold image

Fig. 5.2.5.d Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (0,10,7)$

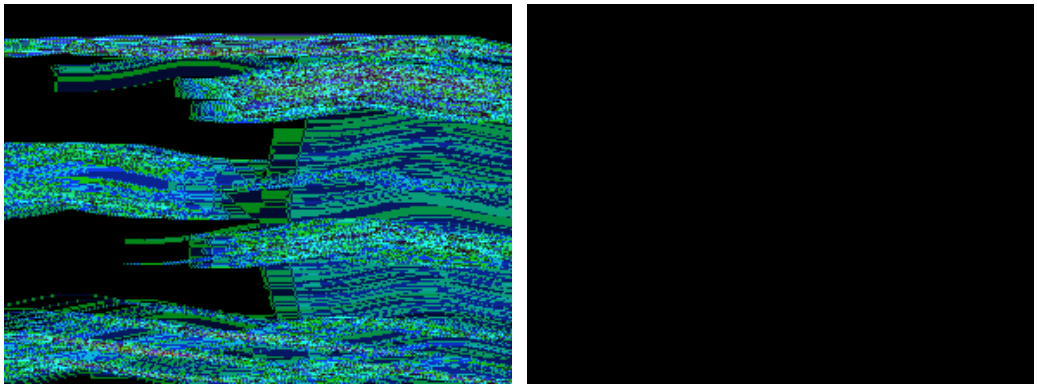
Zero pi



a=10, s=15 threshold image

Fig. 5.2.5.e Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (0,10,15)$

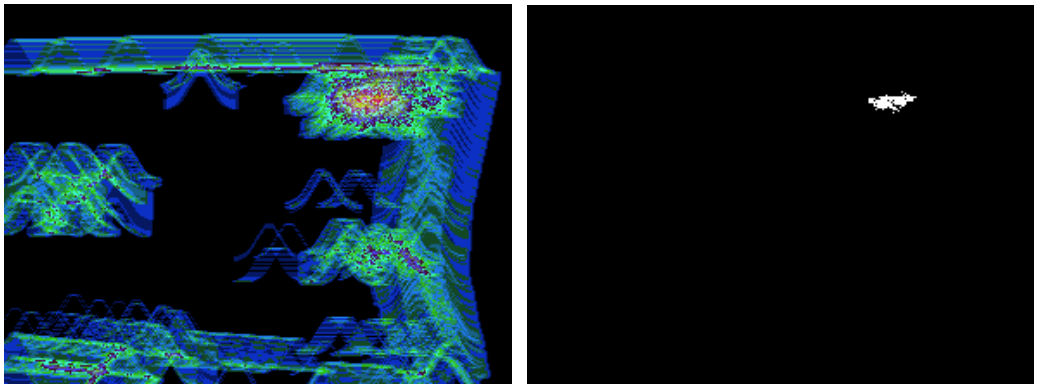
Zero pi



a=10, s=28 threshold image

Fig. 5.2.5.f Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (0,10,28)$

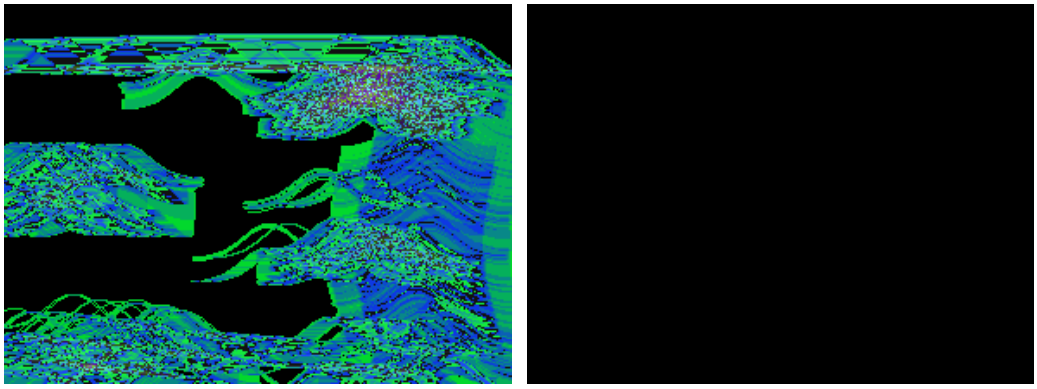
Zero pi



a=20, s=7 threshold image

Fig. 5.2.5.g Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (0,20,7)$

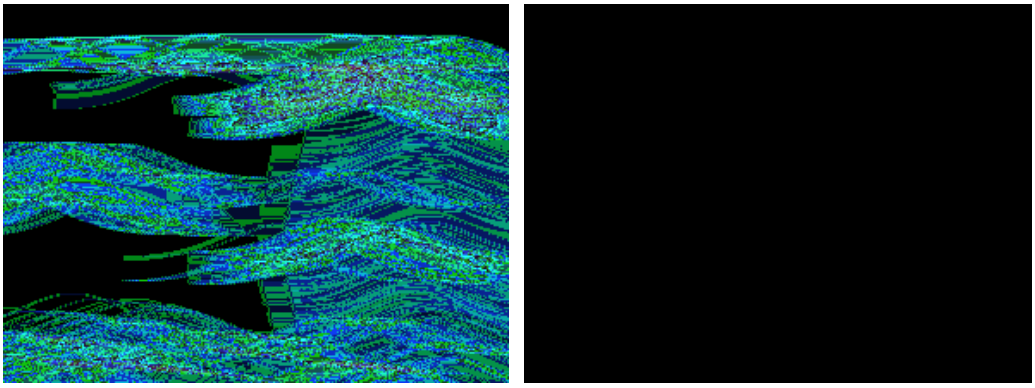
Zero pi



a=20, s=15 threshold image

Fig. 5.2.5.h Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (0,20,15)$

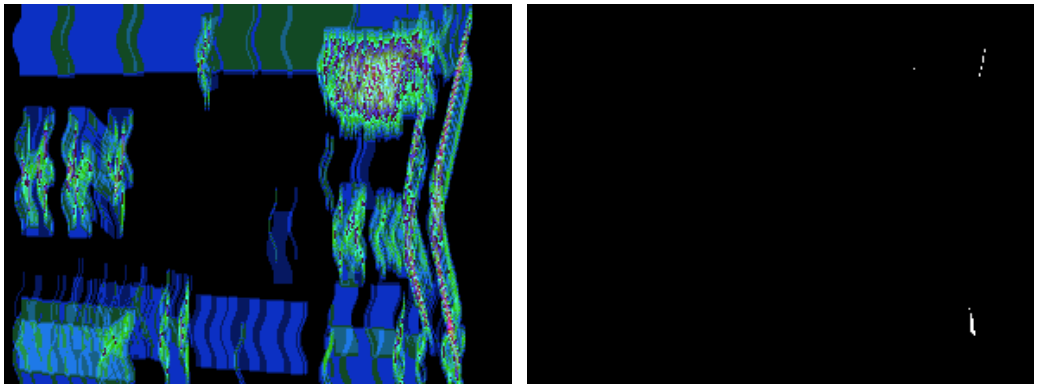
Zero pi



a=20, s=28 threshold image

Fig. 5.2.5.i Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (0,20,28)$

pi/2

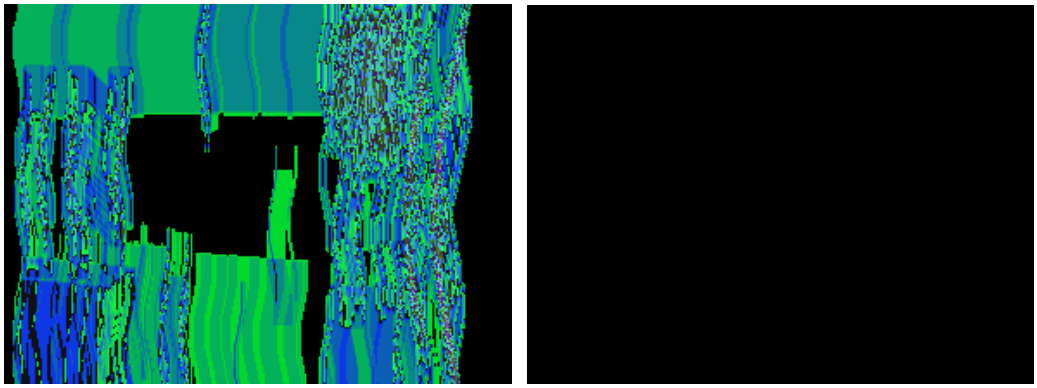


a=5, s=7 threshold image

Fig. 5.2.6.a Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(\frac{\pi}{2}, 5, 7\right)$

pi/2

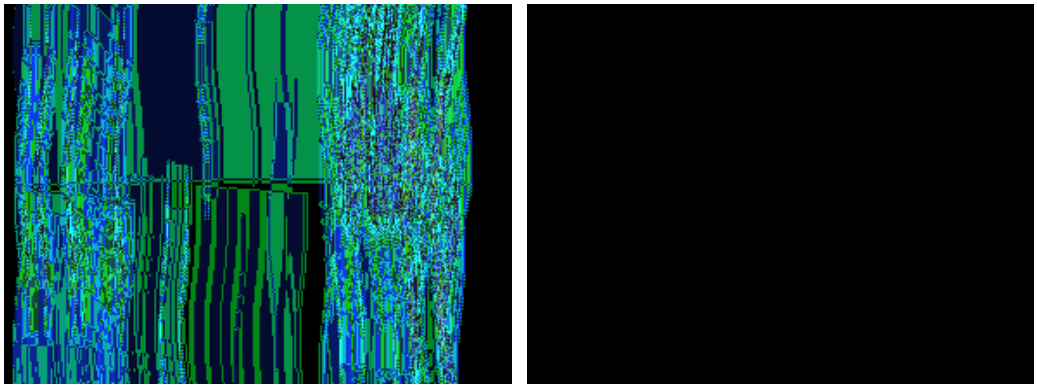


a=5, s=15 threshold image

Fig. 5.2.6.b Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(\frac{\pi}{2}, 5, 15\right)$

pi/2

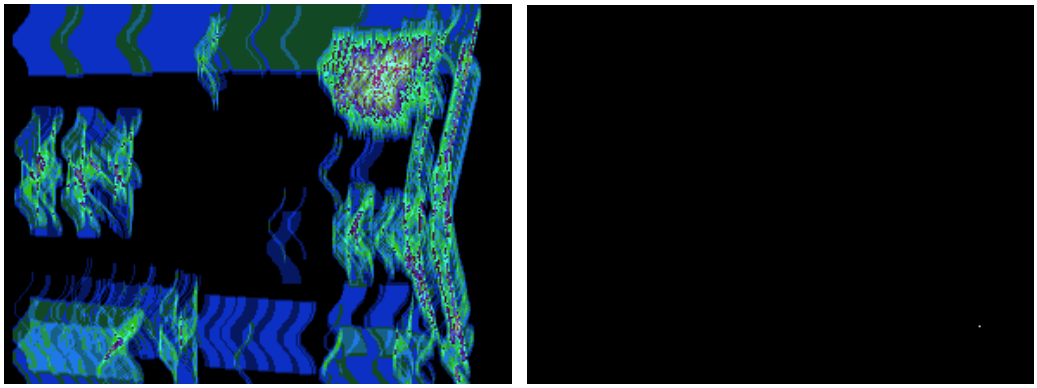


a=5, s=28 threshold image

Fig. 5.2.6.c Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(\frac{\pi}{2}, 5, 28\right)$

pi/2

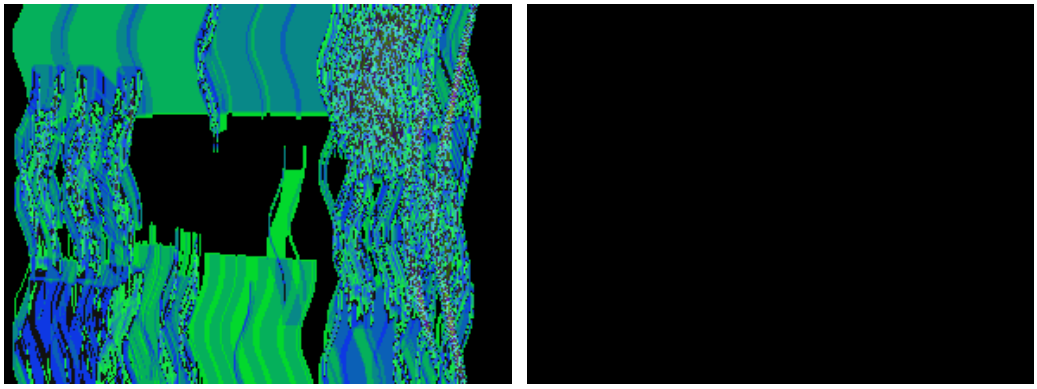


a=10, s=7 threshold image

Fig. 5.2.6.d Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(\frac{\pi}{2}, 10, 7\right)$

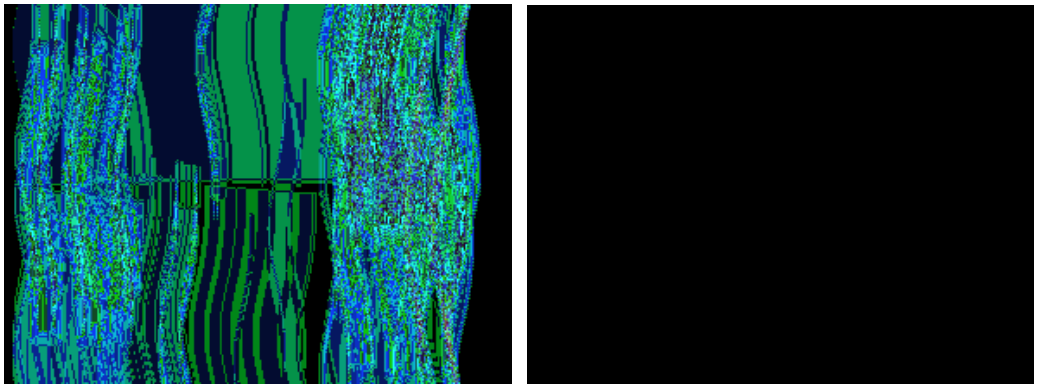
pi/2



a=10, s=15 threshold image

Fig. 5.2.6.e Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = \left(\frac{\pi}{2}, 10, 15\right)$

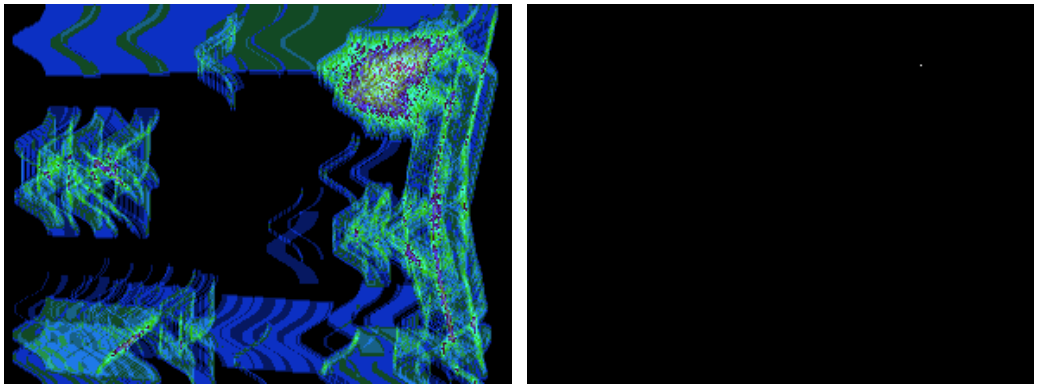
pi/2



a=10, s=28 threshold image

Fig. 5.2.6.f Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = \left(\frac{\pi}{2}, 10, 28\right)$

pi/2

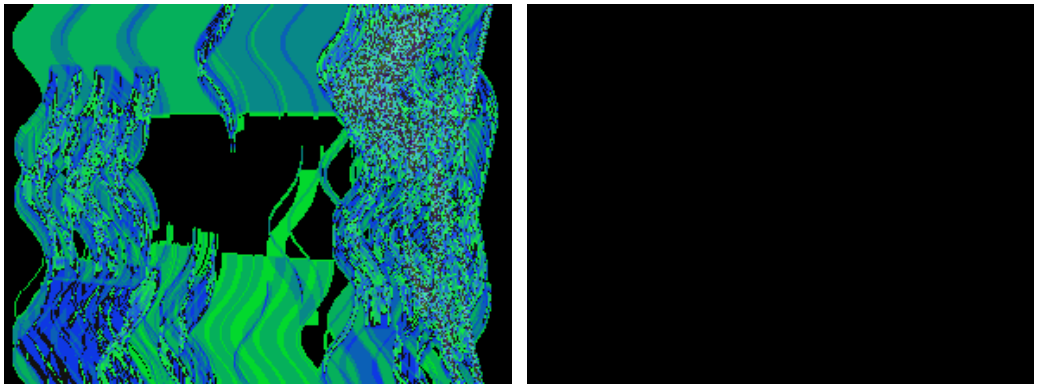


a=20, s=7 threshold image

Fig. 5.2.6.g Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(\frac{\pi}{2}, 10, 7\right)$

pi/2

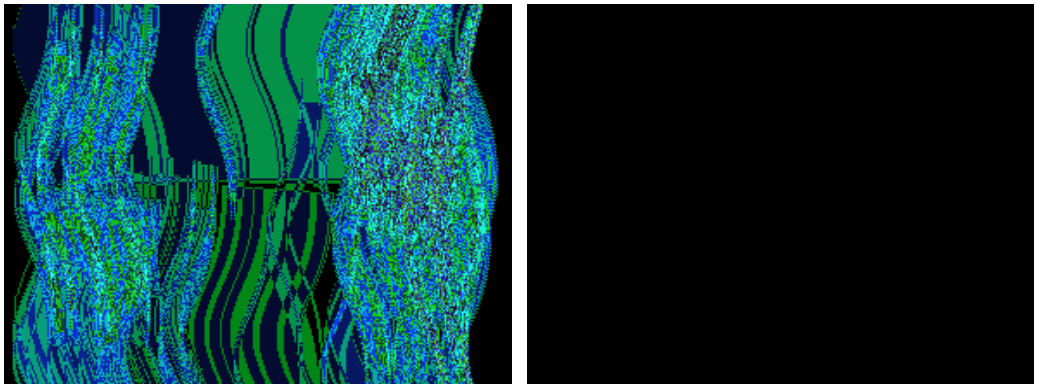


a=20, s=15 threshold image

Fig. 5.2.6.h Feature Images after applied correlation and thresholding

transformations at $(\theta, a, \sigma) = \left(\frac{\pi}{2}, 20, 15\right)$

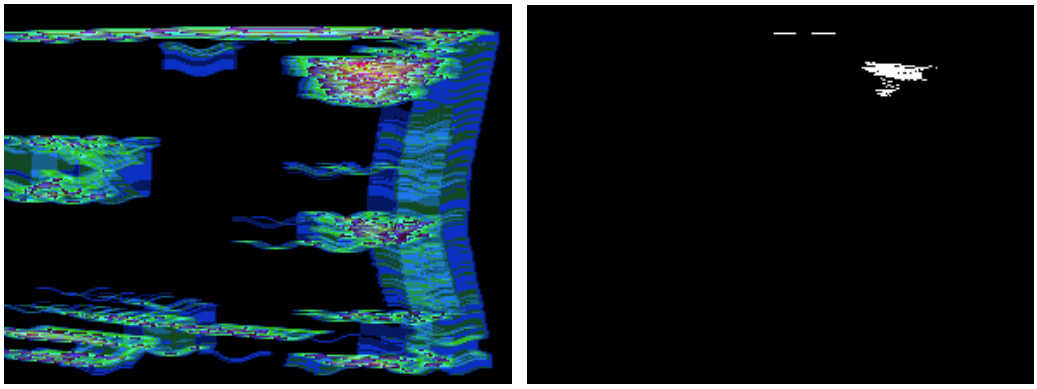
pi/2



a=20, s=28 threshold image

Fig. 5.2.6.i Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = \left(-\frac{\pi}{2}, 20, 28\right)$

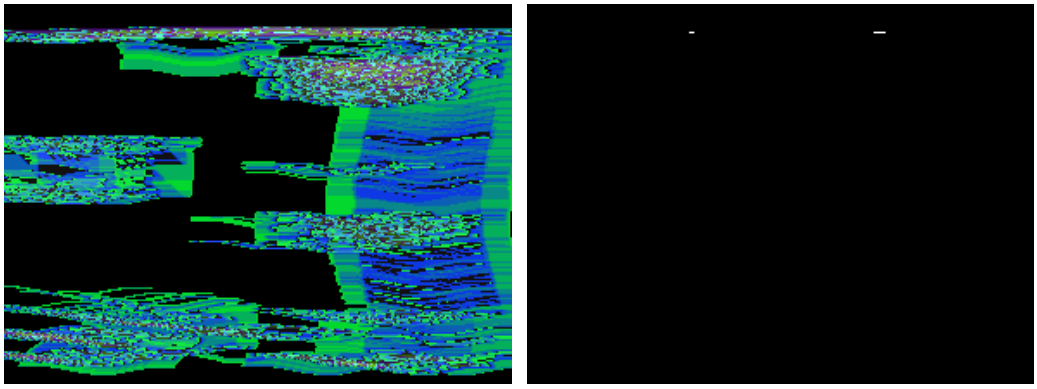
pi



a=5, s=7 threshold image

Fig. 5.2.7.a Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 5, 7)$

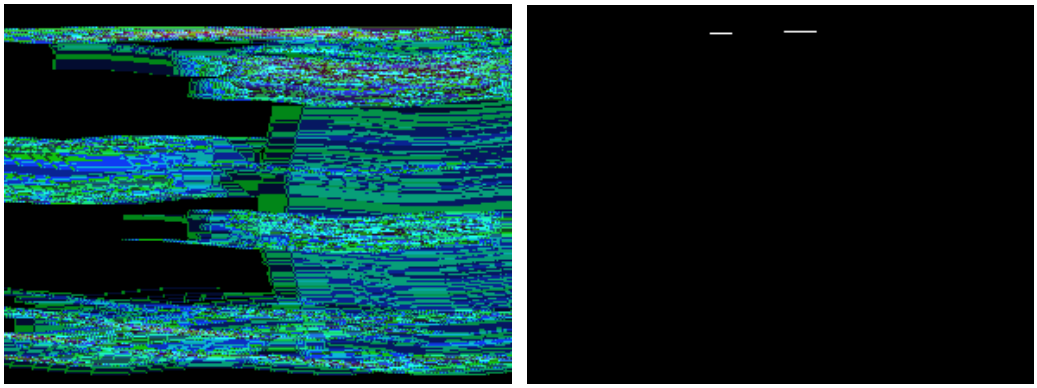
pi



a=5, s=15 threshold image

Fig. 5.2.7.b Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 5, 15)$

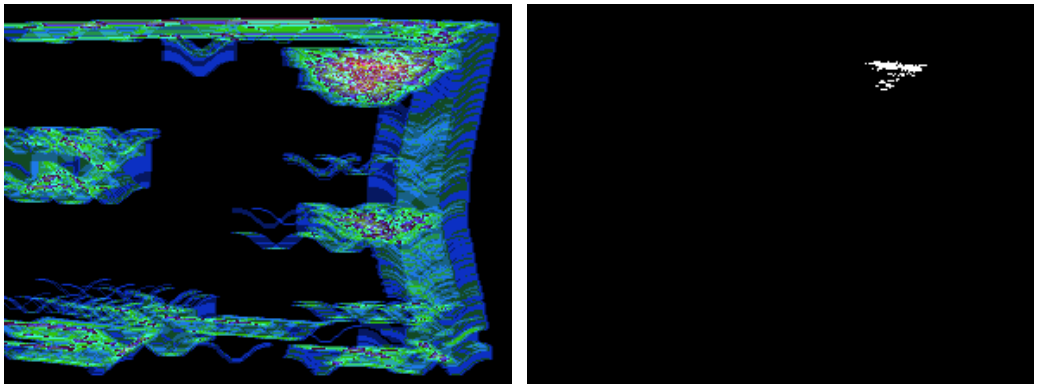
pi



a=5, s=28 threshold image

Fig. 5.2.7.c Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 5, 28)$

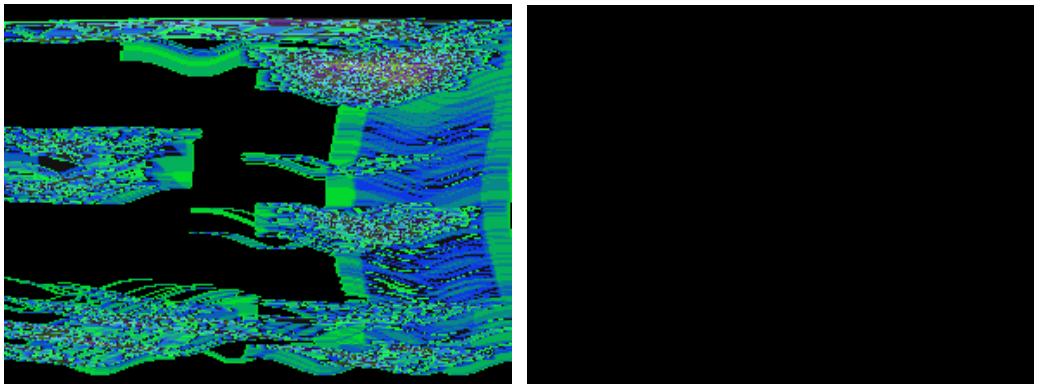
pi



a=10, s=7 threshold image

Fig. 5.2.7.d Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 10, 7)$

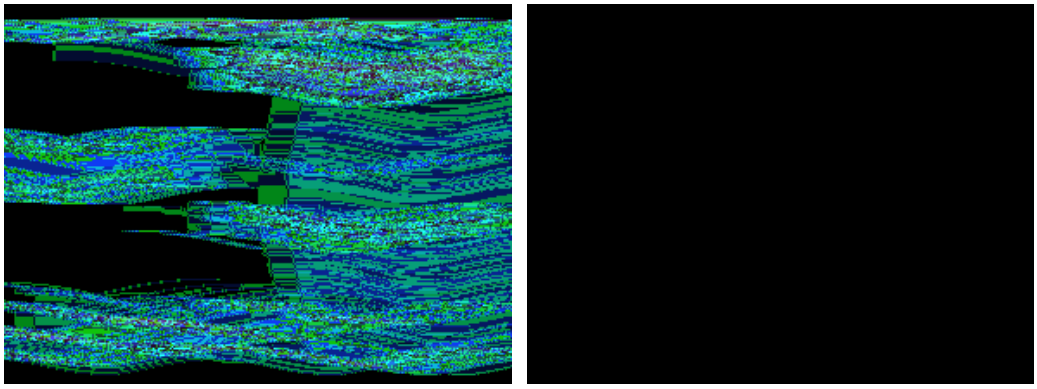
pi



a=10, s=15 threshold image

Fig. 5.2.7.e Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 10, 15)$

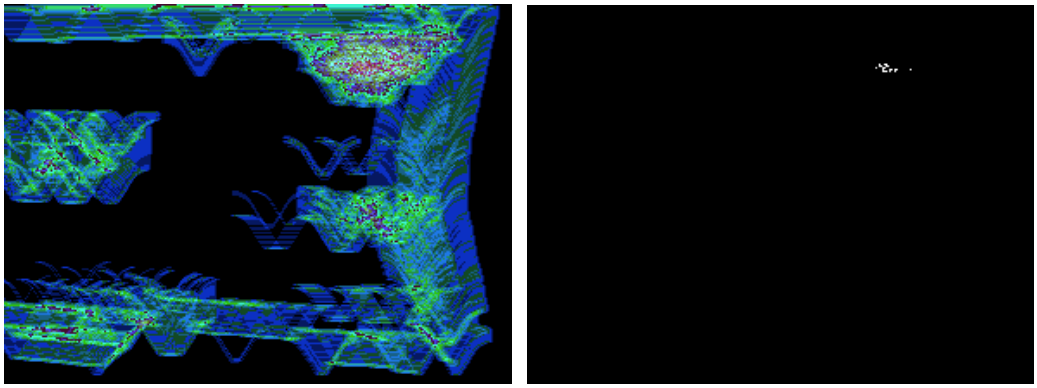
pi



a=10, s=28 threshold image

Fig. 5.2.7.f Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 10, 28)$

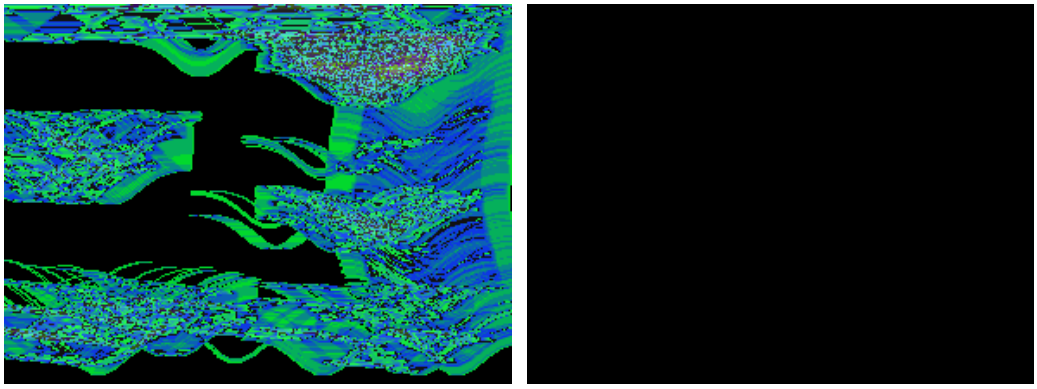
pi



a=20, s=7 threshold image

Fig. 5.2.7.g Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 20, 7)$

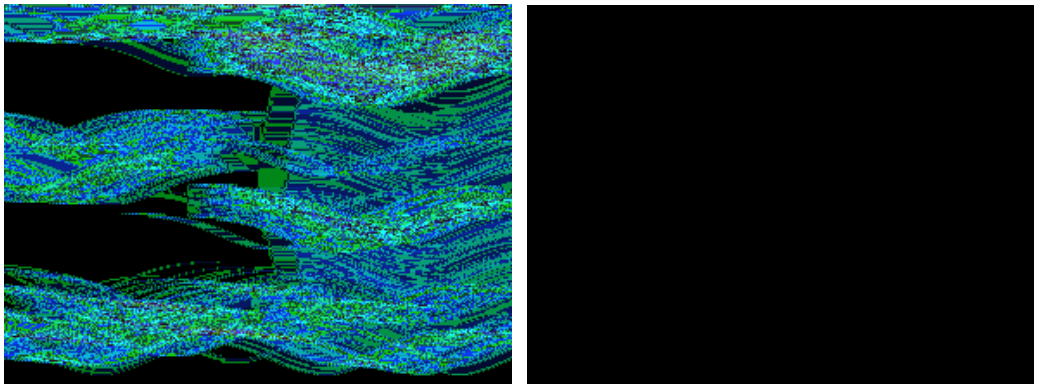
pi



a=20, s=15 threshold image

Fig. 5.2.7.h Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 20, 15)$

pi



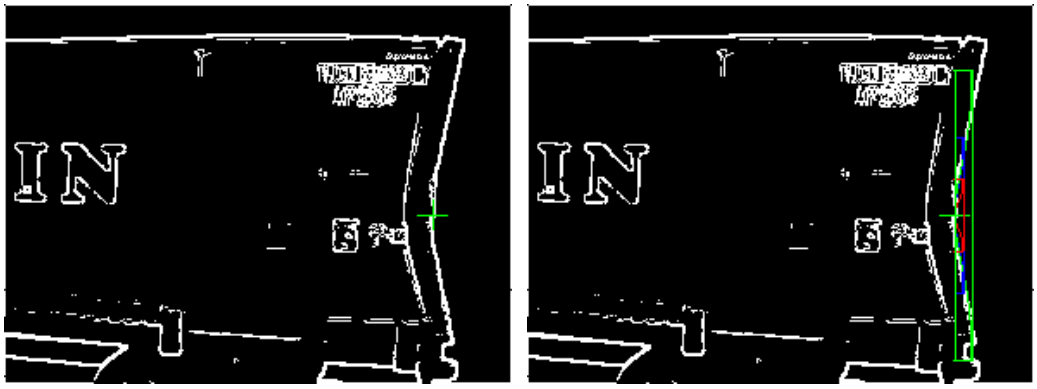
a=20, s=28 threshold image

Fig. 5.2.7.i Feature Images after applied correlation and thresholding transformations at $(\theta, a, \sigma) = (\pi, 20, 28)$

The deformation can be detected from white points in the feature images Fig. 5.2.4.a,b,f correspond to the parameters:

$$(\theta, a, \sigma) = \left\{\left(-\frac{\pi}{2}, 5, 7\right), \left(-\frac{\pi}{2}, 5, 15\right), \left(-\frac{\pi}{2}, 10, 28\right)\right\}$$

This set of parameters is a estimator of damage of the container in image. Estimating the deformation area is in Fig. 5.2.8. Fig.5.2.8(a) displays the center of damage with the green cross (green +) sign in the edge image. Fig.5.2.8(b) displays the area of damage with three lets of gaussian masks correspond to three color of red, green and blue in the edge image. And Fig5.2.8(c) shows the damage area in the original image.



(a) (b)

(c)

Fig. 5.2.8 Recognizing and Estimating the deformation: (a) Center of deformation in the edge image, (b) and (c) Size and orientation of deformation in edge image and origin image, respectively.

Chapter 6: Conclusion

To ensure that the clear image of damages is obtained, Canny edge detector has been implemented. However, some of strict corner of some texts or marks on surface of container might be generate fault or error decisions, especially, when the processes are automatic.

If the distribution of the white points is not spread, high density or high the group's width of meaning, these points may be not considered as damage or deformation, we might be guess these points are belong to a particular region. Inside of Fig. 5.2.5.a,d,g has a group of white points, but that is the text region correspondingly in the origin image. And these groups have just occurred from the set of parameters $(\theta, a, \sigma) = \{(0,5,7),(0,10,7),(0,20,7)\}$, we have a attention that if spreads greater than a threshold, there is not exist white group points. So we should limit the range for spread parameter and also another parameter depends on the application.

Although the correlation operation can be normalized for amplitude changes via the correlation coefficient, obtaining normalization for changes in size

and rotation can be hard. Because, in the real systems, the nature of size and rotation of deformation is unknown, so looking for the best match requires exhaustive changes of size and rotations of the mask.

In this thesis, a new ideal, which matching by correlation between edge image and Capsize-Gaussian-Function, to extract information of deformation shapes and recognize these has been presented. However, in future, we need do more experiments and looking for a decision making and measuring the size of damage or deformation shapes. And further more, we will design an automatic identity check import-export containers system in order to improve and modernize port management and technology.

References

1. Lijun Ding, Ardeshir Goshtasby. On the Canny edge detector. The Journal of The Pattern Recognition Society 34 (2001), pp. 721-725.
2. Suzanna Becker, Mark Pumbley. Unsupervised Neural Network Learning Procedures For Feature Extraction and Classification. International Journal of Applied Intelligence, Special Issue on Application of Neural Networks, Vol. 6, No.3,F. Pineda, 1996.
3. K.Z. Tang, S.N. Huang, K.K. Tan, T.H. Lee. Combined PID and adaptive nonlinear control for servo mechanical systems. Mechatronics 14 (2004), pp. 701-714.
4. SIMON HAYKIN: Neural Networks, A Comprehensive foundation, 2nd ed., Prentice Hall.
5. SERGIOS THEODORIDIS, KONSTANTINOS KOUTROUMBAS: Pattern Recognition, Academic Press.
6. SING-TZE BOW: Pattern Recognition and Image Preprocessing, Marcel Dekker, Inc.
7. ABHIJIT S. PANDYA, ROBERT B.MACY: Pattern Recognition with Neural Networks in C++, CRC Press and IEEE Press.
8. ROBERT M. HARALICK, LINDA G. SHAPIRO: Computer and Robot Vision I, Addison-Wesley Publishing Company.

9. ROBERT M. HARALICK, LINDA G. SHAPIRO: Computer and Robot Vision II, Addison-Wesley Publishing Company.
10. DANA H. BALLARD, CHRISTOPHER M. BROWN: Computer Vision, Prentice Hall.
11. RAFAEL C.GONZALEZ, RICHARD E.WOODS: Digital Image Processing, Prentice Hall, 2002.
12. Bill Green, Canny Edge Detection Tutorial, http://www.pages.drexel.edu/~weg22/can_tut.html, 2002.
13. HARRY WECHSLER: Computational Vision, Academic Press, 1990.
14. COLIN ARCHIBALD, PAUL KWOK: Research in Computer and Robot Vision, World Scientific, 1995
15. JAMES P. IGNIZIO: Linear programming in single- & multiple-objective systems, Prentice Hall, 1982.

Appendixes

Filter_Gaussian function:

```
void CTestNewImageClassDlg::Fiter_Gaussian(CxImage* pImage)
{
	if(!pImage) return ;
	long gauss_kernel[5][5]={	{2,  4,   5,  4,  2},
					{4,  9,  12,  9, 4},
					{5, 12, 15, 12, 5},
					{4,  9,  12,  9, 4},
					{2,  4,   5,  4,  2}	};
	EX_RGBQUAD color;
	DWORD x,y;
	CxImage im;
	im = *pImage;
	long sum,sum_elements;
	int i,j;
	DWORD width,height;
	width = im.GetWidth();
	height = im.GetHeight();
	for(x=0;x<width;x++)
	for(y=0;y<height;y++)
	{
		sum = 0;
		sum_elements = 0;
		for(i=-2;i<=2;i++)
		for(j=-2;j<=2;j++)
		{
			if (x+i>=0 && x+i<width && y+j>=0 && y+j<height)
			{
				color.rgbqColor = im.GetPixelColor(x+i,y+j);

				sum += gauss_kernel[i+2][j+2]*color.dwValue;
				sum_elements += gauss_kernel[i+2][j+2];
			}
		}
		if (sum_elements)
		{
```

```
                color.dwValue = Thresholdex(sum/sum_elements);
            }
            else
            {
                color.dwValue = 0;
            }
            pImage->SetPixelColor(x,y,color.rgbqColor);
        }
}
```

Gradient_Sobel function

```
void CTestNewImageClassDlg::Gradient_Sobel(CxImage* pImage,int magnitude_direction)
{
    if(!pImage) return ;
    long H[3][3]={ {-1,-2,-1},{ 0,0,0},{ 1,2,1} };
    long V[3][3]={ {-1, 0, 1},{-2,0,2},{-1,0,1} };
    EX_RGBQUAD color;
    DWORD x,y;
    CxImage im;
    im = *pImage;
    long** array;
    DWORD width,height;
    width = im.GetWidth()+2;
    height = im.GetHeight()+2;
    array = new long*[width];
    if(!array) return;
    //Allocate memory
    for (x=0;x<width;x++)
    {
        array[x] = new long[height];
        if (!array[x])
        {
            DWORD xt;
            for (xt=0;xt<x;xt++)
            {
                delete[] array[xt];
            }
            delete[] array;
            return;
        }
```

```
}
//Initialization
for (x=0;x<width;x++)
{
	ZeroMemory(array[x],height);
}

//Assign the image matrix
for(x=1;x<width-1;x++)
for(y=1;y<height-1;y++)
{
	color.rgbqColor = im.GetPixelColor(x-1,y-1);
	array[x][y] = (long)color.dwValue;
}
long h,v;
int i,j;
byte gray;
long double angle;
for(x=1;x<width-1;x++)
for(y=1;y<height-1;y++)
{
	h=0;
	v=0;
	for(i=-1;i<2;i++)
	for(j=-1;j<2;j++)
	{
		h+= H[i+1][j+1]*array[x+i][y+j];
		v+= V[i+1][j+1]*array[x+i][y+j];
	}
	if (!magnitude_direction)
	{
		color.dwValue = Thresholdex( long(abs(v)+abs(h)) );
		im.SetPixelColor(x-1,y-1,color.rgbqColor);
	}
	else
	{
		angle = atan2((long double)v,(long double)h);
		angle = angle>0 ? angle : angle+PI;
		gray = (byte)( angle/PI * 180 );
		if ((gray>=0&&gray<=22.5)||(gray>157.5&&gray<=180))
		{
```

```
                        gray = 0;
                    }
                    else if (gray>22.5 && gray <=67.5)
                    {
                        gray = 45;
                    }
                    else if (gray > 67.5 && gray < 112.5)
                    {
                        gray = 90;
                    }
                    else if (gray >= 112.5 && gray <=157.5)
                    {
                        gray =135;
                    }
                    color.rgbqColor.rgbBlue = gray;
                    color.rgbqColor.rgbGreen = gray;
                    color.rgbqColor.rgbRed = gray;
                    im.SetPixelColor(x-1,y-1,color.rgbqColor);
                }
        }
        *pImage = im;
        //DeAllocate memory
        for (x=0;x<width;x++)
        {
            delete[] array[x];
        }
        delete[] array;
}
```

NonMaximum_Suppression

```
void CTestNewImageClassDlg::NonMaximum_Suppression(CxImage* pImage)
{
        if(!pImage) return ;
        EX_RGBQUAD color,color1,color2;;
        DWORD x,y;
        CxImage im_amp,im_dir;
        im_amp = *pImage;
        im_dir = *pImage;
        DWORD highThreshold,lowThreshold;
        highThreshold = 0x00fffff0;
        lowThreshold  = 0x00efffff;
```

```
DWORD width,height;
width = im_amp.GetWidth();
height = im_amp.GetHeight();
Gradient_Sobel(&im_dir,1);
Gradient_Sobel(&im_amp,0);
*pImage = im_amp;
int i,j;
byte flag;
for(x=1;x<width-1;x++)
for(y=1;y<height-1;y++)
{
        color.rgbqColor = im_dir.GetPixelColor(x,y);
        if (color.rgbqColor.rgbRed == 0)
        {
                i =  1;
                j =  0;
        }
        else if (color.rgbqColor.rgbRed == 45)
        {
                i =  1;
                j =  1;
        }
        else if (color.rgbqColor.rgbRed == 90)
        {
                i =  0;
                j =  1;
        }
        else
        {
                i = -1;
                j =  1;
        }
        color.rgbqColor = im_amp.GetPixelColor(x,y);
        color1.rgbqColor = im_amp.GetPixelColor(x+i,y+j);
        color2.rgbqColor = im_amp.GetPixelColor(x-i,y-j);
        if (color.dwValue<color1.dwValue
                                        ||color.dwValue<color2.dwValue)
        {
                color.dwValue = 0;
                pImage->SetPixelColor(x,y,color.rgbqColor);
        }
```

```
else
{
        if (color.dwValue > highThreshold)
        {
                color.dwValue = 0x00ffffff;
                pImage->SetPixelColor(x,y,color.rgbqColor);
        }
        else if (color.dwValue < lowThreshold)
        {
                color.dwValue = 0;
                pImage->SetPixelColor(x,y,color.rgbqColor);
        }
        else
        {
                flag=0;
                for (i=-1;i<=1;i++)
                for (j=-1;j<=1;j++)
                {
                        if(i==0&&j==0) continue;
                        color.rgbqColor =
                        im_amp.GetPixelColor(x+i,y+j);
                        if (color.dwValue > highThreshold)
                        {
                                flag = 1;
                                break;
                        }
                }
                if (flag)
                {
                        color.dwValue = 0x00ffffff;
                        pImage->
                        SetPixelColor(x,y,color.rgbqColor);
                }
                else
                {
                        color.dwValue = 0;
                        pImage->
                        SetPixelColor(x,y,color.rgbqColor);
                }
        }
}
```

```
        }
}
```

Gauss_Draw function

```
void CTestNewImageClassDlg::Gauss_Draw(CxImage* pImage,CPoint
point,double angle,int a,int spread,DWORD c)
{
        if(!m_pImage)
                return ;
        EX_RGBQUAD color;
        WORD num;
        num = 2*3*spread;
        DWORD x,y;
        DWORD xx,yy;
        DWORD* array;
        array = new DWORD[num];
        if(!array) return ;
        ZeroMemory(array,num);
        color.dwValue = c;
        for (int i=0;i<num;i++)
        {
                array[i]= DWORD(a-a*exp(-(i-num/2)*(i-
                num/2)/(2.0*spread*spread))) ;
                x=i;y=array[i];
                xx= (long)(x*cos(angle)-y*sin(angle) -num/2*cos(angle) +
                point.x);
                yy= (long)(y*cos(angle)+x*sin(angle) -num/2*sin(angle) +
                point.y);
                if (xx<0||xx>=pImage->GetWidth()||yy<0||yy>=pImage->
                GetHeight())
                {
                        continue;
                }
                pImage->SetPixelColor(xx, yy,color.rgbqColor);
        }
        delete[] array;
}
```

Vita

Tran Ngoc Hoang Son was born in Nha-Trang, Khanh-Hoa, Vietnam on November 05, 1980, the son of Tran Ngoc Xuan (father) and Tran Thi Phuong (mother). After completing his work at Le Quy Don High School, Nha Trang, Khanh Hoa, Vietnam, in 1998, he entered Polytechnic University, Ho Chi Minh City, Vietnam. He received the degree of Bachelor of Science from Polytechnic University, Ho Chi Minh City, Vietnam in April, 2003. From September 2003 to February 2004, he worked as an assistant in Electrical and Electronic Laboratory of Open University, Ho Chi Minh City, Vietnam. In March, 2004, he entered the Graduate School at Korea Maritime University.

Permanent Address : Room 802C, Dormitory of Korea Maritime University,
Pusan, Korea.

This thesis was typed by the author.