REAL-TIME IMAGE ERROR DETECTION IN KNIFE-EDGE SCANNING MICROSCOPE

A Thesis

by

WENCONG ZHANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Yoonsuck Choe
Committee Members,	John Keyser
	Louise Abbott
Head of Department,	Nancy M. Amato

December 2014

Major Subject: Computer Science

ABSTRACT

Research about the microstructure of the brain provides important information to help understand the functions of the brain. In order to investigate large volume, high-resolution data of mouse brains, researchers from Brain Network Lab (BNL) at Texas A&M University (TAMU) have been developing the Knife-Edge Scanning Microscope (KESM) in the past decade. The KESM can simultaneously section and image brain tissues at sub-micrometer resolution. However, malfunctions of the system can cause imaging errors, which make images fail to provide valid information. Moreover, malfunctions, especially due to obstructions (such as tissue fragments) in the light path of the system, result in continued cutting while the obstructions are present. Since KESM is generally not attended by a full-time human operator, this results in data loss.

To solve the problem, I developed an image error detection method to automatically find imaging errors in real-time. The method can detect errors by analyzing newly acquired images, report results to human operators and even stop the KESM cutting process if necessary so that data loss is avoided. The basic idea of the method is to solve error detection problem through image change detection algorithm as the images acquired by KESM are well-registered and they do not change too much from one slice to the next when there is no error. As a result, the method can detect imaging errors with 86% accuracy (F1-score) and finish a detection routine within 2 seconds, which is sufficient to achieve real-time detection. By integrating the error detection program into the KESM control system, the method enhanced the robustness of the system and reduced data loss.

DEDICATION

To my family.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Yoonsuck Choe, for his steadfast support and guidance during my graduate program. Without his advice and encouragement, this research work would not have been possible. Also, I would thank my committee members, Dr. John Keyser and Dr. Louise Abbott, for their valuable comments and advice. I am also grateful to my colleagues, Daniel Miller, Raj Shah, Jaewook Yoo and Wenjie Yang, and my friends for their advice during my research work.

I would like to extend my gratitude to department faculty and staff for all the rich memories and for making my time at Texas A&M University such a great experience. Furthermore, this research was funded by the National Science Foundation (#1256086).

Finally, I am grateful to my family for their support and love.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation

Research about the microstructure of the brain provides important information to help understand the functions of the brain. In order to investigate large volume, high-resolution image data of mouse brains, researchers from Brain Network Lab (BNL) at Texas A&M University (TAMU) developed Knife-Edge Scanning Microscope (KESM) [9, 3] in the past decade. The KESM is an instrument that can simultaneously section and image brain tissues at sub-micrometer resolution. It has been successfully used to scan whole mouse brains stained in Golgi (neuronal morphology), Nissl (somata), and India ink (vasculature), generating large amounts of brain image data that have been visualized and accessible through KESM Brain Atlas [5].

However, it has been an issue for a while that malfunctions of the system can cause imaging errors and data loss. Imaging errors are generally caused by different malfunctions, like floating tissues which obstruct the light path, and interruption of illuminations. These system malfunctions generate artifacts and errors on the images that make the images failed to provide valid information. What's more, sometimes malfunctions, especially occurring in the light path of the KESM system, result in continued cutting while the obstructions are still present. Since KESM is generally not attended by a human operator, this leads to further data loss. To enhance the robustness of the system and to preserve the integrity of the data, we need to solve the problem. The current workaround is human intervention during imaging where the operator checks and eliminates errors every 30 minutes. This helps reduce the imaging error and data loss to some degree, but it is time consuming and not very

robust. An automated approach is required.

1.2 Approach

In order to solve the problem, I developed an image error detection method to automatically find imaging errors in real-time. The method detects imaging errors, records them and reports to the operators. Additionally when it is necessary, the method stops the KESM cutting process and notifies the operators to remove the obstructions so that further data loss is avoided. There are basically two requirements for the approach: (1) the method should be able to automatically detect imaging errors; and (2) the method should run in real-time. To be specific, it is necessary for the method to detect errors in the interval of two consecutive cut, which is usually around 7 seconds.

The method detects errors by analyzing the acquired images to find abnormal regions. The basic idea is to solve the error detection problem through image change detection. Due to the technique in KESM, images acquired by KESM are well-registered, which sets the foundation of change detection method. Normal consecutive images do not change too much from one slice to the next because the content does not change dramatically in only $1\mu m$ distance. However, abnormal imaging can cause artifacts and errors on the images which show significant change from the normal images. Therefore, the error detection problem can be solved by comparing the newly acquired images with the existing images which in fact serve as prior knowledge.

The change detection program mainly consists of three steps: preprocessing, change detection, and postprocessing. In the preprocessing step, raw image data were first down-sampled to reduce computational complexity while keeping the image content and feature. Then, the images were compensated for illumination variation

caused by illumination noise and variable-speed cutting. For the change detection algorithm, I implemented two methods based on image difference. The first method models the noise in the difference image and sets up a global threshold to find abnormal regions. The second method, adapted from [2], is based on Bayes classifier to classify between change region and no-change region on the consecutive image pair. In the postprocessing step, isolated change regions, which exist in the result of change detection, are filtered because most of them are caused just by noise.

Finally the method decides when to stop the cutting process. In some cases, it is not necessary to stop the cutting when an error is detected. For example, a floating tissue section, which causes artifacts in one image, can be removed by the pump very soon. When some other errors like illumination interruption and disk full error take place, we need to immediately stop cutting. In order to make the method more precise, a finite-state-machine was designed to confirm such cases based on the recent change detection results.

1.3 Evaluation

The error detection method was evaluated in terms of accuracy and speed. First, in order to evaluate its accuracy, I manually labelled three different data sets acquired by KESM as ground truth and compared the result generated by the method with the ground truth. The result shows that the method successfully detects imaging errors with around 86% accuracy (F1-score). Secondly, the speed of the program, the executing time, was evaluated to test if real-time detection is possible. Program execution time data was collected to measure the performance of the method. It shows that the program could finish a routine in less than 2 seconds per slice, which is sufficient for real-time detection.

1.4 Outline of the thesis

This thesis is organized as follows. In Chapter 2, background and related work will be introduced. In chapter 3, I will present the problem formally and then describe the basic approach to solve the problem. In Chapter 4, detailed steps of the method will be described. Next, in chapter 5, implementation and results of the method will be discussed. Finally, in Chapter 6, discussion about the experimental results, open issues, future work, and conclusion will be presented.

2. BACKGROUND AND RELATED WORK

In order to investigate the microstructure in the brain, researchers have developed various techniques and tools to generate high-resolution image data of animal brains. The Knife-Edge Scanning Microscope (KESM) is one of the first instruments in literature which was developed to acquire high-resolution, large volume, three dimensional mouse brain images. In this chapter, technique of KESM will be first introduced as the background. Due to robustness issue in KESM, malfunctions of the system cause KESM imaging error and data loss. Related work and techniques will be reviewed in the second section, including image change detection algorithms as a potential tool for imaging error detection.

2.1 Background

The KESM is a unique instrument that is able to acquire high-resolution, large volume image data of the entire mouse brains. It was initially invented as a brain tissue scanner by Bruce H.McCormick, who was the founding director of Brain Network Laboratory (BNL) in Texas A&M University. Now the instrument is adapted to be able to scan whole small animal organs at a sub-micrometer resolution so that it can be applied widely.

The key feature of the instrument is that it serves as a high-resolution imaging system as well as a high-throughput 3D physical sectioning machine [5, 9]. Accordingly, the instrument mainly comprises four subsystems: (1) a precision positioning stage, (2) knife assembly, (3) microscope and image capture system, and (4) a computer server. The structure of the instrument is shown in Figure 2.1.

The positioning stage was constructed by stacking three different parts to provide mechanical movement along the X, Y and Z axises. By adopting an air-bearing stage,

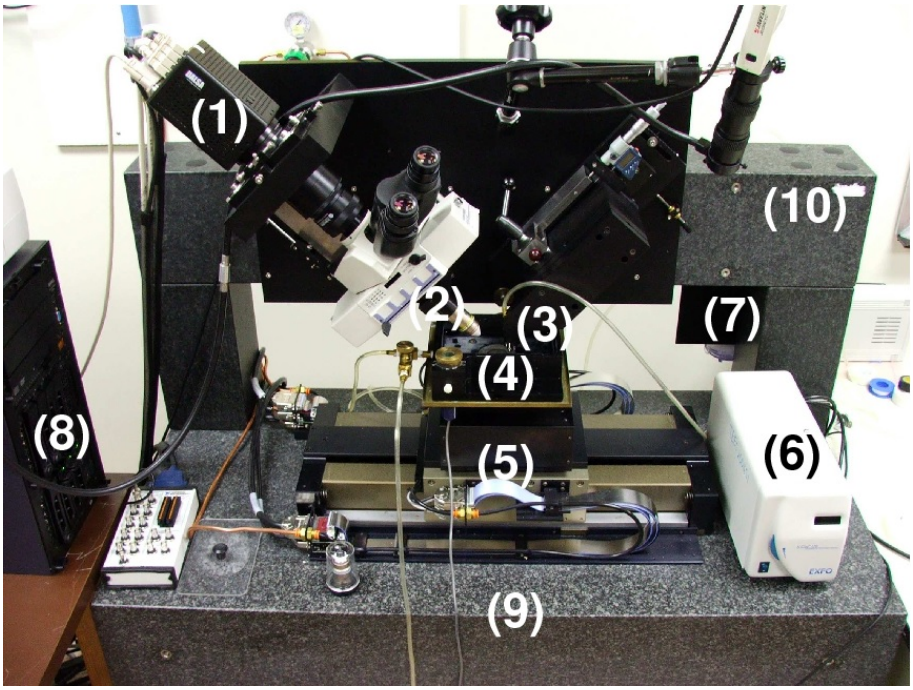


Figure 2.1: A photo of the Knife-Edge Scanning Microscope (KESM). (1) line-scan camera, (2) microscope objective, (3) diamond knife with light collimator, (4) specimen tank, (5) three-axes precision stage, (6) white-light illumination device, (7) water pump (in the back) for the removal of sectioned tissue, (8) computer server, (9) granite base, and (10) granite bridge. Adapted from [3].

the system provides ultra-precision movement which can be achieved as precise as 20 nm in X and Y axises and 25 nm in Z axis [10]. The precision movement also guaranteed the acquired images are well registered.

A custom knife is rigidly mounted to a massive granite bridge over the three-axis stage. A microscopy objective is aligned with the knife on the other side of the stage so that they are oriented in 45 degree and 135 degree respectively. When the objective is perfectly aligned, it focuses on the very tip of the knife.

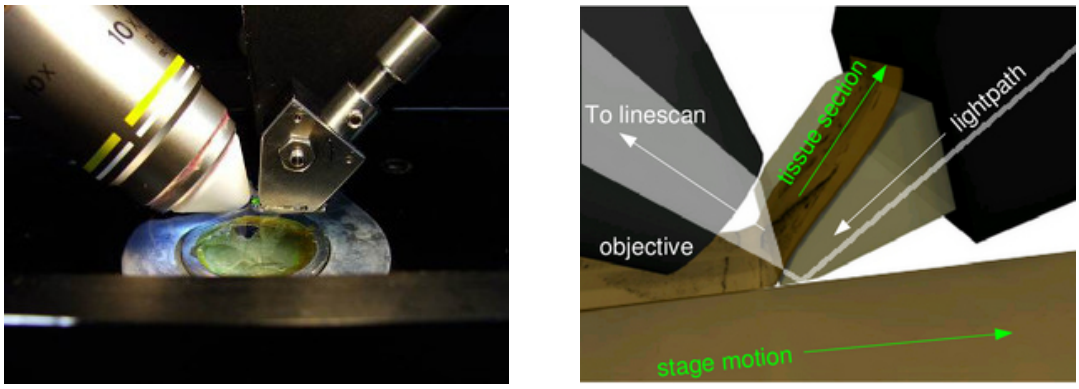
The image capture system mainly includes a high-speed line-scan camera. The camera is able to repeatedly sample a line the at the tip of the diamond knife. Meanwhile, a white light source illuminates the rear of the diamond knife. A stripe of intense illumination is reflected from the rear of the diamond knife to the objective, providing illumination for the tissue at the leading edge. This is shown in Figure 2.2b. The diamond knife plays two different roles: an optical prism in the light path as well as a microtome for physically cutting the thin serial sections.

A computer server is connected to the whole system. In the server, imaging acquisition board and stage controller board are installed to communicate with the camera and the positioning stage. In addition, custom software are developed and installed to precisely control the movement and imaging.

2.1.1 Operation

The operational principle of KESM is to simultaneously section and image the tissue block. The specimen is usually a whole mouse brain which is embedded in a hard polymer resin block. It is mounted on the three-axis precision positioning stage. Instead of moving the knife and objective, the specimen is moved by the stage to be sectioned against the diamond knife, generating a very thin tissue section (usually $1\mu m$). Meanwhile, the newly cut tissue section, which is illuminated by the custom

knife-collimator assembly, is scanned through the microscope objective. The tissue section is imaged at the tip of the knife because the distortion is minimized. In the end, the image signal is transferred through the image acquisition board in the computer server.



(a) Close up of KESM (b) Imaging principle

Figure 2.2: (a) A close look at knife, objective, and stage. (b) A specimen of mouse brain is cut by diamond knife as the illumination passes through the knife. Images are taken while the tissue is being sectioned. Adapted from [3].

One important technique in KESM is stair-step sectioning. Since the field of view in microscope objective and width of the knife are limited, the whole face of tissue block cannot be sectioned in one sweep. We used a lateral sectioning approach called stair-step sectioning to overcome this limitation. As shown in Figure 2.3, a whole tissue block is sectioned laterally into different parts, which are called 'columns'. The number of columns are determined by the width of the tissue block. In order to prevent the tissue block from being damaged, the cutting depth varies in different columns. KESM usually sections and images several piece of tissues in one column before it goes to the next. The numbers in Figure 2.3 illustrates the sectioning

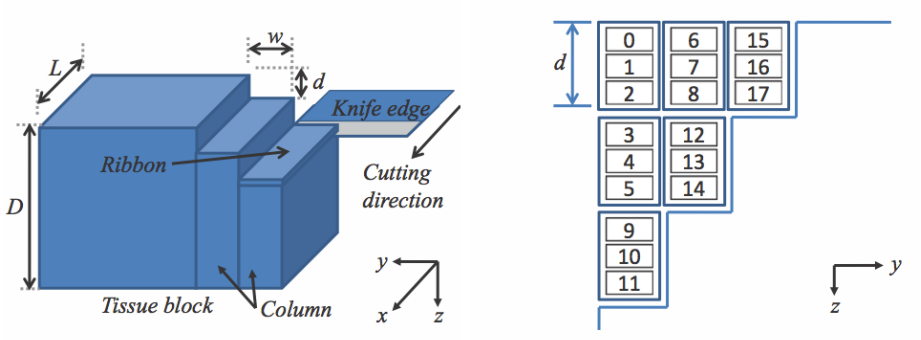


Figure 2.3: Stair-step sectioning. Adapted from [8].

order KESM takes. As a result, images in the same column are consistent and well-registered.

2.1.2 The KESM data sets

By using KESM, we have been able to successfully scan the mouse brain stained in Golgi (neuronal morphology), Nissl (somata), and India ink (vasculature) at a whole-brain scale [4]. These results have been reported in [4, 5]. Selected results are shown in Figure 2.4. Fundamentally, the KESM data set are 3D image stack consisting of 2D images. Techniques have been developed to achieve 3D visualizations of the data sets. The results provide important insights into the system-level architectural layout of microstructures within the mouse brain.

2.2 Related work

In this thesis, I will present an error detection method based on image change detection. Image change detection is to find regions of change in images of a given scene acquired at different times [2]. It has been already well-studied in the image processing field. Change detection is also a foundational image analysis method that can be applied to many diverse applications, including remote sensing, video

surveillance and civil infrastructure monitoring. In this section, applications and methods which are based on change detection will be reviewed.

In remote sensing, change detection methods have been explored in applications like earth's surface monitoring and land cover analysis. In [15], the authors proposed an object-based classification method for change detection in remote sensing. The goal was to compare remote sensing images acquired at the same location but different times to find changed region and then match with the existing data in a GIS database. The algorithm adapted an object-based supervised classification approach in which groups of pixels instead of single pixel were classified. Each object could be described by an n-dimensional feature vector and could be classified to the most likely class using maximum likelihood (ML).

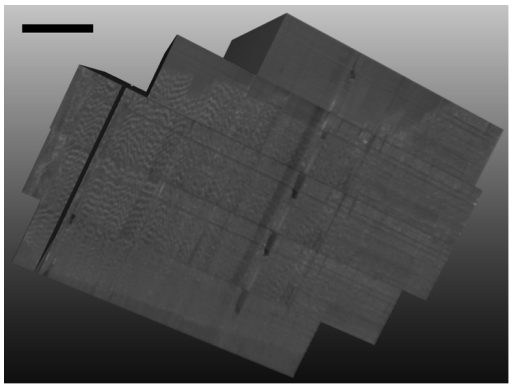
Moreover, change detection in video surveillance solves similar problems as what we are facing in KESM. In [1], Brocke describes their work about detecting abnormal scenes in a video surveillance system. Basically their goal was to detect irregular welding to control the quality of a mass production laser welding process. By detecting sudden changes in image sequences, their method was able to find brighter regions which were likely to be abnormal. Their change detection algorithm was based on statistical method in which change regions and no-change regions were classified accordingly. Then the change region was segmented from the background and was analyzed in the next step. The overall result showed that the proposed method was able to detect abnormal scenes with acceptable recall rate, but it still suffered from a high false alarm rate. Other applications in video surveillance are interested in tracking moving objects. For example, in a traffic monitoring system, moving vehicles are the main concern. Foreground and background analysis are often adapted in this kind of application. In [6], the authors presented a video surveillance system based on change detection. The main task was to detect moving objects

by comparing image sequences and subtracting objects from the background. By integrating statistical assumptions about object-level knowledge, they enhanced the background model. Apparent objects and shadows were also detected to update the background dynamically. Finally, moving objects were detected in the video stream by subtracting from the background. Tian et al.[14] pointed out that the computational complexity of the previously mentioned method was too high to achieve a real-time workflow. They presented a method to efficiently and effectively analyze foreground objects based on Gaussian Mixture Model (GMM). The background was modeled by mixture of Gaussian so that they could be subtracted from the image and foreground parts could be found. By adapting texture information and illumination intensity, their method could remove object shadows and work with quick lighting changes. Instead of using tracking and motion information, foreground was also described by mixture of Gaussian model.

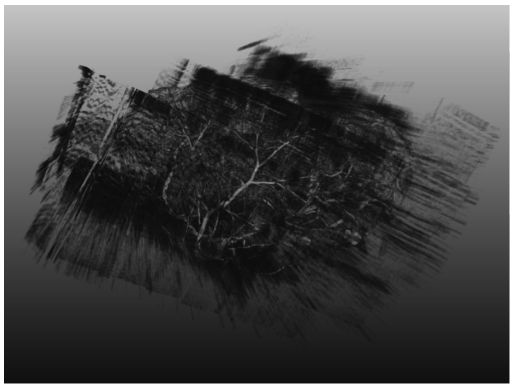
In video processing, an application similar to the above is the detection of abrupt scene change in order to segment a video. Video segmentation is useful when separate clips are characterized, stored and retrieved. A simple idea is to analyze image frames in the video by computing the sum of absolute pixel-wise intensity difference between two consecutive frames. An improved method would compute the difference based on image histogram as different scene shows significantly distinct histogram property. In [7], the authors presented a method which combined difference features from pixel-based value and histogram-based value to measure the general difference between frames. The advantage of their method was to consider the spatial information provided by pixel-based difference while ignoring changes caused by fast camera and object motion as much as possible. Based on the idea, they developed a concrete workflow and set up an adaptive threshold for the difference value. However, Meng et al.[11] thought that the previously described method required decompressing the

video first since pixel-based and histogram-based differences were calculated from pixel-wise. For the MPEG standard, decompressing was a computationally intensive process. Therefore they presented a different algorithm to detect abrupt scene change in a compressed MPEG/MPEG-2 bitstream with minimal decoding. Scene change detection was achieved by computing and comparing the Discrete Cosine Transform (DCT) coefficients and motion vectors. The idea was somehow similar to histogram similarity, but it did not require decomprsessing.

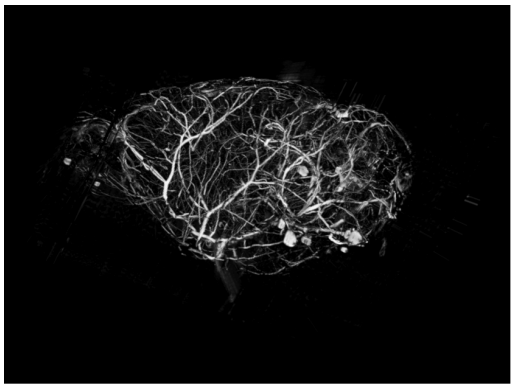
In the above mentioned applications, change detection was implemented by different methods, including statistical model, object-based method, and classifier-based method. In practice, a simple method to start with is to compute the difference of two images and then apply a global threshold. This is an easy but widespread method. Threshold can be chosen empirically and by trial and error to achieve a specific false alarm rate. However the threshold is chosen, simple difference is not robust enough. A statistics-based strategy often model the distribution of change and no-change regions. Then a hypothesis test is applied. Since this method requires assumptions about the distribution of change and no-change region, a reasonable assumption is dependent on a specific application. More complicated change detection methods also exist in the recent research work, including predictive models, shadowing model and background model. By adopting spatial, temporal and illumination information, more complicated change detection methods require higher computational complexity. In general, Radke et al. systematically investigates image change detection algorithms in [12]. In their survey, the problem was well defined and the common processing steps were presented, which is a good starting point for a survey of specific change detection applications.



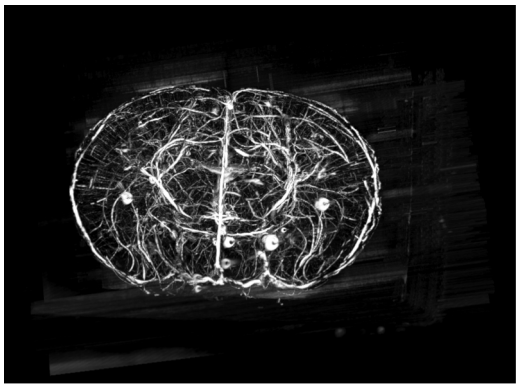
(a) Raw data volume(bar = 1.44 mm)



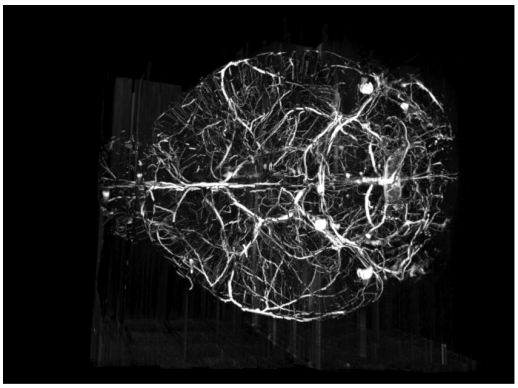
(b) Initial thresholding of (a)



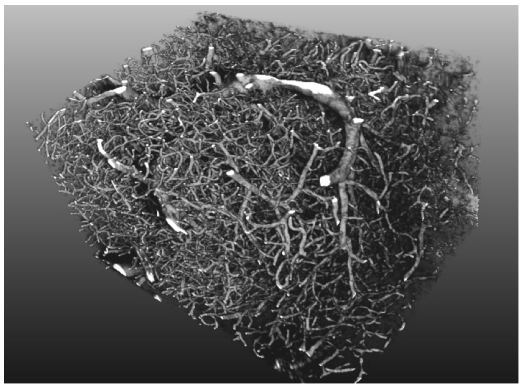
(c) Sagittal view.



(d) Coronal view.



(e) Horizontal view.



(f) Close-up (1.5 mm-wide block).

Figure 2.4: KESM Vasculature Data. (a) shows a 3D view of raw data in the sagittal. (b) shows a lightly thresholded version of (a) which reveals the boundary of the content. (c)-(e) are fully thresholded version of (a). The data were shown in different views, including sagittal, coronal, and horizontal. (f) Close-up of the intricate details. Adapted from [4].

3. PROBLEM DESCRIPTION AND BASIC APPROACH

Although KESM has been successfully used to acquire high-resolution image data of mouse brains, there is still room for improvement. One of the existing problems is imaging errors and data loss caused by system malfunctions. The problem impairs the robustness of the system and the integrity of the data set. In this chapter, I will describe the problem in detail and present the basic rational of my proposed method.

3.1 Issues in KESM imaging

In the existing KESM system, system malfunctions could lead to imaging error and data loss. To be specific, malfunctions are system errors that could happen in positioning stage, linescan camera, light path, diamond knife, and the pump. They are caused by different reasons, including floating tissues that obstruct the light path, illumination interruption etc. As a result, the malfunctions damage the quality of the acquired images, by causing artifacts and errors on the images, as shown in Figure 3.1. Images with artifacts and errors fail to provide valid information about the tissue. Moreover, since the KESM sectioning process is generally unattended, some malfunctions are likely to persist and result in the system to continue cutting with the existence errors. If we do not eliminate the errors in time, large amount of data can be affected by errors, therefore causing a substantial data loss.

In practice, there are three major causes for imaging errors:

(1) Floating tissue sections between the diamond knife and objective. Ideally, a newly cut tissue section should be removed by the pump quickly. However, in practice, floating tissue sections are often not removed in time and obstruct the light path. As a result, they generate unwanted artifacts on the acquired images, such as a white abnormal region. When this occurs individually, it does not lead to severe

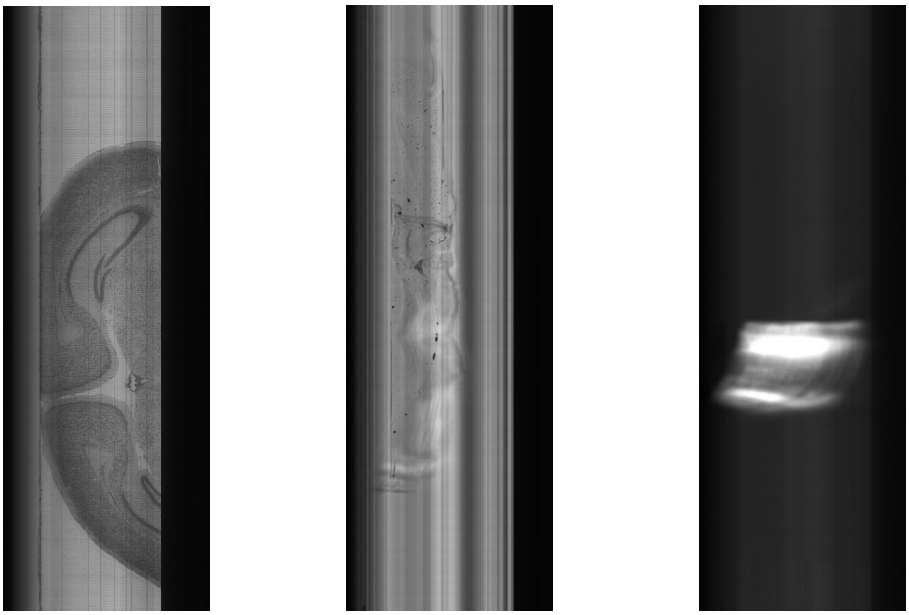


Figure 3.1: KESM raw image data. (a) a normal image, (b) an image with artifacts (c) an image with error caused by floating tissue between objective and knife.

data loss, because the errors usually disappear quickly once the obstructing tissues are absorbed by the pump in the subsequent run. However, there are cases when the pump is clogged by sectioned tissues or just not working, causing the error to appear consistently and a potential data loss. In such a case, we have to stop the cutting process and clean out the obstructing tissue. The floating tissue problem is the most frequent error in practice.

(2) Illumination interruption. Illumination interruption can be caused by malfunctions of the illumination device and obstructions on the light path. With the existence of illumination interruption, the illumination intensity of the newly acquired image is changed dramatically. When critical illumination interruption happens, resulting in very dark images, it is necessary to stop the cutting and fix the problem. Although, there are cases when illumination intensity variations in the image sequence are acceptable. Most of the time, they are caused by variable-speed cutting,

which does not cause significant intensity change.

(3) Disk full issue. This type of error is different from the above as it happens on the computer server and cannot be solved through image analysis. When there is no sufficient free space for newly generated images, tissue block will still be cut but not saved as images, resulting in a potential data loss. We need to stop the cutting process immediately and make space on the disk.

Our current workaround is human intervention during the cutting process. To be specific, an operator checks the system status and eliminates errors every 30 minutes or so. The approach helps reduce the imaging error and data loss to some degree, but there are some limitations.

(1) The approach is time consuming for human operators. The operators have to be in the lab or to have access to the computer server during the experiments.

(2) The approach is not robust enough. There are still possibilities that errors take place in the interval of human intervention, resulting in data loss. Operators can check the status more frequently to enhance robustness, but it would be more time consuming.

We need an automated method.

3.2 Basic approach

The basic idea of the proposed method is to automatically detect imaging errors in real-time through image analysis. Besides, the method is designed to record errors, report to the human operators, and even stop the cutting process when it is necessary. As a result, two aspects are improved accordingly:

(1) Operators will have less burden thanks to automatic error detection.

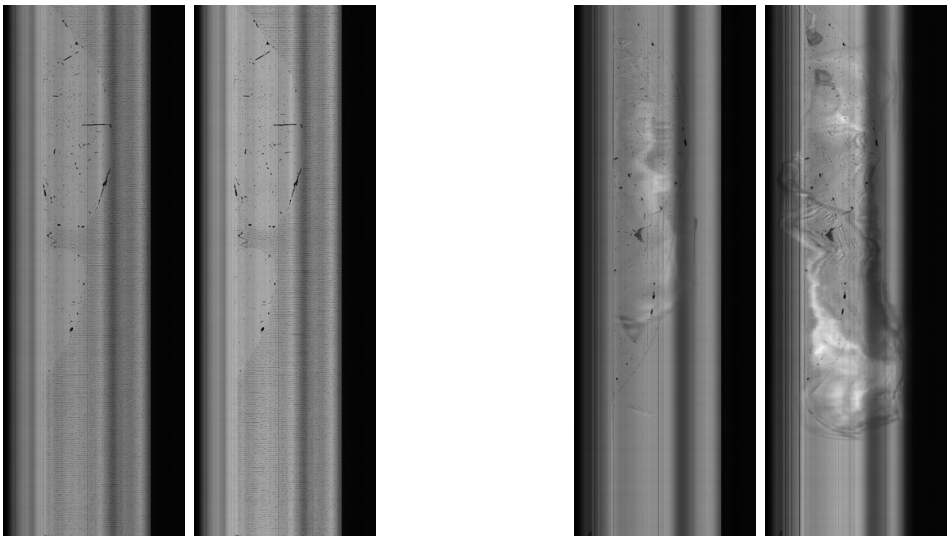
(2) Imaging errors are detected in a more robust way, resulting in a better solution to prevent data loss.

We prefer the method to be designed as a software solution so it can be integrated with the existing KESM control system. There are generally two major requirements for the method: (1) the method detects imaging errors automatically. (2) The method detects errors in real-time. To be specific, the method is expected to execute between the time period of two consecutive KESM cutting passes. The interval is usually 7 seconds in practice.

To detect imaging errors, image processing and analysis were incorporated into my method. Although disk full issue is different and it can be monitored by computer program, imaging errors caused by the other reasons could be found through image analysis. One approach to start with is object-based detection as error regions could be viewed as a special kind of object. However, image errors are caused by different reasons, making the appearance take various shapes: it is hard to extract common features from the various erroneous shapes.

A more realistic method is to compare newly acquired images with existing images. The KESM image data hold several characteristics which may prove useful. One in particular is that the images are generated in strict sequential order, thus resulting in good registration across the images. Since the KESM only sections tissue in $1\mu m$ for each step and the brain structure does not change dramatically over such a thin section, the content of a consecutive image pair should be consistent. On the other hand, images with artifacts and errors are likely to exhibit significant change compared to the previous images. As shown in Figure 3.2, a normal consecutive image pair shows little change while the abnormal image sequence often appears to have a sudden change.

A key insight from the above is that error detection in KESM can be solved through change detection because imaging errors often come with significant changes in the image sequence. In our experience, floating sections which obstruct the light



(a) (b)

Figure 3.2: Examples of consecutive image pairs. (a) is a consistent image pair. (b) is an image pair with significant change regions.

path often generate a large white region in the error image. In addition, interruption of illumination and false positioning can also be detected by exploring changes in the image pairs. The existing images provide prior knowledge so that abnormal regions can be detected. Therefore, the basic rational behind the method is to adopt change detection to find error regions.

4. METHOD

In the previous chapter, I presented the research problem and the basic approach of my method. In this chapter, I will elaborate on the methodologies. The error detection method mainly consists of three steps: preprocessing, change detection, and postprocessing. Based on the results of error detection, I developed a finite-state machine which decides when to report to the human operators and even stop the machine in case the situation is critical. The major steps are shown in Figure 4.1.

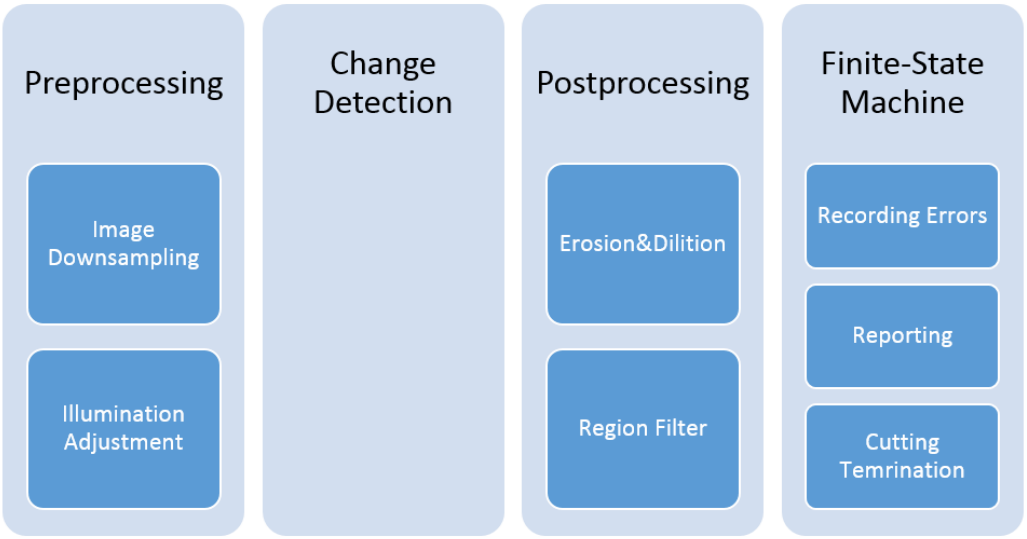


Figure 4.1: Major steps for error detection.

4.1 Problem statement

To make our following discussion precise, the following mathematical expressions are involved. Suppose the image sequence acquired by KESM is $I_1, I_2, ..., I_N$. In

each image, pixel coordinate $(i, j) \in \mathbb{R}^2$ has intensity ranging from 0 to 255. In our application, we mainly focus on two consecutive images whose intensity can be expressed as $I_1(i, j)$ and $I_2(i, j)$, respectively. The width and height of the image are I and J respectively. The histogram of gray-scale image shows the number of pixels in a particular intensity, which can be expressed as H(x), where $0 \leq x \leq 255$. A common intermediate result in change detection is the difference image. A difference image $D(i, j)$ is generated by

$$D(i, j) = |I_1(i, j) - I_2(i, j)|$$

A change detection algorithm should take $I_m(i, j)$ as input and generate a resulting binary image B(i,j) where:

$$B(i, j) = \begin{cases} 1 & : \text{change detected at pixel } (i, j) \text{ between image } I_1 \text{ and } I_2 \\ 0 & : \text{otherwise} \end{cases}$$

4.2 Preprocessing

Preprocessing is an important step to remove unwanted elements in an image so that image processing can be more effective in the next step. There are two major issues with the raw image data. First, the size of the raw images acquired by KESM is too large for efficient processing. Second, noise and illumination variation that exist in the KESM images are not the changes we desire to detect. In order to get rid of these issues, preprocessing is required. Preprocessing mainly consists of two steps: image downsampling and illumination adjustment.

In most change detection applications, image registration is necessary because

there are significant intensity changes due to camera motion. They are considered as real changes if we focus on the change of the content. Image registration, also known as geometric adjustments, is to align several images into the same coordinate frame so that the changes due to camera motion can be removed. However, as I mentioned before, images acquired by KESM are already well-registered, the registration step can be just skipped. What's more, any movement in the camera should be detected as they are likely to be caused by mechanical errors in the system.

4.2.1 Downsampling

In signal processing, downsampling is the process of reducing the sampling rate of a signal. When it comes to image processing, it means resizing the images to a smaller size by skipping or averaging pixels. Since the size of the raw image data acquired by KESM is around 46 MB, it is not ideal for efficient processing. Before we apply any change detection algorithm, the raw images were downsampled by 16 times in each dimension, generating a resized image of 256×750. The resulting size is also suitable for display.

The main reason for downsampling is to achieve a better computational efficiency. Resized images allow faster image processing and help improve the overall performance in the following steps, because the computational cost of an image processing algorithm often highly depends on the size of the image. By comparison, image processing on raw images is highly time-consuming and unaffordable. For example, the change detection algorithm to be discussed below would need more than 180 seconds to finish computation on the original-size image when it was tested on a modern PC. It is impossible to achieve real-time detection in such cases.

Image scaling is not a trivial process because it involves a trade-off between efficiency, smoothness and sharpness. There are different methods for image sub-

sampling, including nearest-neighbor sampling, bicubic sampling, and anti-aliasing.

Nearest-neighbor sampling is the simplest method. In downsampling, nearest-neighbor sampling picks only one pixel from the nearest region in the original image to generate the rescaled image. An illustration figure is shown in Figure 4.2.

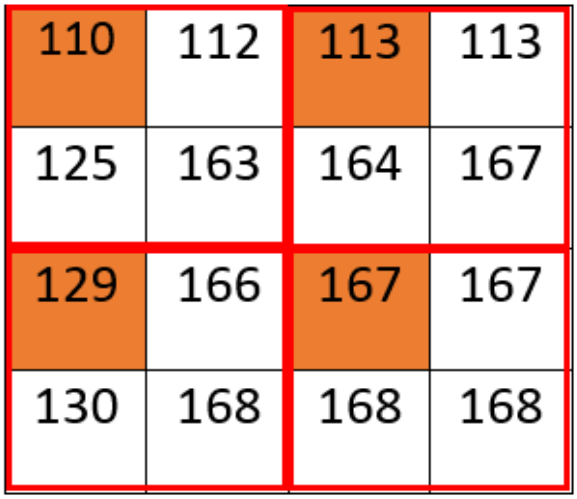


Figure 4.2: Illustration of nearest-neighbor sampling.

In the figure, the original image with dimension 4×4 is scaled to small image with 2×2. The nearest-neighbor sampling just picks one pixel (the pixel highlighted with orange color) from every 2×2 region (highlighted with red line). In our case, the algorithm will pick one pixel from 16×16 region. Nearest-neighbor sampling is the fastest method while the image quality is the lowest. Artifacts and jaggies can be seen in the rescaled images.

Bicubic sampling is the most common method in image processing softwares. The idea behind bicubic sampling is using convolution kernel to average the pixel values of the nearest region in the original-size images. The convolution kernel is composed

of piecewise cubic polynomial. The output pixel value is a weighted sum of pixels in the nearest 4×4 neighborhood. The kernel k(x) :

$$k(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & : \text{ if } x \leq 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & : \text{if } 1 < x < 2 \\ 0 & : otherwise \end{cases}$$

where $a = -0.5$ is the common value. Compared with nearest-neighbor sampling, bicubic sampling can generate smoother images while it is more computationally expensive.

Spatial anti-aliasing is the technique of minimizing the distortion artifacts known as aliasing when representing a high-resolution image at a lower resolution. It generates the best quality images and requires highest computation cost compared to the above. In image processing, a simple approach to achieve anti-aliasing is using the average intensity of a rectangular area in the original image corresponding to the pixel. An example is shown in Figure 4.3.

(a) (b)

Figure 4.3: Examples of anti-aliasing. (a) is a large atmark rendered without antialiasing. (b) is a large atmark rendered with antialiasing. Antialiasing smoothes out the jaggies.

In summary, the three different sampling methods have different output image qualities and different computational cost. I tried all the three methods to test their speed. Here is the comparison:

Table 4.1: Downsampling methods

Name	Image quality	Computational cost	Execution time(s)
Nearest-neighbor	Lowest	Lowest	0.87
Bicubic	Intermediate	Intermediate	0.98
Anti-aliasing	Highest	Highest	4.12

From the table, the average execution time for anti-aliasing on the computer server is above 4 seconds, which makes real-time detection difficulty. To balance the image quality and execution time, bicubic sampling was chosen in our case. If the aging computer server for KESM is updated in the future, anti-aliasing method can be adopted.

One of the major concerns with downsampling is the reduced image quality, as downsampled images lose resolution. However, it is not an issue in our case, as the low resolution image still maintains the basic image features. Significant changes can still be detected.

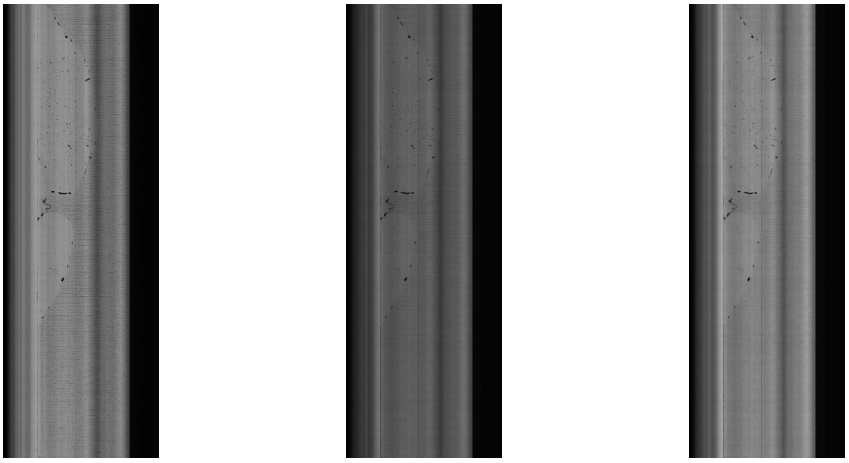
4.2.2 Illumination adjustment

In KESM imaging, illumination interruption errors lead to significant intensity changes, which need to be detected. However, there are cases where normal illumination variations take place in image sequence. They are expected and manageable because normal variations are usually caused by variable-speed cutting. In KESM, variable-speed cutting is a technique to random the speed of cutting in order to

improve the image quality. Due to variable-speed, the exposure time of the line-scan camera changes, causing variation in image intensity. As a result, the normal illumination variations can still affect the result of change detection. In order to compensate for these variations, illumination adjustment was applied. By using illumination normalization, the pixel intensity values in one image are normalized to have the same mean and variance as those in another [12], i.e.,

$$\tilde{I}_2(i,j) = \frac{\sigma_1}{\sigma_2}\{I_2(i,j) - \mu_2\} + \mu_1$$

where $I_m(i,j)$ is the intensity of image i at position (i,j), $\tilde{I}_2(i,j)$ is the normalized second image intensity and μ_i, σ_i are the mean and standard deviation of the intensity values of I_m.



(a) (b) (c)

Figure 4.4: Illumination adjustment examples. (a) and (b) are consecutive images with illumination variation. (c) is generated from (b) with illumination normalization relative to (a). Illumination normalization makes pixel-wise change detection more robust.

As shown in Figure 4.4, there is apparent illumination difference between image (a) and (b). However, the illumination difference is considered harmless since there is no dramatic change on (b) except for illumination. Then, (c) is generated from (b) using illumination normalization based on statistics of (a) and change due to illumination is filtered out.

On the other hand, dramatic intensity change caused by illumination interruption should not be compensated or ignored. Such changes are detected before we apply illumination adjustment. Due to the restriction of the illumination device, a normal illumination difference would not exceed 50% of the original image intensity. By checking the image intensity statistics and comparing with the context, illumination interruption can be detected.

4.3 Change detection

Change detection is to find regions of change in images of the same scene taken at different times [12]. In our application, changed regions in consecutive images acquired by KESM are likely to be caused by imaging errors, which are our major interest. Various methods and applications of change detection exist in the literature as we saw in the related work. My main goal is to balance the detection accuracy and computational complexity in order to achieve automatic detection as well as real-time detection. In my method, a statistics-based method was employed.

To make the analysis simple, the difference of the image pair $(I_1(i,j), I_2(i,j))$ is expressed by the intermediate result, difference image $D(i,j)$. The basic question for change detection is whether or not significant change takes place at one pixel (i,j). Therefore, there are two classes in the difference image, no-change pixels and change pixels. In order to classify the two classes, we need knowledge of the two classes. Statistical models try to characterize the distribution of the two classes.

Those pixels which are better described by change class, are considered to be caused by real change.

The difference image could be described by two different models: (1) Noise model that assumes the difference image is from a no-change distribution. It assumes all the differences are solely generated by random noise in the case of no-change. (2) Mixture model takes into account both change and no-change distribution to be responsible for the difference image. Both of the models make reasonable assumptions describing the intensity distribution of the difference image.

4.3.1 Noise model

Noise model characterizes the no-change pixels. As a result, it makes an assumption that the intensity difference between two consecutive image are caused by noise alone in the absence of any change. Any difference that could not be described by random noise can be considered as actual change. Therefore by adopting a significance test, the method could decide whether the intensity difference is a change or not.

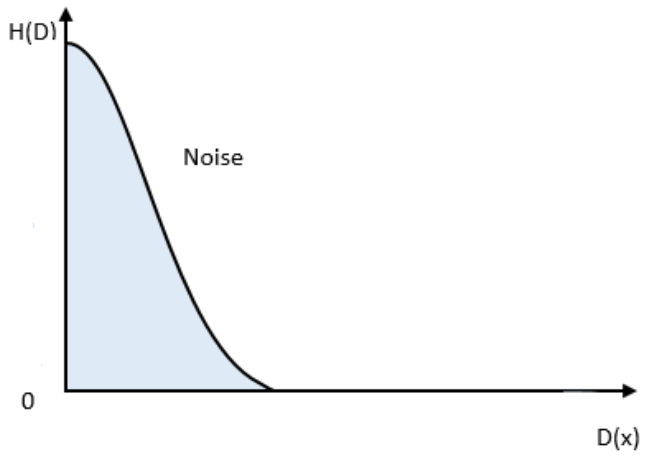


Figure 4.5: Noise distribution.

The above figure shows the intensity distribution of the difference image $D(i,j)$, solely caused by random noise. As shown in the figure, most pixels in the difference image are low intensity differences. Those with high intensity differences tend not to be caused by noise. In our case, a reasonable assumption was that the difference image $D(i,j)$ under noise could be modeled by a zero mean Normal distribution $N(0,\sigma^2)$. In order to classify the changed regions, I applied the three-sigma rule. In statistics, three-sigma rule states that nearly all values (99.73%) lie within three standard deviations of the mean in a normal distribution. All values outside of the three-sigma area were considered not to be solely caused by noise, which means they were caused by significant changes. This can be written as

$$B(i,j) = \begin{cases} 1 & : D(i,j) > 3\sigma \\ 0 & : \text{otherwise} \end{cases}$$

, where $B(i,j)$ is the binary image that marks change regions.

Parameter σ could be estimated offline from the unchanged regions in the previous image sequence. Using an online method [13], I estimated parameters by median value $\bar{\mu}$ and $\bar{\sigma}$ instead. Figure 4.7 shows an example of noise model.

The noise model made a reasonable assumption that difference in images caused by noise can be modeled by a Gaussian distribution. In addition, since the only parameter that need to be estimated were mean and variance of the difference image, the computational complexity was acceptable.

4.3.2 Mixture model

The mixture model characterizes both the no-change pixels and change pixels. In the difference image, there are two opposite classes, W_0 and W_1, no-change and change, respectively. Our goal was to classify each pixel into these two classes. A

reasonable assumption was that the difference intensity at pixel (i, j), $D(i, j)$, was caused by a combination of W_0 class and W_1 class. Therefore, the probability density function of the difference image $D(i, j)$ could be described as a mixture density distribution consisting of W_0 and W_1. The mixture density distribution $p(D)$ for the difference image can be written as:

$$p(D(i,j)) = p(D(i,j)|W_0)P(W_0) + p(D(i,j)|W_1)P(W_1)$$

Before we try to solve the classification problem, the first task is to estimate the likelihood functions $p(D(i, j)|W_0)$, $p(D(i, j)|W_1)$ and the prior probabilities $P(W_0)$, and $P(W_1)$. By modeling class W_0 and W_1 as two different Gaussian distributions, I was able to calculate the likelihood functions and prior probabilities. In order to define the Gaussian distribution, mean μ and standard deviation σ should be estimated. I adopted Bruzzone and Prieto's method [2] to estimate the means and standard deviation of the class conditional distribution $p(D(i, j)|W_k)$.

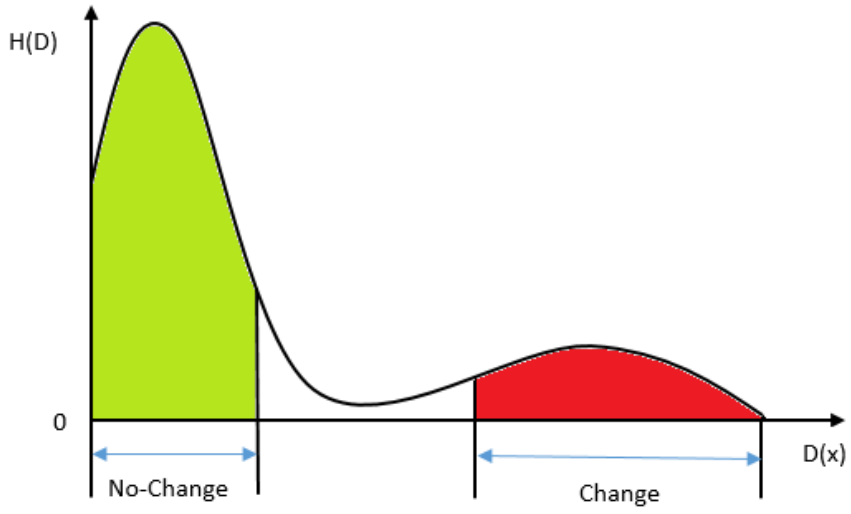


Figure 4.6: Histogram of difference image distribution.

Figure 4.6 describes the mixture distribution of the two classes. The two classes are model by a Gaussian distributions with different mean and variance. From the figure, no-change pixels tend to be in low intensity while the change regions are in higher intensity. In order to estimate the parameters, Expectation-Maximization (EM) algorithm was used.

EM algorithm is an iterative process for finding maximum likelihood (ML) or maximum a posteriori (MAP) estimates of parameters in statistical models. It can be computationally expensive. Its iterative step is described as,

$$p^{t+1}(W_k) = \frac{\sum_{(i,j)\in D} \frac{p^t(W_k)p^t(D(i,j)|W_k)}{p^t(D(i,j))}}{I \times J}$$

$$\mu_k^{t+1} = \frac{\sum_{(i,j)\in D} \frac{p^t(W_k)p^t(D(i,j)|W_k)}{p^t(D(i,j))} D(i,j)}{\sum_{(i,j)\in D} \frac{p^t(W_k)p^t(D(i,j)|W_k)}{p^t(D(i,j))}}$$

$$\sigma_k^{t+1} = \frac{\sum_{(i,j)\in D} \frac{p^t(W_k)p^t(D(i,j)|W_k)}{p^t(D(i,j))} [D(i,j) - \mu_k^t]^2}{\sum_{(i,j)\in D} \frac{p^t(W_k)p^t(D(i,j)|W_k)}{p^t(D(i,j))}}$$

where t and t+1 stand for current and next iteration respectively, $p(W_k)$ is the probability of a class W_k, and $p(D(i,j))$ is the probability of a specific intensity in the difference image.

From the expression, it was found that the algorithm would traverse through the whole image in each iteration, which was computationally expensive. In practice, I adapted an improved method by using histogram information to do the estimation, because pixels with same intensity at different location would have the same effect on the final calculation result. This reduced the computational complexity to a large constant number in theory and made the iteration speed up significantly. The

improved estimation equations are shown below:

$$p^{t+1}(W_k) = \frac{\sum\limits_{0 \le x \le 255} \frac{p^t(W_k)p^t(x|W_k)}{p^t(x)} H(x)}{I \times J}$$

$$\mu_k^{t+1} = \frac{\sum\limits_{0 \le x \le 255} \frac{p^t(W_k)p^t(x|W_k)}{p^t(x)} xH(x)}{\sum\limits_{0 \le x \le 255} \frac{p^t(W_k)p^t(x|W_k)}{p^t(x)}}$$

$$\sigma_k^{t+1} = \frac{\sum\limits_{0 \le x \le 255} \frac{p^t(W_k)p^t(x|W_k)}{p^t(x)} H(x)[x - \mu_k^t]^2}{\sum\limits_{0 \le x \le 255} \frac{p^t(W_k)p^t(x|W_k)}{p^t(x)}}$$

With the estimated parameters, classification could be done using Bayes decision rule. Let $W(i,j)$ denote the class at pixel (i,j), then $W(i,j)$ can be written as:

$$\begin{aligned} W(i,j) &= \arg\max_{W_i} \{P(W_i|D(i,j))\} \\ &= \arg\max_{W_i} \{P(W_i)P((D(i,j)|W_i))\} \end{aligned}$$

. Therefore, the generated binary image can be written as,

$$B(i,j) = \begin{cases} 1 & : P(W_1|D(i,j)) > P(W_0|D(i,j)) \\ 0 & : \text{otherwise} \end{cases}$$

.

Figure 4.7 shows the result generated by mixture model in a consecutive image pair that contained changed regions.

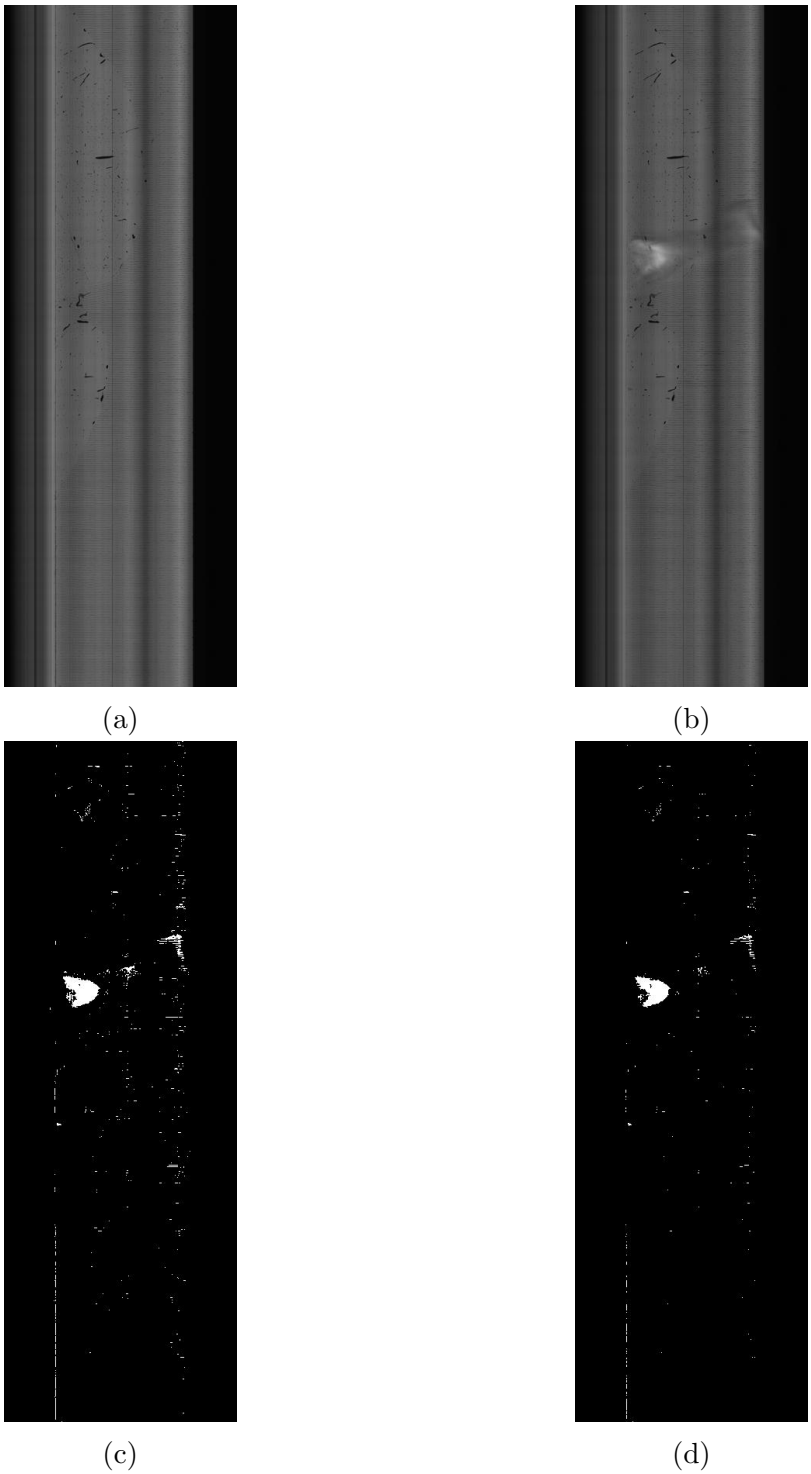


Figure 4.7: Change detection results from noise model and mixture distribution. (a) and (b) are consecutive images with obvious change regions. (c) and (d) show the highlighted change regions after using change detection algorithm. (c) is the change detection result generated using the noise model. (d) is the change detection result generated using the mixture model.

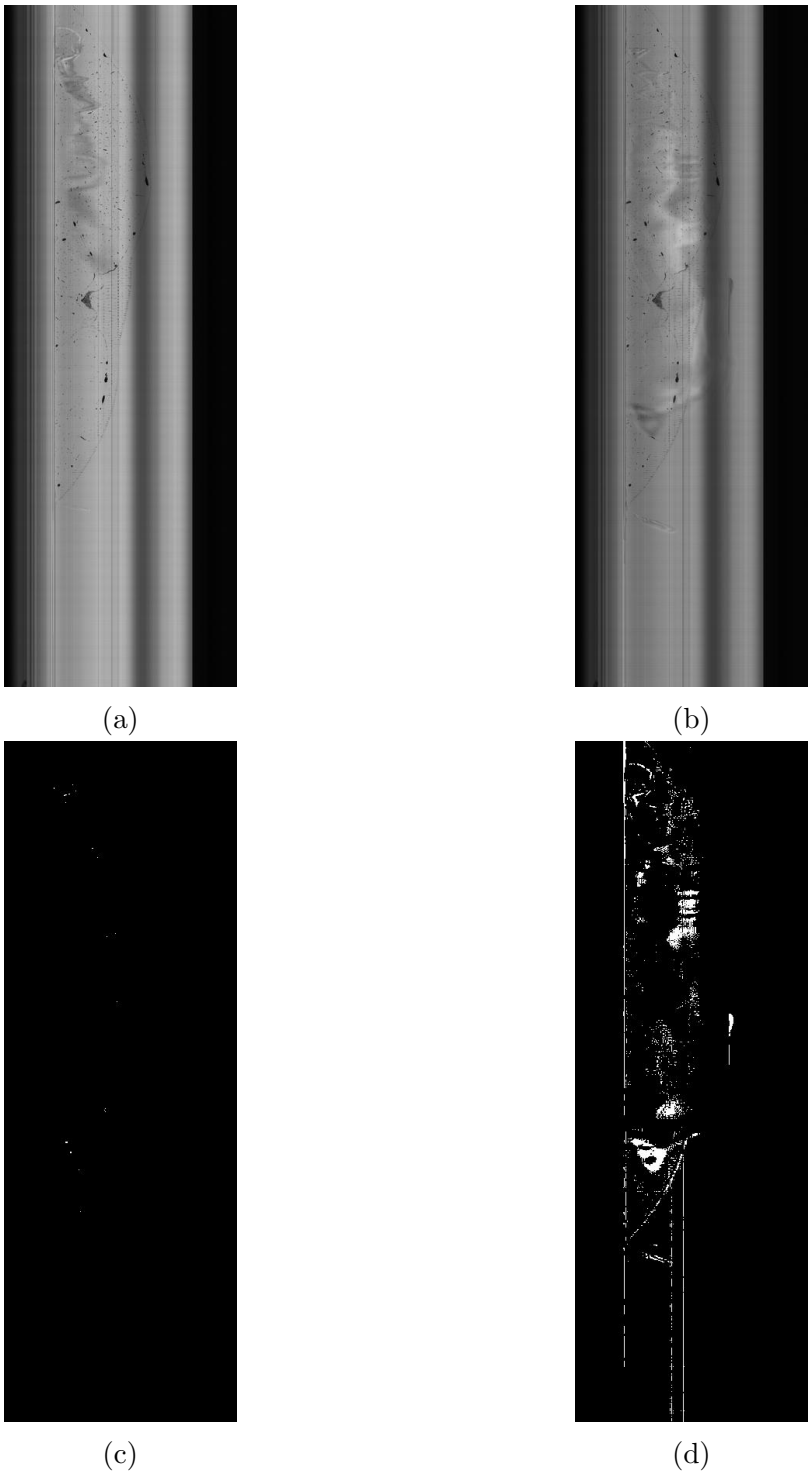


Figure 4.8: Change detection results with large change regions. (a) and (b) are consecutive images with large change regions. (c) is the change detection result generated using the noise model. (d) is the change detection result generated using the mixture model. Mixture model is performing better.

Mixture model was a stronger assumption about the distribution in the difference image. It can detect smaller changes as well as dramatic changes. Noise model performs worse when the change region is large. As shown in Figure 4.8, change regions are barely detected by noise model.

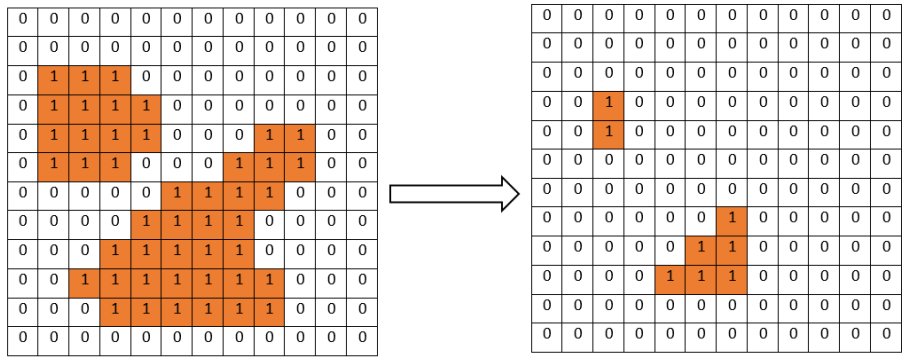
I implemented the mixture model in my method.

4.4 Postprocessing

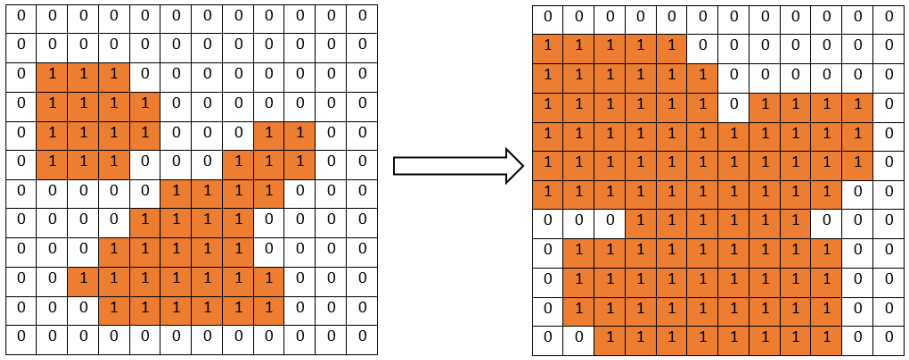
Postprocessing step took the result generated by the change detection method as input and then generated the final result by highlighting the change detected regions in the image. This was a necessary step because of the existence of noise. Noise such as isolated change regions would appear in the result of the change detection step, as we could see in Figure 4.7. Besides, considering change regions caused by real errors were likely to appear as a large connected region, those large regions were our major interest while the small size change regions are mostly due to image noise. Therefore, I applied postprocessing to remove isolated change regions and to filter out any change regions that are small in size. The postprocessing mainly consists two steps: (1) erosion and dilation to remove noise and (2) region filtering to filter out change regions that are small in size.

4.4.1 Erosion and dilation

In image processing, erosion and dilation are basic morphological operations. Erosion transforms a binary image to shrink by eroding the boundary pixels of an existing foreground object. On the contrary, dilation transforms a binary image to grow by adding the boundary pixels of an existing foreground object. The illustrations of erosion and dilation are shown in Figure 4.9.



(a)



(b)

Figure 4.9: Examples of erosion and dilation. (a) shows the result of erosion. Objects are shrinked. (b) shows the result of dilation. Objects are growing.

Erosion and dilation can be combined to find a specific component without image distortion. Opening is defined as erosion followed by dilation, which removes sharp peaks and thin connections with various structural elements (diamond, disk, rectangle, line, and square) to smooth out the contour. Closing is the opposite operation defined as dilation followed by erosion. As a result, closing links narrow breaks, fills long thin gulfs, and fills holes smaller than the structural elements.

In postprocessing, I applied opening to remove noise as well as isolated regions by erosion and dilation. An example of the result is shown in Figure 4.10 .

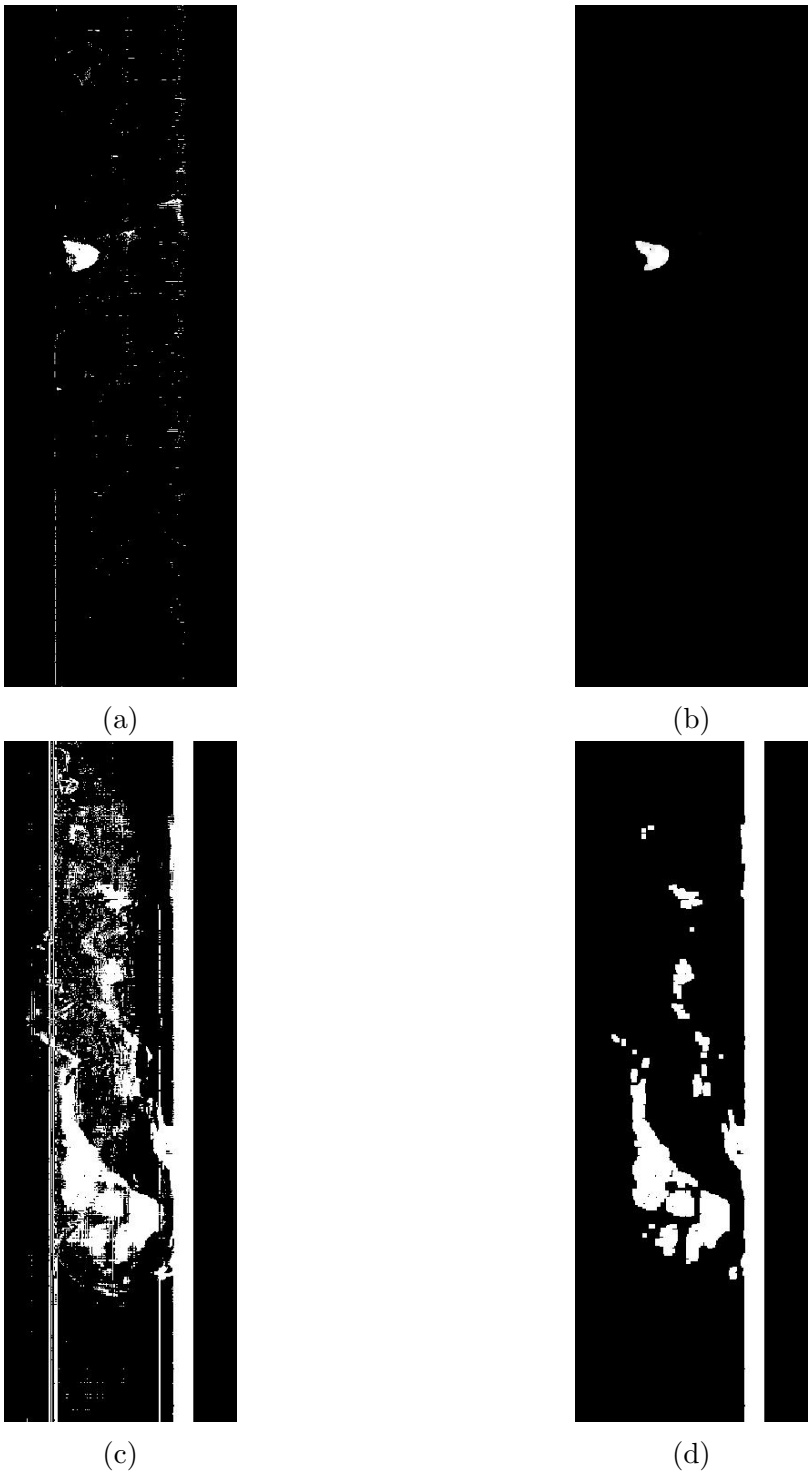


Figure 4.10: Results of openning. (a) and (c) are images before we apply openning. (b) and (d) are the result of openning. Most of the noise is removed.

4.4.2 Region filtering

Region filtering was a step to filter any changed region whose size was small and considered unimportant. By setting the size threshold, users were allowed to control the change tolerance on size, as large connected change regions were our major concern. By comparison, small change regions in the image were less likely to be caused by a real error.

In order to check the size of every connected change region, I used a traverse method based on depth-first-search. The basic idea was to go through the whole image to calculate each connected component's size. The algorithm would go to four directions (positive X, negative X, positive Y, and negative Y) in the image to check the connectivity. An example is shown in the Figure 4.11,

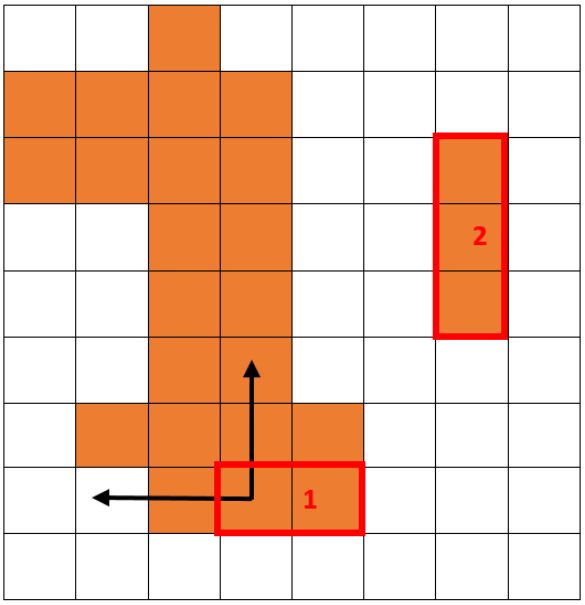


Figure 4.11: An example of region filter. Region 1 will expand in positive Y and negative X direction. Region 2 is fully explored and it will be removed because of the limited size.

While the algorithm explore the regions, it kept updating informations of the connected component, like size, width, height etc. By using the connected component information, smaller regions were filter out. Figure 4.12 shows an example:

In practice, the size of region filter is highly dependent on the tolerance of change regions. For example, in experiments which have a lot of knife chatters, it is likely to generate uncleared images. In such cases, we have higher tolerance of change regions considering knife chatters are not real errors. On the other hand, a lower region filter size would be better when clean images are generated. Generally, choosing a higher region filter size would result in lower false alarm rate but relatively higher missing rate. In our experiments, we chose the region filter size from 500 to 1300 empirically. The number is the minimum size of an error region that could attract human operators' attention.

4.5 Error detection

From the previous discussion, error detection based on change detection helps find change regions between two consecutive images, which are likely to be caused by imaging errors. However, in practice, it is not necessary to stop the cutting process every time an error is detected. There are generally two reasons: (1) Image artifacts and errors caused by floating tissues often disappear if they are removed by the pump in the following run. (2) Some image errors are tolerable as long as no data loss is caused. On the other hand, stopping the cutting process is also time-consuming especially when the error could have been removed by itself quickly. Therefore, in practice, we have to balance the different aspects.

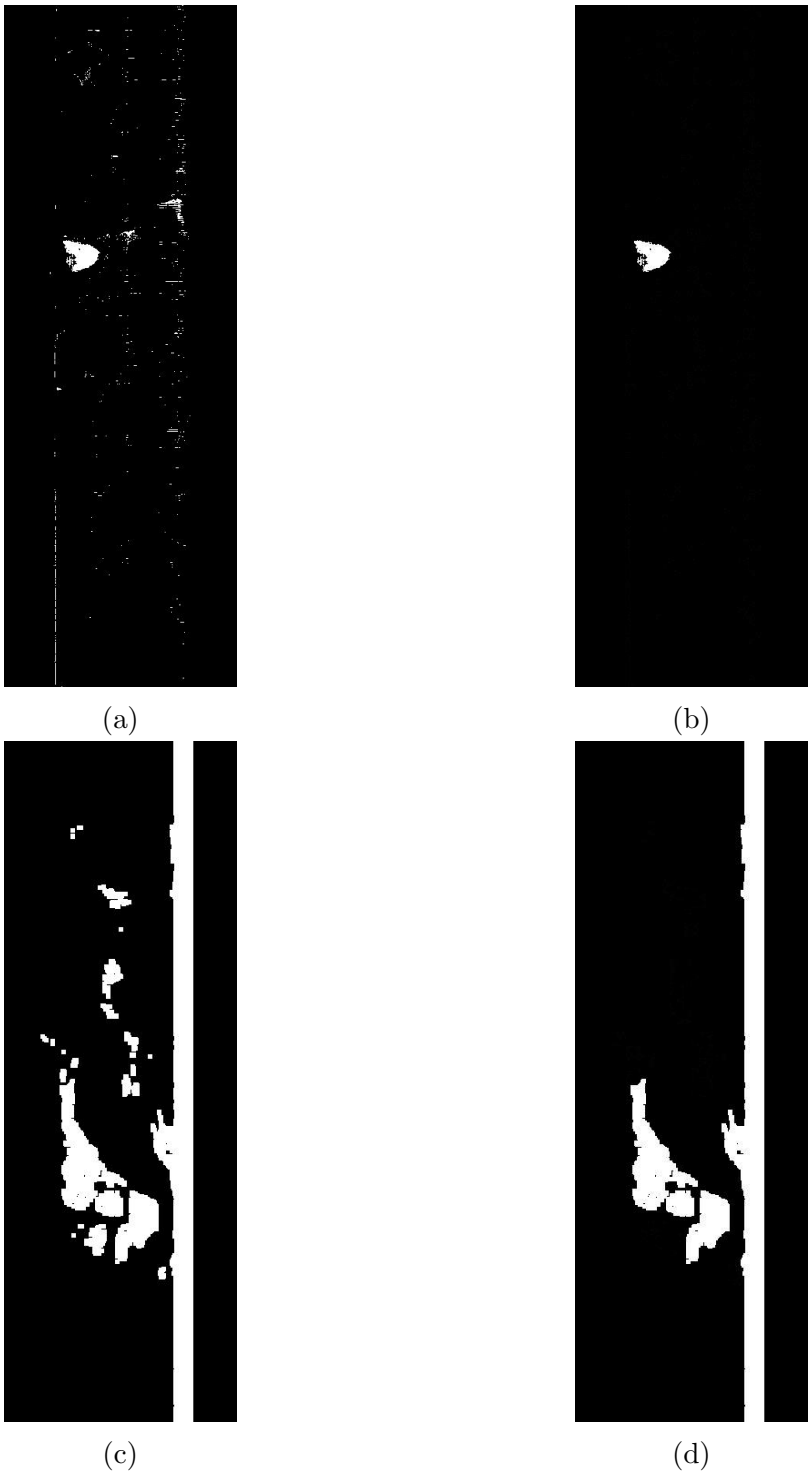


Figure 4.12: Results of applying region filter. (a) and (c) are images before we apply region filter. (b) and (d) are result images after applying region filter (size=1300). The small regions are removed.

In order to solve the problem, I developed an error detection strategy to enhance the robustness of the system. There are three different actions the strategy would take based on the severity of the problem: (1) recording the error, (2) reporting to the operators, and (3) stopping the cutting process. Recording the error is easy to understand. Whenever an error is detected, it is recorded in the change detection log. This helps us to locate the error and remove the unnecessary images when we are processing the acquired images. In the next part I will discuss the strategy and cases when we need to inform the operators and even stop the cutting.

4.5.1 Continuing errors

In most cases, one-time imaging errors are likely to disappear soon, so we do not need to pay too much attention to them. However, continuing errors are more likely to be caused by a severe obstruction in the light path, which should be informed to the operators. What follows is a detailed analysis of continuing errors.

Looking into one image pair is not sufficient to know if continuing errors take place. The basic idea is to look at the image sequence and change detection history. By combining the change detection results, we can make a more confident decision. To be specific, change detection results include two different types of knowledge: (1) change detection history in the same column (column is introduced in background), which is the local knowledge and (2) combinational results in different columns, which provides global knowledge. Global knowledge is also important because severe errors which has appeared in one column are likely to spread to the next column.

Considering the change detection histories, there are generally three important cases.

(1) Isolated change in one column. Isolated change means a detected error which appears in one image and disappears immediately in the next image. When the

change is first detected, my method continues to check the next acquired image to see if the change has disappeared. To be precise, let I_1, I_2 and I_3 be three consecutive images. I_1 is a normal image. A change region caused by floating tissues appeared in I_2, which can be detected by comparing between I_1 and I_2. Then the change disappeared in I_3 and this was detected as well. In such case, comparing I_1 and I_3 directly would allow us to know whether or not the error caused by floating tissue has disappeared. In this way, comparing the newly acquired image with multiple previous images would allow us to find isolated change. In practice, isolated change is the most common case so that only recording the error is sufficient. We don't need to report to operators nor to stop cutting in such cases. By comparison, there is a different case when a constant change first occurred, causing only one change detection in the image sequence. This happens not frequently. Looking into one column is not sufficient to confirm the error. It is necessary to check the change detection results in the neighboring columns. This will be also discussed in case (3).

(2) Consecutive changes in one column. Other than isolated change in the same column, consecutive changes (more than two) means the acquired images change continuously. By checking the image sequence and change detection history, we can find consecutive changes in the same column. In most cases, the consecutive changes are likely to be caused by floating tissues which are not removed in time. In such cases, it is highly recommended for the operators to check the system status. Therefore, it is necessary to report to the operators.

Note that isolated change will also generate two consecutive change detections (one for appearance and the other for disappearance). However, we should not count those changes as consecutive changes. My method would keep comparing the newly acquired images with multiple previously images to eliminate such cases.

(3) Continuing changes in two columns. This is the case when an error first

appeared in one column, causing a change detection, and then spread to the next column. This type of error is likely to be caused by a constant error. Although it is not very frequent, we need to take care. By combining the detection results in the neighboring columns, we are more confident to detect such a potential error spread. In this case, it is also necessary to inform the operators.

In order to find continuing errors and report to the operators, I developed a finite-state machine to implement the strategy. There are seven statuses in total for the machine, representing different change detection cases. Two types of actions are involved: (1) change detection between newly acquired image and the immediately preceding image; (2) change detection between the newly acquired image and the previous images in the same column. In addition, for checking status in different columns, it is necessary to check if a column shift has taken place. The finite-state machine is shown below:

The implementation of the finite-state machine allowed us to detect continuing errors, including (1) consecutive changes in the same column and (2) continuing errors in different columns. Besides, isolated change and change detection event caused by false alarm are ignored. When continuing errors are detected, it is still not sufficiently convincing to stop the cutting process. Therefore, the method will simply inform the operators.

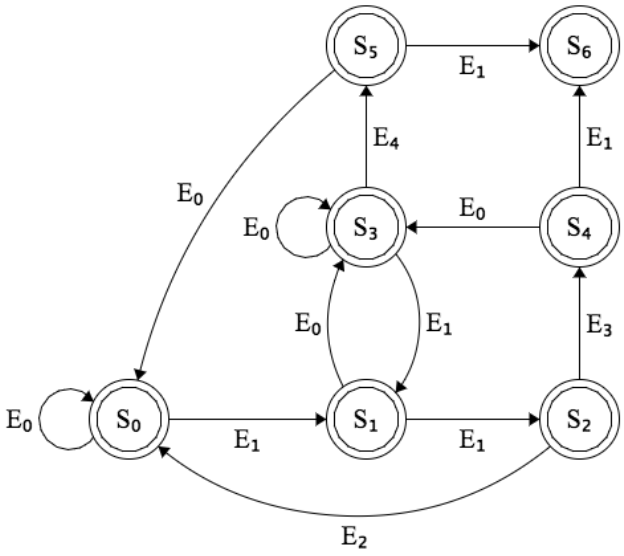


Figure 4.13: Error detection finite-state machine. The machine has two types of actions: (1) change detection between newly acquired image and the immediately preceding image; (2) change detection between the newly acquired image and the previous images in the same column. There are totally five events: (1) E_0 is under action (1), resulting in no-change, (2) E_1 is under action (1), resulting in change detected, (3) E_2 is under action (2), resulting in no-change, (4) E_3 is under action (2), resulting in change detected, and (5) E_4 is column shift. There are seven statuses representing different change detection cases: (1) S_0 is the start status, indicating no error is detected and everything is working fine, (2) S_1 is the case when change is detected only in the newly acquired image. The system was working fine previously. (3) S_2 is the case when two consecutive changes are detected in the newly acquired images. It can be isolated change or a actual continuous change. We have to check with the previous images in this case. (4) S_3 is the case when change detection is followed by no-change detected in the most recent image, (5) S_4 is the case when two consecutive changes are detected and it is not an isolated change (error does not disappear), (6) S_5 is the case when column shift takes place, (7) S_6 is the case when at least three consecutive changes or error spread are detected. We need to report the errors in S_6.

4.5.2 Cutting termination

Forced cutting-session termination happens when severe errors are detected and serious data loss would be caused if the cutting is not stopped immediately. We have to stop the cutting process so that further data loss can be prevented. There are basically three different reasons to stop the cutting.

(1) Floating tissues could clog the pump circulation, generating continuing artifacts and errors on the newly acquired images. It is necessary to stop cutting in this critical case. Since the pump is not clogged very often, we don't have enough data to analyze. Instead, I manually turned off the pump in the experiment to mimic the status of pump clogging. The results showed that errors appear on more than 80% of the newly acquired images in such cases. Therefore, by checking the change detection history, my method could find out how frequently errors are detected. If the frequency reaches a threshold set by the users, the method would decide to stop the cutting process. In the experiments, we set up the threshold to 80% empirically.

(2) Illumination Interruption. By checking the illumination statistics of the newly acquired image and comparing with the context, the method could detect interruption in illumination. In most cases, the error images become very darker than normal value. It is necessary to stop the cutting process immediately.

(3) Disk malfunction and disk full errors are different from image errors because they cannot be detected directly through image analysis. A simple workaround is to check the disk status every time before the KESM starts a cutting. Users are allowed to set up the minimum free space to be warned. This feature is also integrated in my program.

To summarize, the error detection method would inform operators when continuing errors are detected. In addition, it will also stop the cutting process when it is in

a critical situation. In this way, the method enhanced the robustness of the system and preserved the integrity of the data.

5. IMPLEMENTATION AND RESULTS

In the previous chapter, a detailed description of KESM error detection method was presented. In this chapter, the implementation of the method will be presented at the beginning. Then I will present the results of error detection in section 2. In section 3, the method will be evaluated.

5.1 Implementation

The implementation of image error detection in KESM integrates several complicated components. The workflow in the error detection and interaction with operators as well as existing KESM control system must be properly executed. It is necessary to develop a program to achieve image error detection.

There are four requirements for the error detection program.

(1) Since the main purpose of the program is error detection, the first requirement is to implement the error detection method that I described in the previous chapter. This mainly includes image downsampling and change detection.

(2) The program must display and records the error detection results properly. The program was designed to utilize a graphic user interface (GUI) to interact with users. Users should be allowed to see the results and configure the detection parameters.

(3) As one component of the whole KESM control system, the program was designed to integrate with the existing system. Interactions with the existing control system, including loading newly cut images and handshaking with stage controller, are required.

(4) The program is required to inform operators and even terminate the KESM cutting when necessary to protect the data integrity.

I developed a KESM image error detection program to fulfill the requirements above. It was developed in Python, which provides useful packages and libraries, including image processing library, system functions, and graphic user interface library. Besides, the program was developed as a lightweight module, which makes it easy to integrate with the existing KESM control software.

The architecture of the program is shown in Figure 5.1

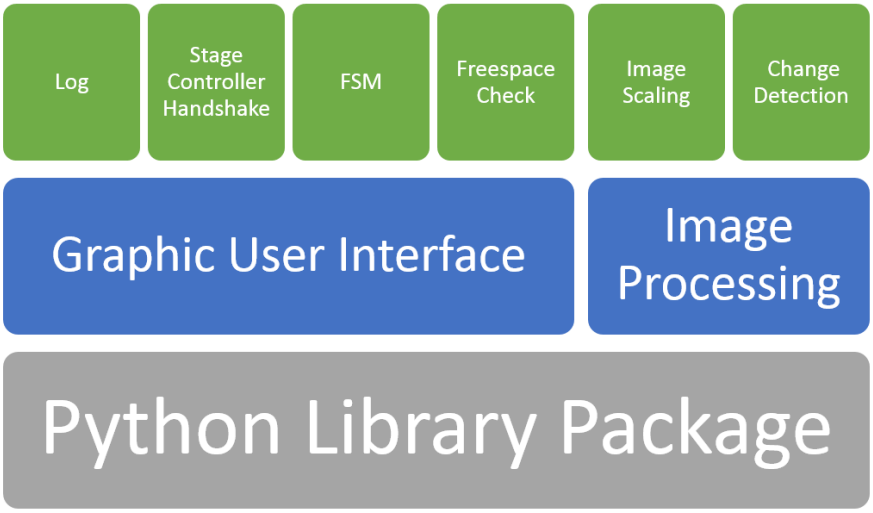
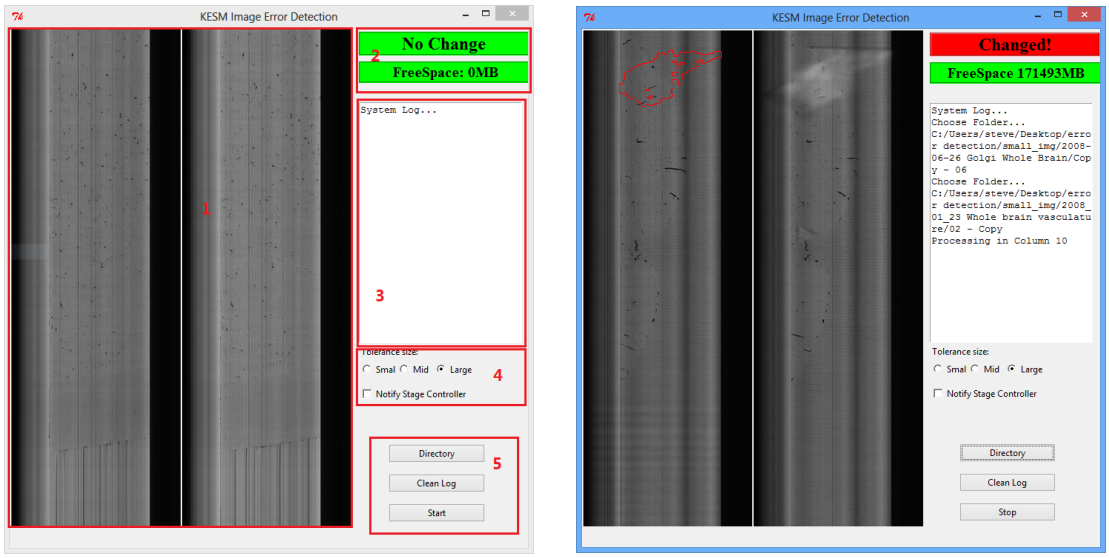


Figure 5.1: The architecture of KESM image error detection system.

According to the requirements, the architecture of the program can be described mainly as four components:

(1) The image processing module takes care of the error detection and image downsampling. It was built on Python Pillow, a python image processing library which provides basic image operations, such as image loading, resizing, erosion and dilation. In the program, image processing module is called from upper layer to generate the error detection results.

(2) A logging module was specially developed to record the error detection results. Important information is shown on the user interface and recorded in a log file. To display and interact with users, a GUI was built based on TKinter, a standard python GUI package. Users can see the comparison results of error detection and any other monitoring status, such free space in the disk. In addition, users are allowed to interact with the interface by configuring different parameters. Configuration options include whether to inform operators or not and whether to stop the cutting when necessary or not, and the region filter size etc. An example of the interface is shown in Figure 5.2.



(a) (b)

Figure 5.2: (a) GUI of the KESM error detection system. Panel 1 displays the consecutive images and error detection results. Panel 2 is the system status indicating whether an error is detected and the volume free space. Panel 3 is the system log. Panel 4 provides configuration options to the users. Panel 5 includes buttons to start and stop the program, to choose the directory, and to clean the log. (b) Screenshot of the program when a change is detected.

(3) Stage controller handshake module was designed to interact with the existing KESM control system. The module finds newly acquired images and interacts with the stage controller program.

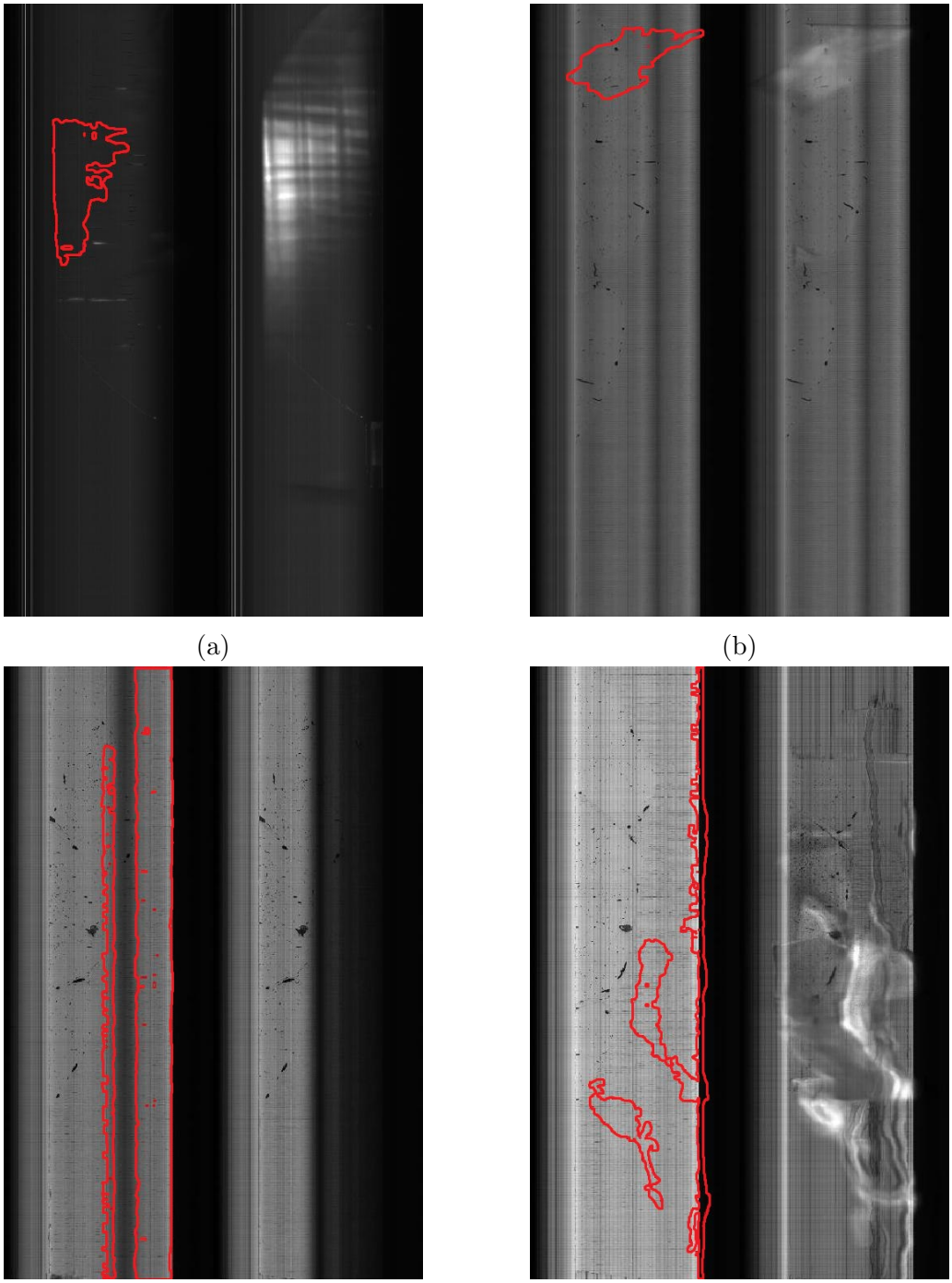
(4) An finite-state machine (FSM) module was developed to inform operators and even stop the KESM cutting process if necessary. Operators can add their email address to the list so that they will be informed through email when an error was detected. In addition, the module decides to terminate the cutting process when it is necessary.

The program works as follows. In the beginning, users can start the error detection program from the existing stage controller program or they can choose the directory where the KESM project is located. After the program is started, it keeps checking every image folder continuously to monitor if a new image is acquired. When a new image is found, the program launches the error detection module to compare it with the previous images. Results with highlighted regions will be displayed on GUI while important information is recorded in the log file. The FSM module will check to see if it is necessary to inform operators by email or even stop the cutting process. Meanwhile, the program keeps checking the free space on the disk to make sure enough free space is available.

5.2 Results of change detection

By using the error detection program, error regions in the images acquired by KESM can be detected. Figure 5.3 shows some selected results.

In Figure 5.3, regions highlighted by red lines were detected as change region. (a) is an example of imaging errors caused by tissue rolling. When the knife was cutting the tissue, the sectioned part was distorted badly. In the result, a large white region was detected. In (b), a small piece of floating tissue resulted in the error region. (c)



(a) (b)

(c) (d)

Figure 5.3: Selected error detection results.

shows a dramatic illumination change in a rectangle region. In (d), an irregularly shaped error region was detected. The error was possibly caused by cutting fault.

5.3 Evaluation

The error detection method was evaluated in terms of two aspects: accuracy and speed. Accuracy was evaluated to show the effectiveness of the detection method. In order to evaluate accuracy, I manually labeled three groups of data as ground truth and compared them with the result generated by the proposed method. Then speed was evaluated to measure the execution time of the program to test if real-time detection is possible.

5.3.1 Accuracy

The idea of accuracy evaluation is to compare the results with ground truth. Here 'ground truth' means a comprehensive data set which includes the correct answers the program should produce. Therefore, I manually labeled three groups of image data acquired by KESM. All three groups of data included around 1,000 images. The first two groups were from existing data sets, acquired in 2008 and 2010, respectively. The third group of data was from the zebrafish experiment which was conducted in 2014. All the images in the data set were labeled either as error image or as clean image. The format of the image was gray-scale with a dimension of 4096×12000. Detailed information about the data set is presented in Table 5.1:

Table 5.1: Data set for evaluation of change detection

Data set	Creation time	Number of images	Images with error or artifacts
1	Jan, 2008	1024	85
2	Feb, 2010	1018	20
3	May, 2014	959	179

After establishing the ground truth, I adopted standard measurements for comparing the results. The following quantities are used in the evaluation: (1) True Positive (TP): image with errors correctly identified as error image; (2) false positive (FP): image without error incorrectly identified as error image; (3) True negative (TN): image without error correctly identified as clean image; (4) False negative (FN): image with errors incorrectly identified as clean image.

Based on the quantities, accuracy is first evaluated on the Percentage Correct Classification(PCC) which is expressed as following:

$$PCC = \frac{TP + TN}{TP + FP + TN + FN}$$

PCC evaluates the accuracy by calculating the proportion of correct detection. In our case, one disadvantage of PCC is that images with errors and artifacts are only a small portion of the entire data set. A large TN usually generates a good PCC result, which only evaluates on the overall accuracy. Therefore, I also adopted F1-measure to calculate more precise evaluation on images with errors. F1-measure is the harmonic score of precision and recall. Precision is the proportion of correctly error detected images among all the error detected images. Recall is defined as the proportion of correctly error detected images among all the images with true error.

They can be expressed as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2TP}{2TP + FP + FN}$$

First of all, the accuracy of error detection is evaluated on the two models and different region filer sizes. Data set 1 was split into three groups to evaluate. The following tables show the result.

Table 5.2: Accuracy on different models

Data group in set 1	Model	PCC	Precision	Recall	F1
1	Noise	98.09%	93.75%	88.23%	90.91%
1	Mixture	98.41%	96.77%	88.23%	92.31%
2	Noise	95.15%	67.74%	80.76%	73.68%
2	Mixture	96.76%	83.33%	76.23%	80.00%
3	Noise	97.74%	83.33%	80.00%	81.63%
3	Mixture	98.74%	95.45%	84.00%	89.36%

Table 5.2 shows the comparison between noise model and mixture model with region filer size fixed at 900. From the table, it shows that mixture model is generally performing better than noise model by having a higher PCC and F1 score, which confirms theoretical analysis in the previous chapter. In the final implementation of error detection program, I adopted the mixture model.

Table 5.3: Accuracy on different region filter sizes

Data group in set 1	Region filter size	PCC	Precision	Recall	F1
1	500	98.41%	91.67%	94.28%	92.95%
1	900	98.41%	96.77%	88.23%	92.30%
1	1300	97.14%	96.42%	77.14%	85.71%
2	500	94.35%	60.00%	92.30%	72.72%
2	900	96.86%	83.33%	76.23%	80.00%
2	1300	96.23%	84.00%	72.41%	77.78%
3	500	97.49%	74.19%	92.00%	82.14%
3	900	98.74%	95.45%	84.00%	89.36%
3	1300	98.74%	95.45%	84.00%	89.36%

Table 5.3 shows the accuracy of mixture model different region filter sizes. It indicates that the detection result also highly depends on the region filter size. When the filter size increases, precision gets higher and recall becomes lower. In clean image sequences, small filter size is acceptable while in images with more noise, a large filter size is better. Therefore in practice, users should adjust the region filter size to fit the specific need.

Finally, the overall accuracy was evaluated on three different data sets. Table 5.4 shows the final result.

Table 5.4: Evaluation result of error detection

Data set	PCC	Precision	Recall	F1
1	98.24%	94.67%	83.53%	88.75%
2	99.51%	94.11%	80.00%	86.48%
3	94.79%	85.25%	87.15%	86.19%

To sum up, the error detection method is able to automatically detect imaging error with PCC around 95% and F1 score around 86%.

5.3.2 Speed

The other important requirement for the program is real-time detection. It means that the method should finish detection within the time period of two consecutive cutting, which is around 7 seconds. In order to achieve this goal, the implementation of the program was optimized. I conducted speed evaluations in different machines and different experiments. Table 5.5 shows the result.

Table 5.5: Speed evaluation

Machine	Experiment	Average(s)	Minimum(s)	Maximum(s)
1	1	1.819	1.265	3.828
1	2	1.817	1.233	3.202
1	3	1.873	1.250	3.578
2	4	0.304	0.241	0.676
2	5	0.311	0.259	0.680
2	6	0.334	0.233	0.672

In the table, machine 1 is the aging KESM server (Dell PowerEdge 2800, 3.8 GHz Dual Intel Xeon, 6GB DDR2, Windows Server 2003) and machine 2 is a modern

laptop (Apple Macbook Pro, 2.5 GHz Intel Core i5, 4GB DDR3, OSX 10.9.3). The speed evaluation result shows that the program is able to finish a detection routine in less than two seconds on average, which is sufficient to achieve real-time detection. If the KESM server is updated in the future, the error detection will be greatly improved.

6. DISCUSSION

The new KESM image error detection algorithm and its implementation enables the detection of imaging errors in real-time. It will help enhance the robustness of the system and to reduce the data loss. In this chapter, the contributions of the KESM image error detection, its limitations, and future work will be discussed.

6.1 Contribution

In order to solve the imaging error problem in KESM and to enhance the robustness, I designed and implemented the KESM image error detection method. The major contribution of my research is the image error detection method: (1) By achieving real-time error detection, the method detects errors and reduces data loss, enhancing the robustness of KESM imaging. (2) The method largely reduces the burden of human operators through automatic error detection. Human operators will be notified when an error is detected, and the cutting process can be terminated when it is in a critical situation.

6.2 Limitation and future work

There are several open issues: (1) limitation of parameter configuration in change detection algorithm, and (2) lack of an advanced method to decide critical situation when cutting termination is necessary. I will discuss these open issues and the corresponding future work.

6.2.1 Limitation of paramter configuration in change detection

In my method, one parameter in change detection algorithm needs to be set up by users. By setting up the parameter region filter size properly, we can achieve a better result. The reason behind this is that the tolerance of error size varies from one

experiment to another. In the experiment which has a lot of knife chatter, generating unclear images, a large region filter size would be better. In our experiments, we set up the filter size empirically. However, the manual parameter configuration is still limited.

In the future, I will work on improvements of the parameter configuration. The pixel-wise change detection algorithm is the main reason why the parameter need to be set up manually. I will explore other methods that do not need parameter settings. An alternative approach is to decide the filter size by using data training methods. In such a case, we need more data to support the methods.

6.2.2 Limitation on cutting termination

In critical situations, we need to stop the cutting process so that further data loss is avoided. In the proposed method, a critical situation is defined as a case where several consecutive errors appear or errors appear very frequently. This kind of situation can be forcefully reproduced by cutting off the pump manually. In practice, critical situations happen rarely and we do not have enough data to characterize the real cases. Moreover, although stopping the cutting often helps reduce data loss, it is also time consuming that one of the goals of KESM is high-throughput imaging. This poses a tradeoff when defining a critical situation. In the proposed method, we determined empirically when to stop cutting. A more precise method is required.

In the future, I will develop a more precise method to decide the situation when we need to actually stop cutting. Other than just looking into the frequency of errors, there should be an error pattern which indicates a critical situation. This will help balance between reduction of data loss and the extra time needed after cutting termination.

6.3 Conclusion

The aim of this research was to develop a method to solve the imaging error issues in KESM. I developed a real-time image error detection method to complement the existing human intervention workaround. The method detects errors automatically, reports to the human operators, and stops the cutting process when it is necessary. Using an image change detection algorithm, the method detects imaging error with 86% accuracy. The method also runs in real-time. High accuracy and real-time operation allows the method to greatly reduce data loss in KESM imaging. Moreover, through automatic error detection, human operators now have less burden during the experiments. In summary, the proposed real-time image error detection method for KESM helps enhance the robustness of the system.

REFERENCES

[1] M. Brocke. "Statistical image sequence processing for temporal change detection." *Pattern Recognition. Springer Berlin Heidelberg*, 2002. 215-223.

[2] F. Bruzzone and D. F. Prieto. "An adaptive semiparametric and context-based approach to unsupervised change detection in multitemporal remote-sensing images." *Image Processing, IEEE Transactions on* 11.4 (2002): 452-466.

[3] Y. Choe, L. C. Abbott, D. Han, P.-S. Huang, J. Keyser, J. Kwon, D. Mayerich, Z. Melek, and B. H. McCormick. "Knife-edge scanning microscopy: high-throughput imaging and analysis of massive volumes of biological microstructures." *High-Throughput Image Reconstruction and Analysis: Intelligent Microscopy Applications* (2008): 11-37.

[4] Y. Choe, D. Mayerich, J. Kwon, D. E. Miller, J. R. Chung, C. Sung, and L. C. Abbott. "Knife-edge scanning microscopy for connectomics research." *Neural Networks (IJCNN), The 2011 International Joint Conference on.* IEEE, 2011.

[5] J. R. Chung, C. Sung, D. Mayerich, J. Kwon, D. E. Miller, T. Huffman, J. Keyser, L. C. Abbott, and Y. Choe. "Multi-scale exploration of mouse brain microstructures using the knife-edge scanning microscope brain atlas." *Frontiers in Neuroinformatics* 5 (2011).

[6] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. "Detecting moving objects, ghosts, and shadows in video streams." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25.10 (2003): 1337-1342.

[7] C.-L. Huang and B.-Y. Liao. "A robust scene-change detection method for video segmentation." *Circuits and Systems for Video Technology, IEEE Transactions*

on 11.12 (2001): 1281-1288.

[8] J. Kwon, D. Mayerich, Y. Choe, and B. H. McCormick. "Automated lateral sectioning for knife-edge scanning microscopy." *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on.* IEEE, 2008.

[9] D. Mayerich, L. C. Abbott, and B. H. McCormick. "Knife-edge scanning microscopy for imaging and reconstruction of three-dimensional anatomical structures of the mouse brain." *Journal of microscopy*, 231.1 (2008): 134-143

[10] B. H. McCormick. "Development of the brain tissue scanner." *Brain Networks Lab Technical Report* (2002).

[11] J. Meng, Y. Juan, and S.-F. Chang. "Scene change detection in an MPEG-compressed video sequence." *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology. International Society for Optics and Photonics*, 1995.

[12] R. J. Radke, S. Andra, O. Al-Kofahi and B. Roysam. "Image change detection algorithms: a systematic survey." *Image Processing, IEEE Transactions on* 14.3 (2005): 294-307.

[13] P. L. Rosin. "Thresholding for change detection." *Computer Vision, 1998. Sixth International Conference on.* IEEE, 1998.

[14] Y.-L. Tian, M. Lu, and A. Hampapur. "Robust and efficient foreground analysis for real-time video surveillance." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1.* IEEE, 2005.

[15] V. Walter. "Object-based classification of remote sensing data for change detection." *ISPRS Journal of Photogrammetry and Remote Sensing* 58.3 (2004): 225-238.