OBJECT-BASED BIOIMAGE MATCHING ON IPAD

by

CHANIL PARK

(Under the Direction of Tianming Liu)

ABSTRACT

The object-based image matching algorithms have been widely used in the image processing and computer vision fields. A variety of applications based on image matching algorithms have been recently developed for object recognition, 3D modeling, video tracking, and biomedical informatics. One prominent example of image matching features is the Scale Invariant Feature Transform (SIFT) scheme. In this thesis work, we initially investigated using SIFT algorithms to identify and match objects in biomedical images based on the recently released iPad 2 platform. The major methodological contribution of this work is leveraging the convenient user interface and ubiquitous Internet connection on iPad 2 for interactive delineation, segmentation, representation, matching and retrieval of biomedical images. With these technologies, this thesis showcased examples of performing reliable image matching from different views of an object in the applications of semantic image search for biomedical informatics.

INDEX WORDS: Object-based image matching, iPad, Image segmentation, Scale Invariant Feature Transform (SIFT), Biomedical Informatics, Image Features.

OBJECT-BASED BIOIMAGE MATCHING ON IPAD

by

CHANIL PARK

B.S., University of Suwon, South Korea, 2003

M.S., University of Suwon, South Korea, 2005

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2012

OBJECT-BASED BIOIMAGE MATCHING ON IPAD

by

CHANIL PARK

Major Professor: Tianming Liu

Committee: Kang Li
Krzysztof J. Kochut

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2012

DEDICATION

To my mother,

When the day is hard,

Always support to us,

When the day is enjoy,

Always around us,

Anytime always say thanks to her,

We never make it without her encouragement and sacrifice.

ACKNOWLEDGEMENTS

Thanks to Dr. Tianming Liu for his excellent teaching and devotion on my research. My thesis would not be done without his advice and creative ideas. I would also like to thank Dr. Liu's previous student Mr. Xin Li and Dr. Liu's collaborator Dr. Junwei Han who provided the initial codes for this thesis project.

I would also like to thank Dr. Kang Li for his support and keen advice. I also would like to thank Dr. Krzysztof J. Kochut for his encouragement and advisement.

Lastly I have to say thanks to my friends. They always encourage and support me.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

Many image matching algorithms have been developed in the computer vision and image processing fields recently. As a result, there are a variety of applications based on image matching algorithms that have played important roles in computer vision and other domains. These applications include image stitching, object recognition, image fusion, image registration, motion tracking and gesture recognition [12, 13]. For instance, image comparison and matching have been widely used in methods such as geometrical image transformations, point-based method, matching and feature detection methods [2, 14, 15].

- Geometrical image transformations: It performs various geometrical transformations of images. This transformation deforms the pixel grid, and maps it to the destination image in which the image content is not changed. These methods can be used to align images that were taken at different times, to correct images for distortions, and to correct effects of orientations.
- Point-based method: This identifies point pairs for a given view, and then image registration aligns the points.
- Feature matching: It detects specific attributes between the features detected in the query image and those detected in the reference image. There are many feature descriptors and similarity measures for the features.

- Feature detection method: It detects a specific feature of image information such as lines, angle, edges, and corners, and makes local decision at every image point. These features are typically represented by their image point representatives.

Among many image matching methods and features [3, 4], the scale invariant feature transform (SIFT) scheme has been effective and widely used in the field [5]. The scale invariant feature transform is a well-known invariant local feature method. The SIFT algorithm aims to detect distinctive, invariant image feature points, which can be easily matched between images. These methods combine invariance to rotation, scale, and affine transformation that could be reliably matched.

Notably, Rob Hess implemented in a C version of the SIFT algorithm based on the Lowe's paper [5]. This version used the popular open-source computer vision library OpenCV [6], which is the first open-source version of the SIFT algorithm and has been rapidly used in the computer vision and image processing fields. The open-source SIFT library is available at http://robwhess.github.com/opensift/ and includes several functions for computing SIFT features in images, matching SIFT features by using KD-tree, and computing geometrical image transformations from feature matching by using RANSAC [16].

This library also includes functionalities to import and work with image features from both SIFT detectors and Oxford VGG's affine covariant feature detectors. The open-source library [17] currently provides two versions based on both Linux and Windows (VC++). Therefore, this thesis describes an implementation and design of software components using the open-source library to compare key-points between a query image and a reference image on the iPad 2.

Recently, digital image acquisition devices have been developed and distributed rapidly, driving the growing needs for automatic image matching and recognition. For example, we can assume that patients with diabetes need to control food intake, which is very important to their health. They also need to know the food nutrition components and calories before eating them. Obese people also need to check food nutrients and calories before consumption. In the biomedical field [18], biologists need to analyze bioimage dataset and image matching software can help them to compare biomedical image dataset in solving biomedical problems such as cell segmentation and annotation of biological objects. This software should be easy to use for these average biologists and the GUIs should be simple and user-friendly.

Recently, the Apple's iPad [25] designed for small tablet and high resolution screen has powerful graphic capability, e.g., a quad-core graphics processor. This device will be perfectly matched with human interaction for image matching algorithms. One of major advantages on iPad is the multi-touch graphical user interface which provides the recognition of the presence of two or more points of contact with the surface when a finger or object touches the surface. The iPad also supports Wi-Fi and 3G wireless connection that can be used to access local networks and the Internet in anytime, anywhere.

With the iPad, image matching applications are emerging, which is increasingly an attractive research direction. In addition, iPad has many remarkable features for image matching application. Firstly, the iPad has camera: 5-mega-pixel unit with 1080 HD video recording up to 30 frames per second with audio. It has autofocus, face detection and image stabilization. It can zoom in and zoom out, turn on the grid, set the exposure, lock the exposure, and take a screenshot. The iPad camera is typically good enough for photography and videography. Using this image, users can match input image with reference image. Second, iPad is designed for users

with user-friendly interface. The touch screen display uses state-of-the-art multi-touch technology, which is made simple to use. The finger trajectory can be recorded when a finger touches and moves on the display surface. This technology can help users to perform image segmentation and delineation. Finally, the iPad supports the Wi-Fi network and 3G wireless that is able to connect the Internet. Therefore, the iPad's connectivity contributes to the image matching application with the ability of sending the computing-interactive tasks to the back-end servers or cloud services.

In the biomedical image informatics fields, many applications perform a specific processing about anatomical organs or cell types, among others. Thus, an interactive bioimage segmentation and matching application will contribute substantially for biological scientists to analyze and visualize the cells, molecules, micro-structures, and organisms. Also, the bioimage matching algorithms will provide new opportunity in clinical diagnosis and treatment plans for medical healthcare. With these ideas, this thesis mainly investigated bioimage matching applications including biomedical image segmentation.

This thesis focuses on interactive bioimage segmentation and matching using iPad. At the current stage, this application is able to compare between two images in only 2D, and 3D image matching is beyond the scope of this thesis. Also, the iPad is used as a front-end module as a client for user interaction, and the back-end module are used as a server which perform the computing-intensive image matching algorithm with a database of images. We also showcased two applications for food images and biomedical images.

The rest of this thesis is organized as follows. Chapter 2 presents the background information, and briefly reviews the advancements of bioimage informatics, image matching methods, and iPad attributes. Chapter 3 describes the design and implementation of the front-end

and the back-end modules. Chapter 4 demonstrates two application scenarios for image matching algorithms.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we describe the background information. Chapter 2.1 states the bioimage segmentations and bioimage matching methods for biomedical informatics. Chapter 2.2 describes the OpenCV library including APIs, basic attributes for images, and image algorithms. Chapter 2.3 discusses the SIFT algorithm which can detect key-points in multiple images of the same location. Chapter 2.4 demonstrates the CFNetwork technology. This technology is based on the iOS system that can transfer biomedical image data sets from the client to the server via the Internet. Chapter 2.5 gives a brief explanation of the touch technology on the iPad. We can segment an image into semantic regions by using this technology.

2.1 Bioimage informatics

The bioimage informatics is most related to using computational methods to analyze and visualize bioimage such as cellular and molecular images. Especially, this thesis focuses on biomedical images from image acquisition systems like microscopy or MRI. The need and use of the bioimage informatics in the computer vision will cause an increase in the development of algorithms leading to open and new technologies. It focuses on improving novel image processing, databases, data mining and visualization techniques.

The bioimage segmentation plays an important role in the biological fields. Hence, it will help biologists work in biological image segmentation and tracking. For instance, it can help with localization of pathology, study of anatomical structure, quantification of tissue volumes, diagnosis, treatment planning, and computer integrated surgery. Many researchers have made an

effort to improve its algorithm. Unfortunately, there is no fully optimal method for general images. There are well-established algorithms for biomedical segmentation such as GrabCut, Bayes matting [71], graph cuts [73], and intelligent scissors [72].

Bioimage matching is widely used in biological computer vision, including point matching, shape-specific point matching, template matching, mutually-best matching, principal axis matching, dynamic programming, feature-based matching, and graph matching. These algorithms are applied to cortical structures and actively in human brain MRI data. There are several phases to perform image matching for biomedical images.

First, we need to segment the brain image. The image registration software removes non-brain matter from the subject image. It is used to co-register subject and atlas images. Consequently, labeling the subject images can be divided into segmenting tissue classes. Second, we have to construct 3D pieces using the K-means algorithm. It extracts 3D pieces by constructing 2D skeletons from extracted sulci. The next step is to match pieces from the atlas and subject. It computes differences between each pair of subject and atlas matching pieces by taking the minimum differences. Finally, there exists a label subject image. Atlas label boundaries are locally translated, depending on the difference in their local information between atlas and matching subject pieces. The atlas label volume is warped and transformed to new boundaries. After warping, the segmented gray matter can be assigned as the major label.

2.2 OpenCV library

The OpenCV (Open-Source Computer Vision) [26] is a library of programming functions for general computer vision that includes several hundreds of computer vision algorithms. The OpenCV has a modular structure, which means that the package includes several shared or static libraries as follows.

- Core: A compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions that are used by all other modules.
- Imgproc: An image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, and histograms.
- Video: A video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- Calib3d: It refers to basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- Features2d: Salient feature detectors, descriptors, and descriptor matchers.
- Objectect: Detection of objects and instances of the predefined classes. For example, faces, eyes, mugs, people, and cars.
- Highgui: An easy-to-use interface for video capturing, image and video codecs, as well as simple UI capabilities.
- Gpu: GPU-accelerated algorithms from different OpenCV modules such as FLANN and Google test wrappers, Python bindings, and others.

The OpenCV library [27] also provides numerous algorithms such as loading images and output, optical flow, and gesture recognition as follows in detail.

Table 2.1: Categories of video and image processing algorithms

Image functions	Data Structures	Contour Processing	Geometry
Features	Image Statistics	Image Pyramids	Morphology
Background Differencing	Distance Transform	Thresholding	Flood Fill
Camera Calibration	View Morphing	Motion Templates	CAMSHIFT
Active Contours	Optical Flow	Estimators	POSIT
Histogram	Gesture Recognition	Matrix	Eigen Objects
Embedded HMMs	Drawing Primitives	System Functions	Utility

The scale invariant feature transform (SIFT) library performs by using the popular open-source computer vision library OpenCV. The SIFT library also can use the low-level and high-level algorithms as shown in Table 2.1. Here is one example in the OpenCV library.

The Fourier Transform (TFT) will decompose an image into its sinus and cosines components. This processing will transform an image from its spatial domain to its frequency domain. Any function may be approximated exactly with the sum of infinite sinus and cosines functions. Here are two dimensional images Fourier mathematically transforms:

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-i2\pi\left(\frac{ki}{N}+\frac{kj}{N}\right)} \qquad e^{ix} = cosx + isinx$$

The f is the image value in its spatial domain and F function in its frequency domain. The results of the transformation are complex numbers.

Fig. 2.1: Input image

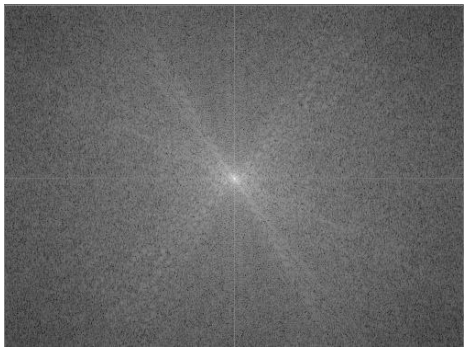

Fig. 2.2: Fourier transform image

As we can see the most influential components of the frequency domain follow the geometric rotation of objects on the image. After that, we may calculate the offset and perform an image rotation to correct miss-alignments. The OpenCV library provides many API functions [28]. Some of them that will be used in this thesis are as follows in detail.

- cvLoadImage(): loads an image from the specified file and returns the image pointer to the loaded image.
- cvCreateImage(): specifies where the image header will be created and allocated.
- cvShowImage(): displays the image in the specified window.
- cvWaitKey(): indicates that the program should wait for a pressed key.
- cvReleaseImage(): means that the image data and header will be deallocated.
- cvSetZero(): describes a set specific array element.
- cvSetImageROI(): sets an image Region Of Interest (ROI).

Table 2.2 illustrates a basic structure called IPL image header. The IplImage structure was inherited from the Intel image processing library. The image information of IplImage attributes is significantly important for handling image data.

Table 2.2: The basic attributes of the IplImage header

Type	Attributes name	Comment
Int	nSize	Sizeof(IplImage)
Int	ID	Version, always equals 0
Int	nChannels	Number of channels
Int	alphaChannel	Ignored by OpenCV
Int	depth	Pixel depth in bits
Char	colorModel[4]	Ignored by OpenCV
Char	channelSeq[4]	Ignored by OpenCV
Int	dataOrder	0 = IL_DATAORDER_PIXEL: Interleaved color channels, 1: Separate color channels
Int	origin	0: Top-left origin, 1: Bottom-left origin
Int	align	Alignment of image rows (4 or 8)
Int	width	Image width in pixels
Int	height	Image height in pixels
struct	_IplROI *roi	Region Of Interest (ROI)
struct	_IplImage *maskROI	Must be NULL in OpenCV
Void	*imageId	Must be NULL in OpenCV
struct	_IplTileInfo *tileInfo	Must be NULL in OpenCV
Int	imageSize	Image data size in bytes
Char	*imageData	A pointer to the aligned image data
Int	widthStep	The size of an aligned image row, in bytes
Int	BorderMode[4]	Border completion mode, ignored by OpenCV
Int	BorderConst[4]	Border completion mode, ignored by OpenCV
Char	*imageDataOrigin	A pointer to the origin of the image data

2.3 Scale Invariant Feature Transform (SIFT)

The Scale Invariant Feature Transform (SIFT) algorithm is a popular approach to detect distinctive and invariant image feature points which can be matched between a query image and a reference image. The SIFT has been one of the most successful and popular methods and has

been applied to a variety of computer vision research problems [29] including object recognition [31, 32], robotic matching [30], image retrieval, gesture recognition and video tracking.

The SIFT algorithm was first published by David G. Lowe in 2004. The features are invariant to image scale and rotation, and identify key-points in scale space. These features can be used to detect distinctive objects in different images and transform image data into scale invariant coordinates [1, 33]. Image features can be extracted from a set of reference images for local geometric deformations by representing blurred image gradients. Lowe's approach is used to detect the features that transform image data into scale invariant coordinates relative to local features. This approach consists of four major computational stages:

1. Scale-space extrema detection: The first stage is to identify candidate key-points by using a Difference-of-Gaussian (DOG) function. The image is iteratively convoluted with Gaussian functions.
2. Key-point localization: The next phase is to find the key-point in the DOG pyramid. The key-points in scale space are added to the local location.
3. Orientation assignment: This step is to assign to each orientation its key-point. The key-points are assigned one or more orientations based on local image gradients.
4. Key-point descriptor: A local image descriptor is computed for each key-point in the local image region.

The SIFT has been increasingly improved in the computer vision fields [7, 8, 9]. With this improvement, Rob Hess released the SIFT implementation in 2006 as an open-source library that implemented in a C version of the SIFT algorithm [34]. Thus, our work aims to implement the image matching application by using the open-source library on the iPad.

2.4 CFNetwork Framework

The iPad strongly supports the ability to deliver advanced functionality over an always-on Internet connection or Wi-Fi. Apple has also provided a framework, named CFNetwork [35], to support basic socket protocol and socket connectivity. The CFNetwork is a low-level, high-performance framework with the ability to communicate across network sockets. The CFNetwork framework is an extension of BSD sockets which provides well-defined standard socket abstraction API functions that can perform tasks supporting FTP and HTTP servers or a resolving DNS host. It can provide a variety of Cocoa classes [36] and networking APIs.

The Cocoa classes include a Web-kit class that displays web content in the browser window. The Web-kit and Cocoa classes are very high level implementation of networking protocols. Here is the structure of the software layers in Fig. 2.3.

Fig. 2.3: The CFNetwork and software layers

The CFNetwork consists of two API functionalities, CFSocket and CFStream, which are part of the Core Foundation framework and are fundamental to using CFNetwork. These API functions are listed as follows [37].

- CFSocket API: Sockets are the basic object of network communications. The CFSocket is an abstraction for BSD sockets and provides all the functionality of BSD sockets. CFSocket provides a number of functions that are specific to the CFNetwork framework. The following relevant functions are available on CFSocket objects.
    - CFSocketCreate: creates a CFSocket object of a specified protocol and type.

- CFSocketCreateWithNative: creates a CFSocket object for a pre-existing native socket.
- CFSocketCreateRunLoopSource: creates a CFRunLoopSource object for a CFSocket object.
- CFRunLoopAddSource: adds a CFRunLoopSource object to a run loop mode.
- CFSocketConnectToAddress: opens a connection to a remote socket.

- CFStream API: Socket streams provide an easy way to exchange data for reading and writing in a device-independent way. Each socket stream can be located in memory, in a file, or on network, allowing for synchronous or asynchronous communication. A stream is operating on a networking path that can be transferred to a sequence of bytes. The CFStream is located on top of the CFNetwork which is the foundation for CFHTTP and CFFTP. The following Fig. 2.4 shows the CFStream API structure.

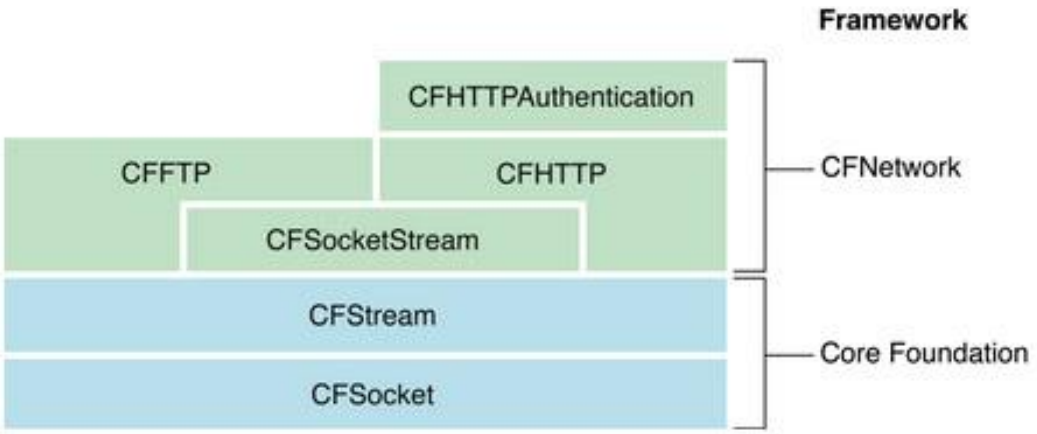

Fig. 2.4: The CFStream API structures

The CFStream supports two CFType objects with sockets: CFReadStream and CFWriteStream. The functions included are as follows.

- CFReadStream: Its functions allows for simple read operations with network sockets.
  - CFReadStreamOpen: opens a stream for reading.

- CFReadStreamRead: reads bytes of data from a readable stream.
- CFReadStreamGetBuffer: returns a pointer to a stream's internal buffer of unread data.
- CFReadStreamGetStatus: returns to the current state of a stream.

- CFWriteStream: This stream writes a byte stream either synchronously or asynchronously and manages the sending data through a CFSocket.
  - CFWriteStreamOpen: opens a stream for writing.
  - CFWriteStreamWrite: writes data to a writable stream.
  - CFWriteStreamCanAcceptBytes: indicates whether a writable stream can accept new data without blocking.
  - CFWriteStreamGetStatus: gets the current state of a stream.
  - CFWriteStreamScheduleWithRunLoop: schedules a stream into a run loop.

2.5 Touch Technology

Since Apple computer's iPad was released in 2010, one of the most important components has been its touch technology, which is a convenient way to input and access information on the iPad. There are several well-established touch screen technologies. However, the most important touch technologies are resistive and capacitive touch screen.

The resistive touch screen panel is composed of several layers. The most important features are thin, metallic and electrically conductive layers which also can be built with two flexible sheets coated with resistive material separated by a narrow gap. When users touch the screen, two metallic layers become connected at the point and register the contact location. Afterwards, the located information can be sent to the controller for processing. The resistive technology works with fingers or any object.

The capacitive touch screen uses a glass surface coated with transparent conductor. When the screen is touched with a finger it creates an electrostatic field, which can be measured as a change in capacitance. This can register more than one touch event simultaneously. Most portable devices are equipped with capacitive touch screens which use finger contact and support multi-touch technology.

One of the high-level technologies is multi-touch technology. If users need to zoom-in and zoom-out from an image, it is possible to control the image using two fingers with this technology. Therefore, Apple's multi-touch technology allows users to manipulate objects on the iPad in a convenient way. The multi-touch technology is also a basic technique component for numerous applications in recent days. With this technology, this thesis tries to implement and design the user interface software used by finger manipulation.

Apple's iOS library also provides a method for touch functions that is the UITouch class, which can store finger's position and generate some specific events when touching the screen. The UITouch object represents movement of a finger on the screen for a particular event. For instance, when the user touches the screen, UITouchPhaseBegan event occurs. If the finger is moving, UITouchPhaseMoved event occurs. All touch events correspond to objects whether it began, moved, or ended the gesture. There are five phases of a finger touch as follows [21].

- UITouchPhaseBegan: a finger starts to touch the screen.
- UITouchPhaseMoved: a finger moved on the screen.
- UITouchPhaseStationary: a finger is touching the surface but does not move on the surface.
- UITouchPhaseEnded: a finger has been released from the screen surface.

As we mentioned above, these five steps are to communicate with objects and occur with touch events. As a result, these events call to the callback function to respond. This response is the UIResponder class which can communicate to handle events for particular objects. The touch event is generated when a finger starts to touch the screen surface, move on the screen, or leave from the screen. There are touch events as follows [22].

- touchesBegan:withEvent: the user starts to touch down on the screen during the first stage of the event.
- touchesMoved:withEvent: keeps track of the movement when the user's finger is still moving on the screen surface.
- touchesEnded:withEvent: indicates that all objects or fingers has been released from the screen.
- touchesCancelled:withEvent: sends a particular event to the framework from the system event to cancel a touch event.

CHAPTER 3

SOFTWARE ARCHITECTURE AND METHODS

This chapter discusses the software architecture and methods for biomedical images. Chapter 3.1 explains the overall structures and processing procedure of the architecture. Chapter 3.2 describes how the front-end module can perform an interactive communication with the back-end module. Also, it demonstrates the GUI prototypes on the iPad and several components for biomedical images including segmentation. Chapter 3.3 discusses the SIFT algorithm which discovers image features between two images. Moreover, we will explain multiple threads. Chapter 3.4 demonstrates the segmentation methods. There are two types of segmentation: GrabCut and rectangle-cut. Chapter 3.5 gives a brief explanation of the image matching methods.

3.1 Architecture

In this section, we describe the overall architecture of the iPad 2 system that performs interactive operations between the front-end and the back-end modules. In Fig. 3.1, this prototype architecture consists of two modules based on the network: Back-end module and Front-end module. They are used in different networking technologies. The back-end module is based on the Winsock technology and the front-end module is based on the CFNetwork technology. One advantage of the iPad is that it can be connected to the Internet using 3G wireless network technologies or a Wi-Fi connection channel. This technology can allow users to communicate between the server and the client.

Users can send a particular image to the server. This means that user interaction should be operated on the potable iPad. The iPad has a great ability to connect to other devices anytime and anywhere. Multiple users can send a specific biomedical image to the back-end module.

This bioimage should be segmented on the front-end module side before sending an image to the back-end module. On the back-end module side, it should perform the image matching algorithm with an interacting database. Also, it begins to create multithreaded sockets and distributes the task into multiple threads for image matching algorithm.

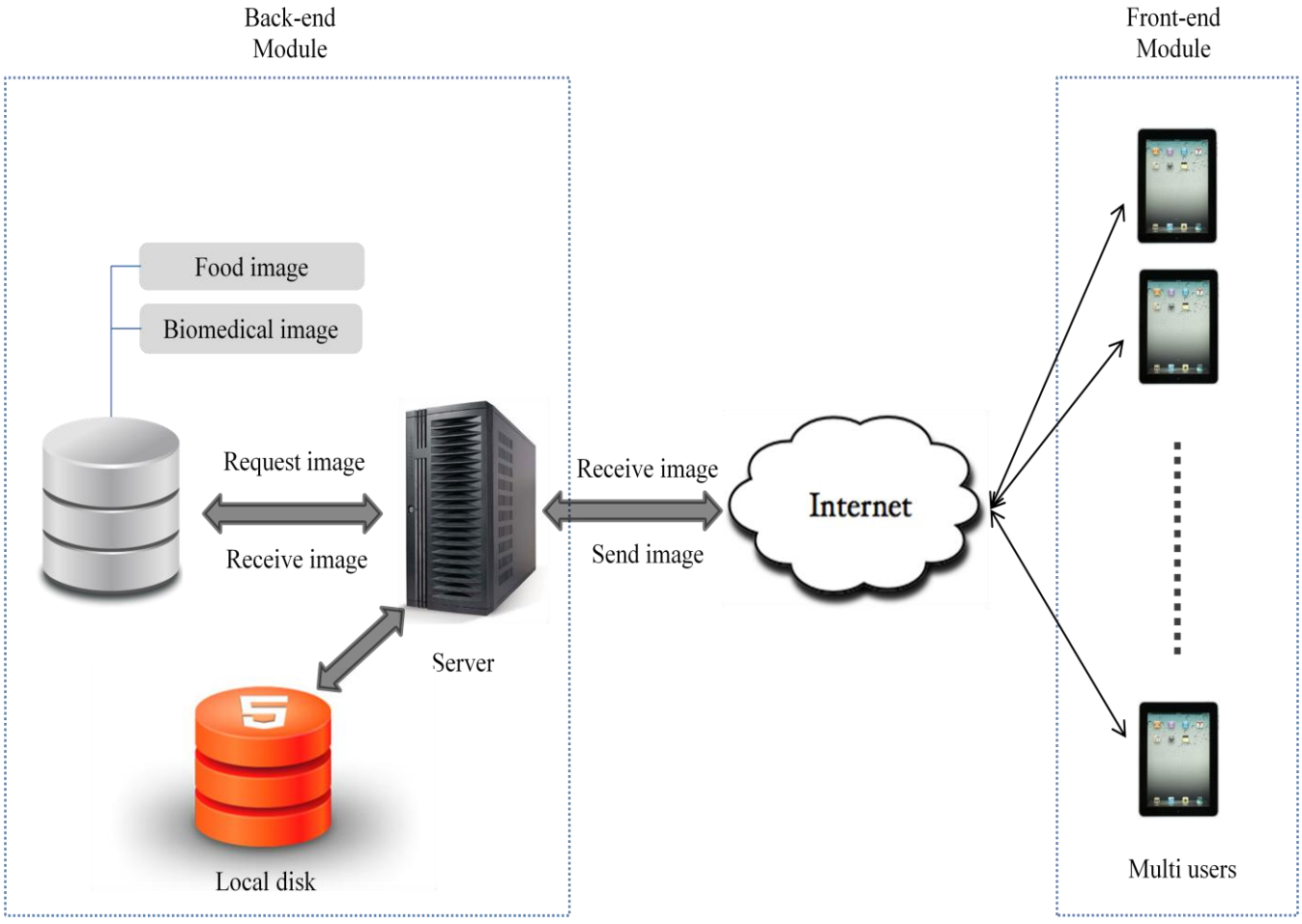

Fig. 3.1: The architecture of the mobile image matching application

There are no limitations in sending specific images to different devices of which devices are available to connect to the Internet. With this concept, we begin to construct and design an

image matching application that can freely communicate between the front-end and the back-end module since the iPad provides the fast network connection with friendly user interface.

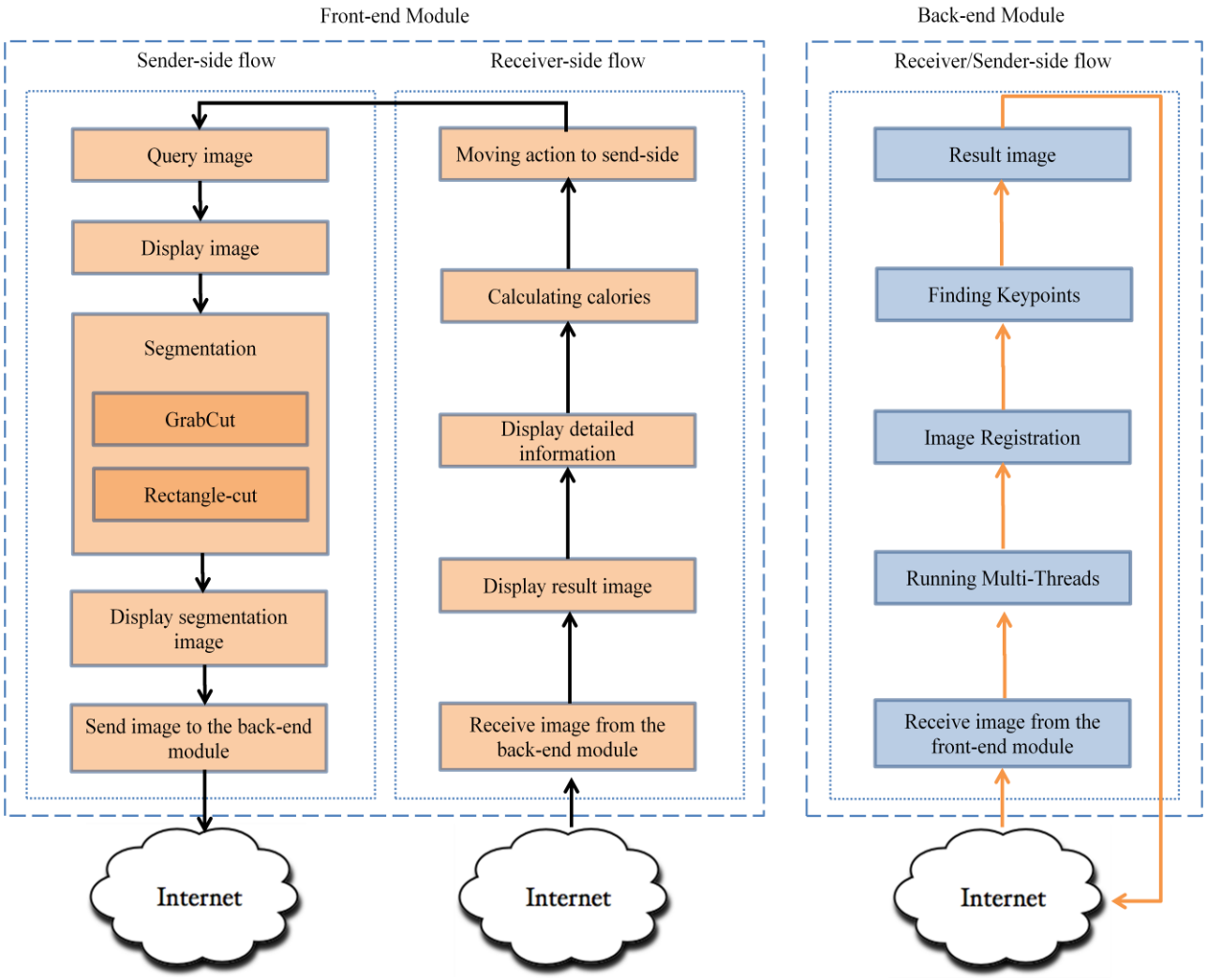

Fig. 3.2: The overall flowchart of the front-end and back-end modules

Fig. 3.2 illustrates the overall processing procedures of image matching prototypes of both the front-end and the back-end module. This flowchart demonstrates how a specific image performs both in the front-end module and in the back-end module.

We will discuss in detail how these modules process information over a network connection. The sender-side procedure in the front-end module begins to acquire a query image

from any kind of input image, and then displays it on the iPad screen. The next step is performing the image segmentation which will cut the real image.

There are two types of cutting methods using touch technologies: GrabCut[11, 41] and rectangle-cut [38, 39, 40]. We will briefly explain two types of image segmentation algorithms. The GrabCut is an image segmentation method based on a graph cut algorithm. This method is an effective and interactive tool for foreground and background segmentation. It segments a specific image over the original image using the GrabCut algorithm [23]. The rectangle-cut is an easy way to crop an image. Users can touch the screen surface and move their finger to select particular points which are located in the top left and bottom right corners of the rectangle.

We will discuss these cutting methods in detail in the image segmentation section. The user can cut a real image using their finger on the iPad screen surface. The cutting image will be displayed on the iPad's screen. Finally, the front-end module sends a segmented image to the remote back-end module over the network connection. The network connection in the front-end module uses the CFNetwork of the Apple's network technology based on the iOS platform. The CFNetwork supports very well-defined classes and it is easy to use them.

On the back-end module side, this server-side module supports multiple threads between the server and multiple clients. The server uses the multithreaded method which allows many users to connect to the server [42]. The server can receive a segmented image from the front-end module via Internet connection. Before the server begins image matching, the server can request images for the database which has several image paths in the database. All of the actual images are located in the local storage. As a result, the server can acquire two images: a query image from the front-end module and a reference image from the local storage. Now, the server can detect the best image matching object between two images.

At this time, the image matching phase is also running with multi-threads which can support matching images from a query image to many reference images. Also, this operation can match 1:1 matching methods. As a result, the server finds the best matching image, and then the back-end module sends a result image to the receiver-side flow in the front-end module via network. The network in the back-end module is used for the windows socket called Winsock. The Winsock technology is developed by Microsoft Windows Corporation. Windows sockets enable programmers to create advanced Internet, Intranet, and other network-capable applications for transmitting data across wireless networks.

The receiver-side flow in the front-end module should receive the result image from the back-end module. The result image will be displayed with the query image together on the iPad's screen. This means that we can do the next action with image information since this image received from the back-end module via Internet contains image information including the index number, company name, food image name, bioimage name, calories, and other information.

Therefore the detail information of the image will be displayed on the iPad. With this detailed information, we can calculate the calories of food. Finally the receiver-side flow will turn the authority over the sender-side flow in the front-end module.

3.2 Front-end Module

The front-end module is applied on the iPad because it is a portable device, provides great GUI components, and maintains strong connection to the Internet network anytime and anywhere.

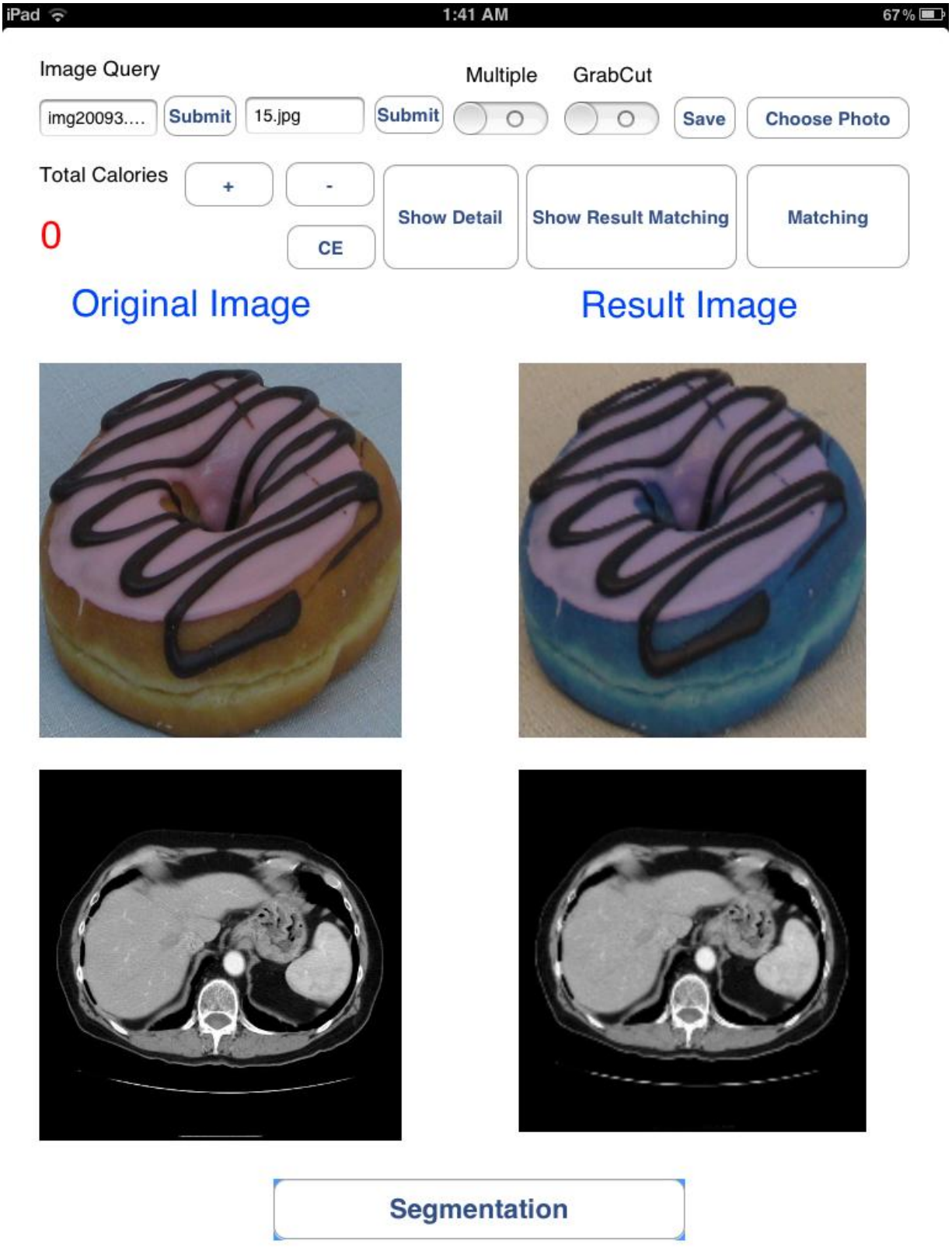

Fig. 3.3: The GUI prototype on the iPad

We designed the GUI prototype and implemented an application that allows users to easily control specific image processing on the iPad. We also programmed the prototype by using Objective-c based on Xcode IDE [19, 20]. As we can see, Fig. 3.3 shows the overall designed GUI prototype on iPad.

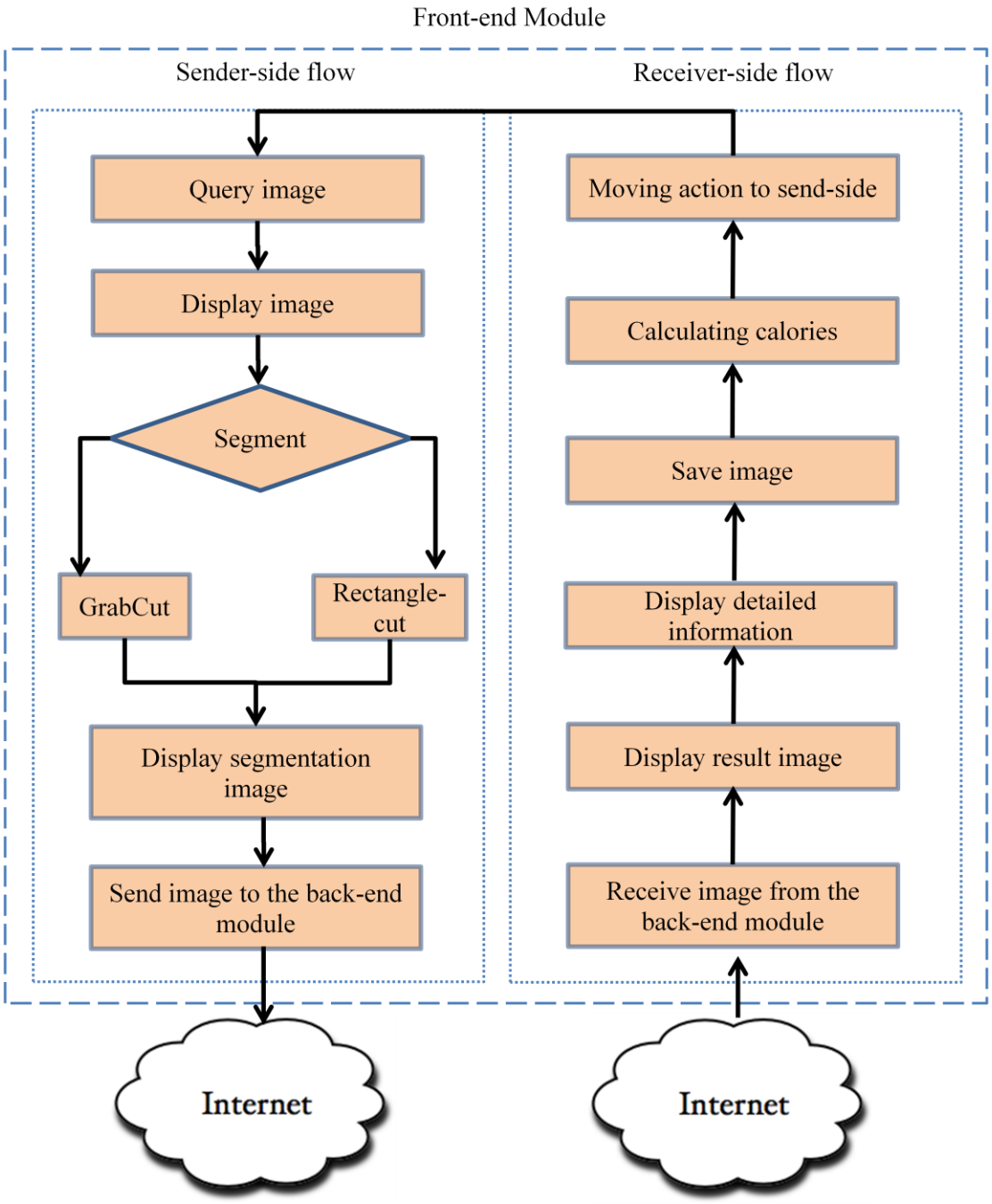

Fig. 3.4: The flowchart of the front-end module

There are several components and functionalities. We designed for the GUI prototype such as input query image, matching, show detail, show result matching, segmentation, and display image on the screen. In this section, we will discuss and analyze the GUI prototype, components, and functionalities in detail.

Fig. 3.4 describes the overall functionality flowchart of the front-end module. This flowchart will be divided into two components, sender-side and receiver-side components. These components are operated with different functionalities. The sender-side part performs image manipulation before sending images to the back-end module via Internet. The receiver-side part runs images displayed on the iPad after receiving images from the back-end module via the network. Also, the flowchart will be slightly different depending on the image.

For instance, the food image data set will provide more functionalities than the bioimage data set. The sender-side component will be followed as detailed. First, we can look over the input query image components. The following figure illustrates input methods as a GUI prototype. There are two methods to input image queries.

Fig. 3.5: The input query image from the text form

As we see in Fig. 3.5, the first method is using direct input from the iPad. We can type the image name on the iPad's screen surface. It will be possible to transmit an actual image from the internal local storage to the iPad's screen. These images are already stored and will be named "query image" meaning that it is an original image.

Fig. 3.6: The input query image from the photo library on the iPad

The second method uses the photo library as shown in Fig. 3.6. The iOS 5 provides access to the photo library through the UIImagePickerController class. The UIImagePickerController class [43] aims to provide applications with either a photo or video. It will be achieved by the user accessing the photo library. This class may also be used to provide applications with access to existing photos. The image picker view will be displayed in a popover when using the image picker in the context on the iPad. The setting source type property defines the source of the media of the UIImagePickerController object to one of the three supported types.

- UIImagePickerControllerSourceTypeCamera
- UIImagePickerControllerSourceTypeSavedPhotosAlbum

- UIImagePickerControllerSourceTypePhotoLibrary

These source type properties can be used when picking a photo. The Apple also supports multiple techniques for accessing photos from the local photo library within iOS. It is an alternative way to select multiple photos in the iPad photo library. Selecting a photo in any album in the photo library shows photos from the top left of the screen to the bottom right of the screen, and a specific photo we want can be selected by tapping it. The image will display on the iPad when users select a specific image either by using the direct input query image method or using the iPad photo library.

The next process is the segmentation of an image from either the direct input method or the photo library. This image segmentation can be used by two algorithms: GrabCut or rectangle-cut. The GrabCut can be used to solve graphical problems in computer vision [44]. It is an effective way to perform 2D image segmentation that can also provide easy control to manipulate an image with the iPad. The user can select the image location on the screen. These points in the location should be segmented between foreground and background with iOS touch technology [45].

The rectangle-cut is a method that segments an image by using the user's finger. A smaller rectangle should be located at the top left on the screen. This small rectangle can move the position and expand the area on the screen which will be selected by the user. Afterwards, the image should be segmented successfully and the image segmentation will be displayed on the screen.

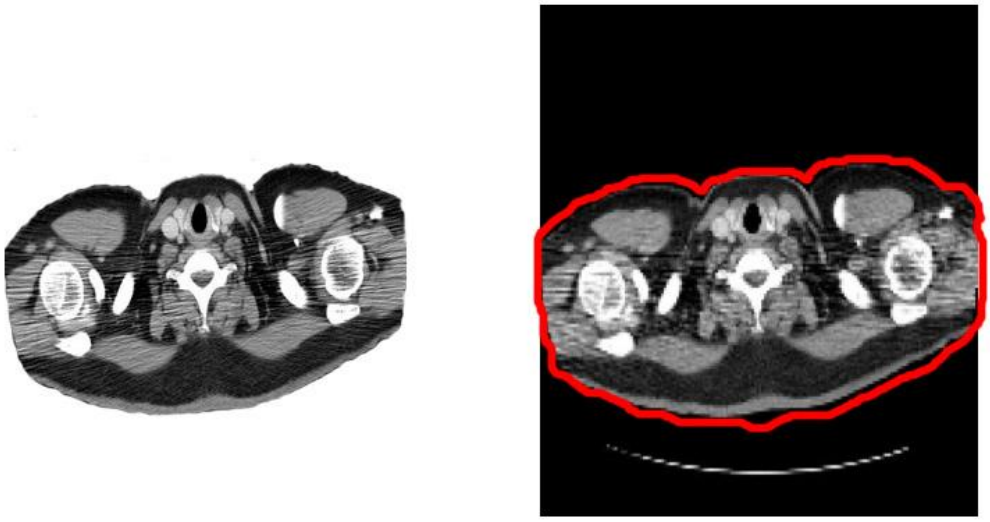

Fig. 3.7: The snapshot of a biomedical image using the GrabCut

Fig. 3.7 shows the segmentation by using the GrabCut algorithm [46, 47]. The right side image will be segmented by a red line and will be displayed on the left side.

Fig. 3.8: The snapshot of a biomedical image using the rectangle-cut

In the resulting image segmentation Fig. 3.8 [24, 48], it shows an image on the right side that will be segmented by the rectangle-cut method and will be displayed on the left side. On the sender-side component, the front-end module transmits the segmented image to the back-end module by using the CFNetwork technology. At this time, the client sends the binary biomedical

image data with image information such as image size, width step, width, height, depth, and channels. This image information is used in the OpenCV library. The OpenCV library uses IplImage structures for processing. The IplImage is one of the basic data structures to store information about the images.

On the receiver-side component in the front-end module, the resulting image sends from the back-end module. This biomedical image data includes binary data, image attributes, and image information which have information such as company name, biomedical/food image name, calories, and other detailed information. The image information is transferred from the back-end module to the front-end module. The resulting image will use in the receiver-side part. Therefore the image data and information will be displayed on the screen. In this part, this module can detect the image property which distinguishes images such as food image or bioimage. The calorie calculator performs when the image is received from the back-end module. This calculator computes calories using add or subtract operations. Finally, the resulting image will be stored in the iPad which can reuse the next operations.

3.3 Back-end Module

The back-end module plays important roles in the entire application. This module can communicate with both the front-end module and the database. The image matching algorithm runs in this module for biomedical images. There are three primary components of the back-end module. The database-side component fetches the image information from the database and sends them to the receiver/sender-side component. The receiver/sender-side component is a primary part in the back-end module. It can create multiple threads for the image matching and performs the image matching algorithms.

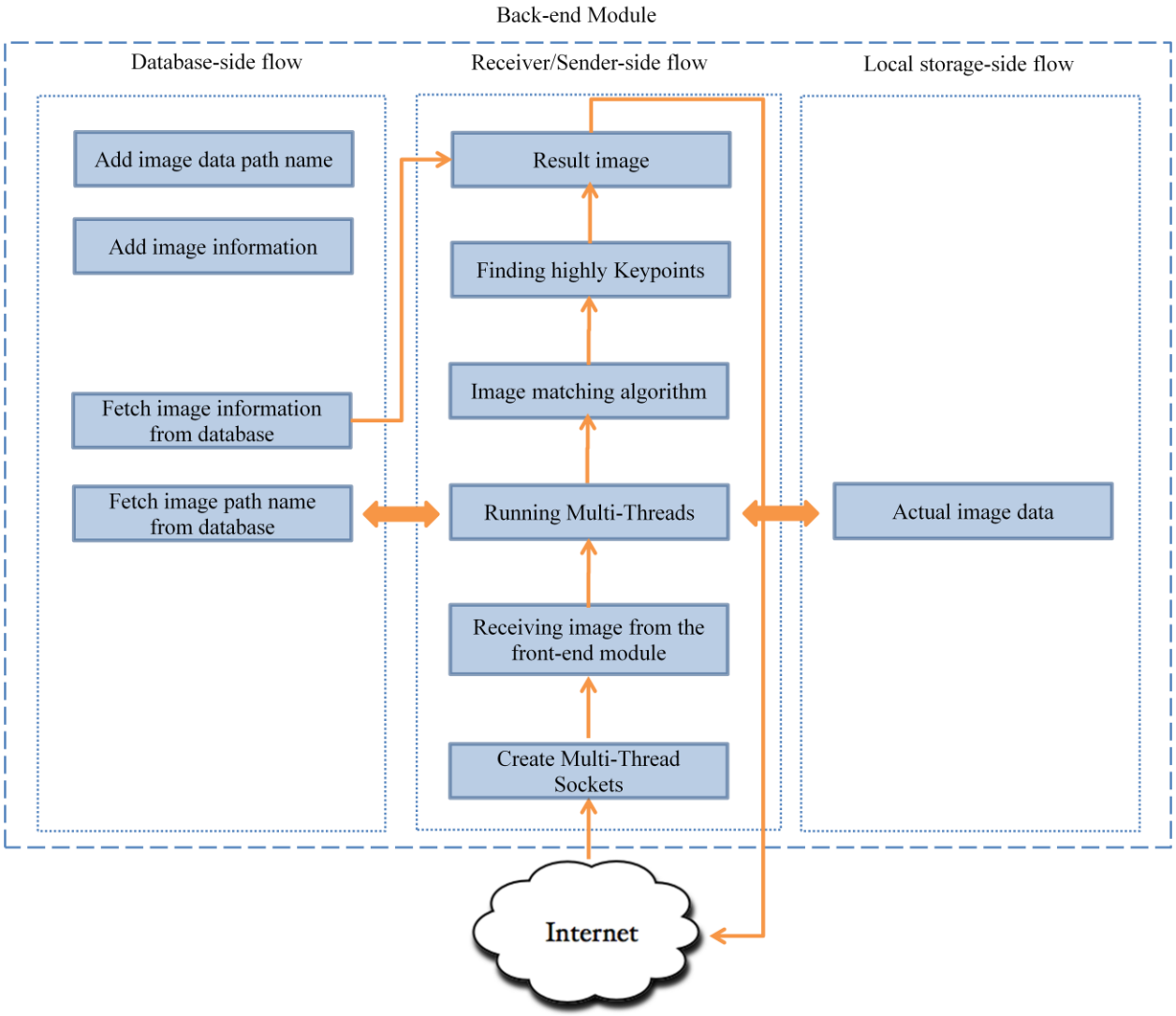

Fig. 3.9: The flowchart of the back-end module

Fig. 3.9 illustrates the overall processing procedures of the back-end module. The back-end module uses many other frameworks, API classes, libraries and methods which include the OpenCV library, Mysql Connector/C++ [49], windows socket [50], and multi threads. This module runs with the Mysql 5.1 database [51] which can connect to the Mysql Connector/C++. The Mysql Connector/C++ is a Mysql database connector for the C++, which is developed by Sun Microsystems. It provides an object-oriented programming interface and a database driver for connecting C++ applications to the Mysql server. The following classes were implemented by the Mysql Connector/C++.

- Driver: creates a pointer to the Mysql driver object.
- Connection: creates a pointer to the database connection object.
- Statement: creates a pointer to the Statement object to hold SQL commands.
- PreparedStatement: creates a pointer to the PreparedStatement object to SQL query by using sql::Connection::prepareStatement().
- ResultSet: creates a pointer to the ResultSet object to hold the results of any query we run.
- DatabaseMetaData: creates a pointer to the DatabaseMetaData object to handle rows and columns.
- ResultSetMetaData: creates a pointer to obtain metadata associated with result set and accessing result set metadata for number of columns and rows.
- ParameterMetaData: creates a pointer to handle metadata.

These classes are very powerful methods for connecting to the database. Using these classes, the back-end module can connect and communicate to the Mysql database. This module uses the windows socket library called Winsock based on Windows 7 while the front-end module uses the CFNetwork based on iOS platform.

The Winsock is a technical specification which defines a standard interface on the windows TCP/IP protocol. The windows socket enables programmers to create advanced Internet, Intranet, and other Internet-capable applications. The Windows sockets 2 [52] runs based on the Windows Open System Architecture (WOSA), as illustrated below.

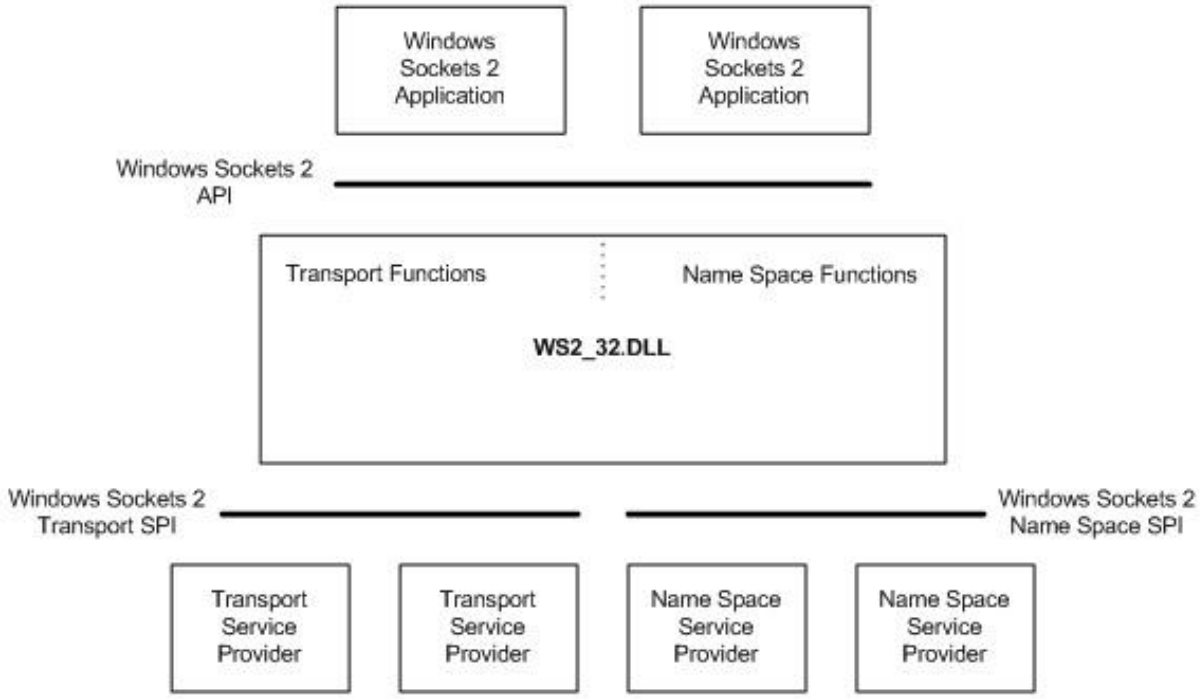

Fig. 3.10: The Windows Open System Architecture (WOSA)

The Winsock also supports APIs allow windows-based applications to access the transport protocols. The following lists provide concise descriptions of each Winsock function.

- Accept: permits an incoming connection attempt on a socket.
- Connect: establishes a connection to a specified socket.
- Bind: associates a local address with a socket.
- WSAStartup: initiates use of WS2_32.DLL by a process.
- WSACleanup: indicates to finish use of the WS2_32.DLL.
- WSACreateEvent: creates a new event object.

The back-end module can provide multiple threads. There are two types of threads used in this thesis: image matching and socket connection. Hence, the image matching involves heavy computing and is very time consuming. The multiple threads provide an effective way to save image matching time based on image contour structure. This module accepts multiple users with

a specified socket which can communicate with more than one client at the same time in the same network. The following figure shows the multithreaded socket architecture.

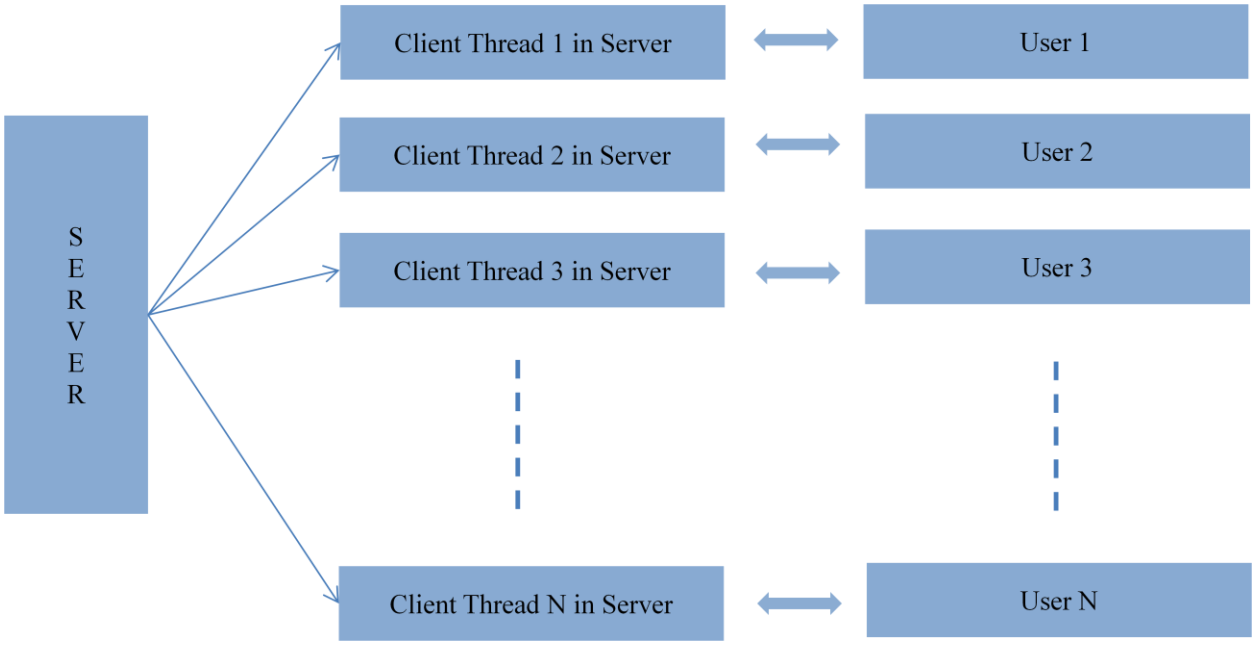

Fig. 3.11: The multithreaded socket architecture

The server creates a separate client thread in the server side for communicating with clients. For each client socket, there is an individual client thread for independent communication. The back-end module performs image matching algorithms with multiple threads. As shown in Fig. 3.12, this server module can create both main thread and worker threads. In this module, there are 5 threads based on the dual-core Intel processor.

In the same multithreaded process on shared-memory multiprocessor, each thread in the process can run on a separate processor at the same time, resulting in parallel execution. In this module each worker thread can performs 20 image matching comparisons between a query

image and a reference image. There are 100 sample reference images in the databases.

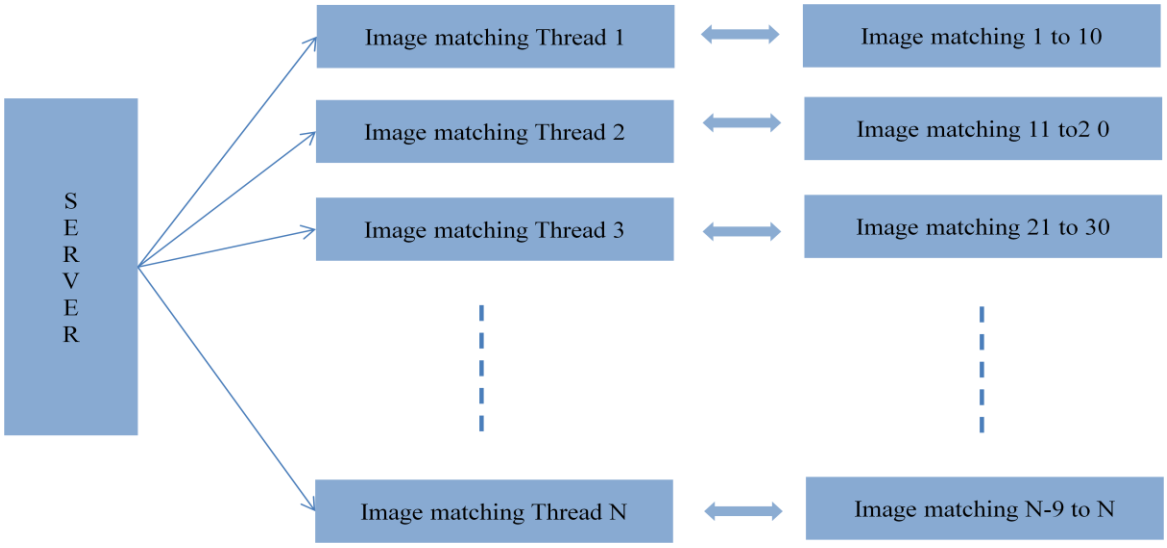

Fig. 3.12: The multithreaded image matching architecture

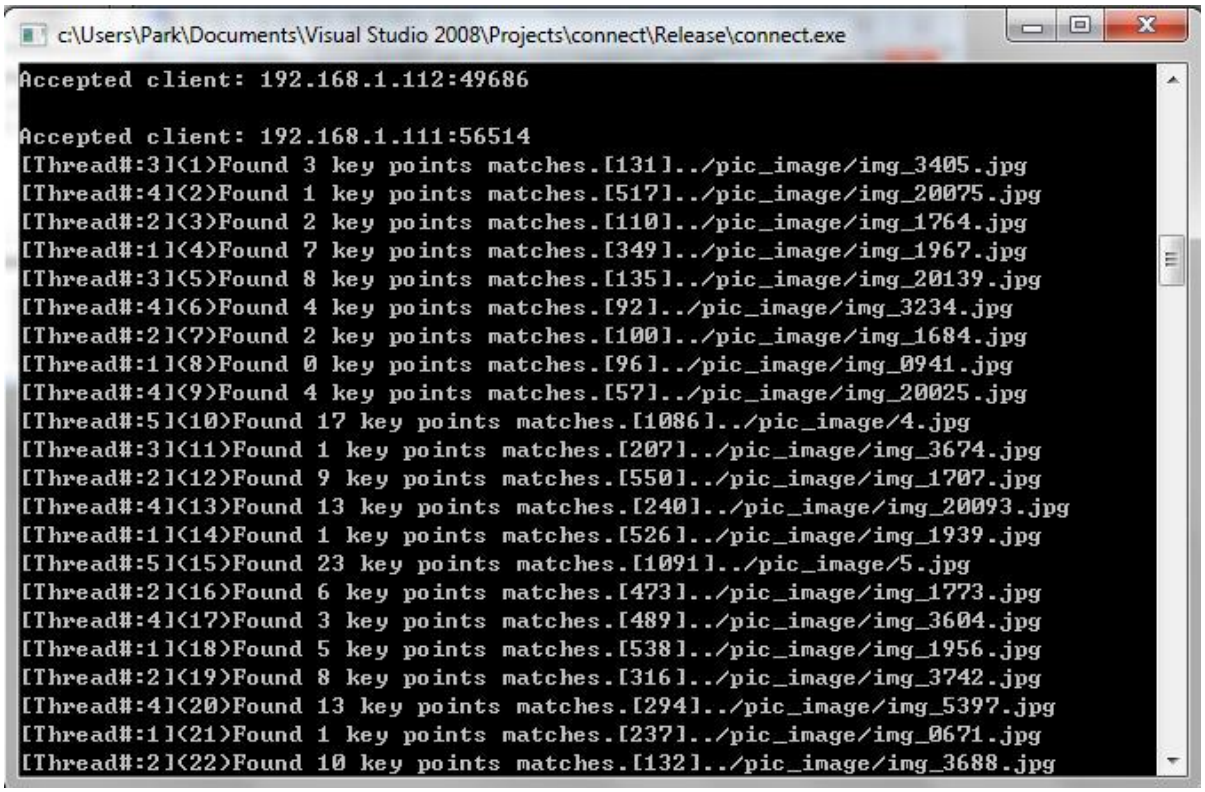

Fig. 3.13: The back-end module execution with multithreaded socket and image matching

Fig. 3.13 demonstrates the sample screen with multithreaded socket and image matching. As we see in Fig. 3.13, there are two accepted clients on the top of the figure; one is

192.168.1.112, another is 192.168.1.111. The multiple threads can communicate one or more than two between the front-end and the back-end module simultaneously. This module runs with 5 threads. Each thread performs image matching with 20 images. Each image matching algorithm should return key-points which allow a single feature to detect candidates with great probabilities in a large database of features [53, 54, 55].

The back-end module can perform with a large database. This module is used in Mysql database and is included 7 columns as following table.

Table 3.1: The table descriptions of the database

Columns	Contents
Id	A primary key to find its information
Image_path	A reference image path name which stored in local disk
Company	A company name of the image
Food	A food name of the image
Calories	A calories of the image
Nutrition	A nutrition information for image
Detail	A detailed information for image

The back-end module requests the image path name to the database when image matching algorithm is running. Through the image path name, the back-end module will get an actual biomedical image in the local storage because actual bioimage datasets will not have been saved to the databases.

The back-end module is implemented base on windows programming by using the Microsoft Foundation Class library (MFC) [56] with windows socket. The MFC is a library which provides the windows API functions in C++ classes, including functionalities that enable them to use an application framework.

In this section, the receiver/sender-side component performs several processing for biomedical image when the image receives from the front-end module. First, this component creates multiple sockets, binds it to a communication port, and waits for connection request from a client. Once the server accepts the client's connection request, it will create another socket and bind it to another port. This component requests the image path name to the database and it will get an actual image data in the local disk. Afterwards, the image matching runs. It detects keypoints between two images. As a result, we should get the same image in many candidate images. At this time, this module fetches detailed information for biomedical images from the database. This resulting image sends to the front-end module with both bioimage data and information.

The database-side part initially adds to image path name and image information by using the receiver/sender-side component. They are connected through communication. This component sends the image path name and information to the receiver/sender component. The local disk-side component has a set of data collection in the local storage.

3.4 Image segmentation Methods

The image segmentation utilizes two methods: rectangle-cult and GrabCut. Both methods will remove unnecessary foreground and background part in the image in consideration. They will help to detect accurate keypoints. Both perform a graph cut to separate an object from the background.

The rectangle-cut dissects a rectangle into an image. This method uses vertical and horizontal straight-line cuts. The rectangle-cut can be segmented by a straight-line which expands area toward either vertical or horizontal simultaneously. There is an initial rectangle

shape which is located in the top left of the screen. The following Fig. 3.14 shows the performance of the rectangle-cut for food and biomedical image.

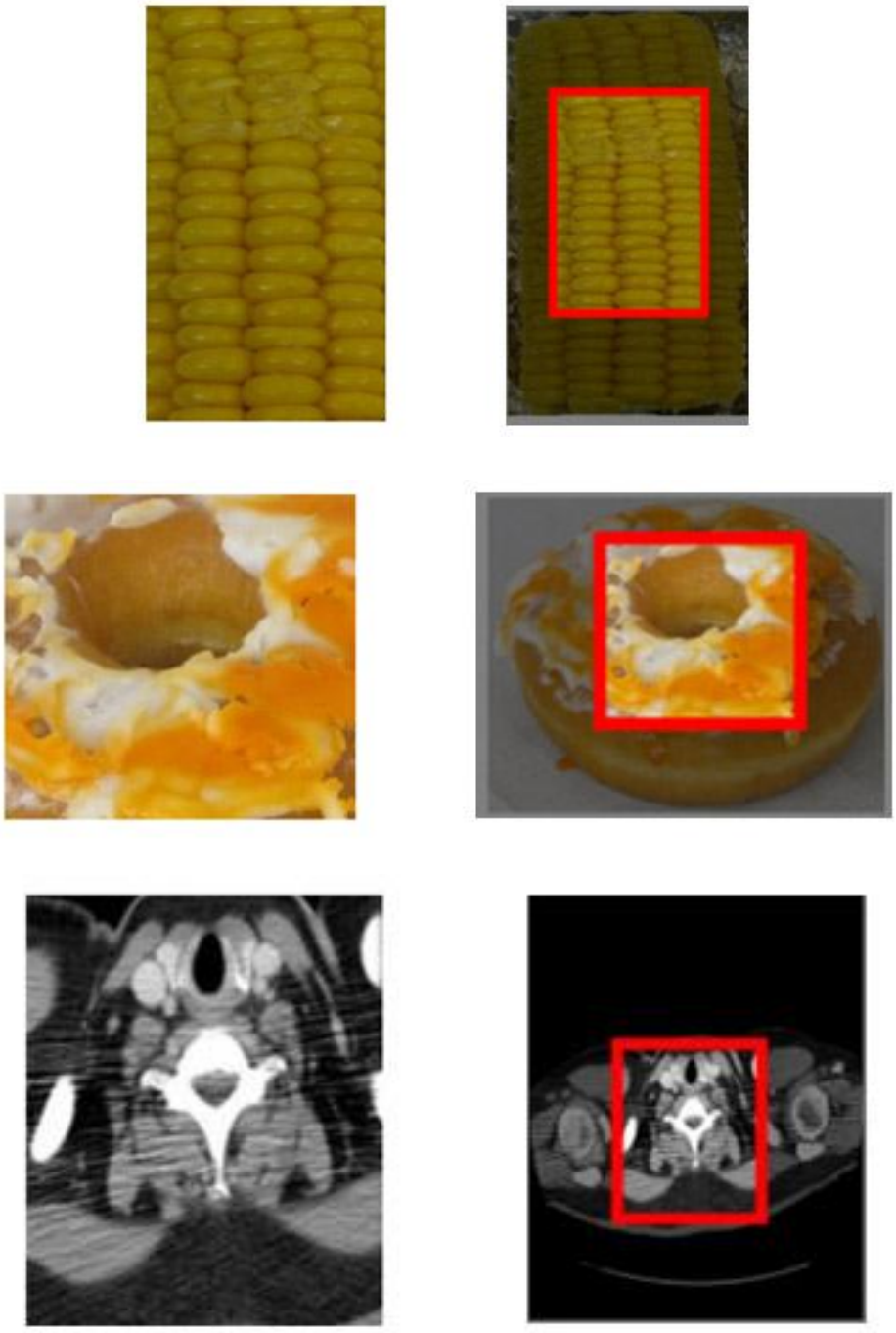

Fig. 3.14: The rectangle-cut snapshot in food images and bioimage

As we illustrate in Fig. 3.14, the right side shows an original image before the segmentation. After segmentation, the cropped image shows on the left side. The red rectangle line indicates the position which will be dissected.

The user touches the rectangle line which controls the rectangle size by using the user's finger. This algorithm initially measures the information of the line and points. This line will be expanded by using the touch technology which involves the UIResponder class. This class calls the touchesBegan:withEvent method when the user touches the screen. This method can record the initial starting point. When the user's finger is moving on the screen, the UIResponder class calls the touchesMoved:withEvent method. At this time, the coordinates of a pointed position will expand by the finger's moving. The line of the rectangle is not larger than the bottom line and is not smaller than 0. This means that this line should move in the coordinate geometry. All points can be recorded in the system when the user's finger leaves the screen.

The GrabCut [61, 62] is a very powerful algorithm for interactive segmentation of monochrome image. It has been detailed in the paper [10] that detected optimal segmentation of the 2D images. First, they described the image segmentation all of images that are indicated to an array $\mathrm{Z} = (z_1, ...z_n, ...z_N)$ of grey scale values. This segmented image represented as an array of "opacity" values $\underline{\alpha} = (\alpha_1, ...\alpha_N)$ at each pixel. The value of α_n can be between 0 and 1, but for hard segmentation {0, 1} belongs to α_n , with 0 for background and 1 for foreground. Foreground and background grey-level distributions are described by $\underline{\theta}$ parameter. $\underline{\theta} = \{h(z;\alpha), \alpha = 0, 1\}$, one for foreground and another for background.

The segmentation of grey scale task indicates the opacity variables $\underline{\alpha}$ from the given image data z and the model $\underline{\theta}$. There exists energy minimization of the segmentation. An energy function can correspond to segmentation, $\mathrm{E}(\underline{\alpha}, \underline{\theta}, \mathrm{Z}) = U(\underline{\alpha}, \underline{\theta}, \mathrm{Z}) + V(\underline{\alpha}, \mathrm{Z})$. The data term U

evaluates the fit of opacity distribution $\underline{\alpha}$ to the data Z, given the histogram model $\underline{\theta}$ and data term *V* refers to smoothness term. This energy helps to improve coherence in particular spot of similar grey-level. It encourages obtaining better image segmentation results which are acquired by defining pixels to be neighbors. As a result, the energy function is defined; the segmentation will use a standard minimum cut algorithm $\hat{\underline{\alpha}} = \arg\min_{\underline{\alpha}} \mathrm{E}(\underline{\alpha}, \underline{\theta})$ for hard segmentation.

There are three types of improvements which are open to new hard segmentation algorithms. Firstly, the Gaussian Mixture Model (GMM) helps to expend to color space image for segmentation in place of histogram. Secondly, a more powerful, typically iterative, method can be generated by using the one-shot minimum cut estimation algorithm between estimation and parameter learning. Lastly, the interactive user activities can be reduced by incomplete labeling.

We can take image pixel Z_n for soft segmentation in RGB color space. The GrabCut uses the GMM to construct appropriate color space histogram. There is a supplementary vector K $= (k_1, ...k_n, ...k_N)$ to each pixel by using the GMM component. Each GMM is either for background or for foreground models. After applying vector K to the original equation, new energy function will become $\mathrm{E}(\underline{\alpha}, k, \underline{\theta}, \mathrm{Z}) = U(\underline{\alpha}, \underline{k}, \underline{\theta}, \mathrm{Z}) + V(\underline{\alpha}, \mathrm{Z})$, where the term *U* is taking account of the color GMM models.

With the GrabCut algorithm [57, 58], we have applied to image segmentation on iPad with touch technology. When the user can touch the screen, the system remembers its position by using the touchesBegan event. When the finger is moving on the screen, the line will draw by following the finger, and calls the touchesMoved event. When the user's finger is released on the screen, the touchesEnded event occurs. At same time, the line points or positions are stored

in the system. The result in Fig. 3.15 shows that images can be segmented by using the GrabCut algorithms.

Fig. 3.15: Snapshots of segmentations by using GrabCut

As illustrated in Fig. 3.15, it shows the performance of GrabCut on the food images and biomedical images. The image of the right side displays an original image before using the GrabCut algorithm [59, 60]. After applying the GrabCut methods, the segmented image shows on the left side.

3.5 Image matching Methods

The Scale Invariant Feature Transform (SIFT) algorithm was introduced by the Lowe's paper. This approach can perform extracting distinctive invariant features from images. The image features have many properties which are invariant to image translation, rotation, and scaling for matching images. It can be easily matched between two images performing particular tasks: image detection, recognition, and geometrical transformations.

The SIFT algorithm [63, 64] is performed in four optimization phases. The first stage of the SIFT algorithm detects keypoints using a cascade filtering approach. These keypoints identify the features of candidate keypoints by using local maxima and minima of a Difference-Of-Gaussian pyramid (DOG). The Gaussian pyramid represents a function $L(x, y, \sigma)$, where term of L describes for the scale space of an object. Two images are computed by DOG pyramid $D(x, y, \sigma)$ with an input image $I(x, y)$. $L(x, y, \sigma) = D(x, y, \sigma) * I(x, y)$. The DOG function can be performed from the difference of two adjacent scales using constant factor k which can be efficiently detected key-point locations. $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$.

The Gaussians perform iteratively to generate the set of scale space images. The DOG of images will be subtracted from nearby Gaussian images. The local extrema of DOG algorithm is detected by comparing a pixel to its 26 neighbor in the scale space. The DOG function can be divided each octave of scale space into an integer number. After all of octaves are processed, it begins to resample the Gaussian object which has twice the initial value of σ.

The second step finds the keypoint localization. It detects maxima and minima which are great candidates for key-points. This approach is for fitting a 3D quadratic function to local sample point to choose the interpolated location of the maximum. This quadratic function used a second order Taylor expansion of scale-space function. Local extrema of edges with low contrast are discarded since they are defected.

The next step assigns a consistent orientation to each keypoint. After finding good keypoint localization, an orientation can be assigned to each feature based on local object. Each pixel around feature location and orientation are computed as:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)$$

The Gaussian function weighted the gradient magnitudes that rely on the feature octave with σ that is 1.5 times that of the scale. This weighted gradient magnitudes used to an orientation histogram that has 36 bins covering the 360 degree range of orientations. The highest orientation histogram peak and other local peak are greater than 80% of the highest peak is used to create a key-point with this orientation. Consequently there are multiple key-points at the same location and scale.

Lastly, after assigning an orientation, local image descriptor is computed for the local image region. A key-point descriptor in a region around the key-point location is divided into 4 by 4 boxes. The key-point orientation and gradient magnitudes will be weighted by Gaussian window. Each arrow and its gradient orientation are rotated relative to the key-points orientation. Therefore an 8 orientation bins are established with 4 by 4 array of histograms in each box.

The open-source SIFT library is implemented in C by using the OpenCV library. This library provides useful functions for computing SIFT features in biomedical images. The open-

source SIFT library is used both KD-tree key-point formation and RANSAC transform computation. The KD-tree [66, 67] performs to match efficiently key-points with nearest neighbor search. The RANSAC algorithm [65] is widely used to compute image transforms from feature matches.

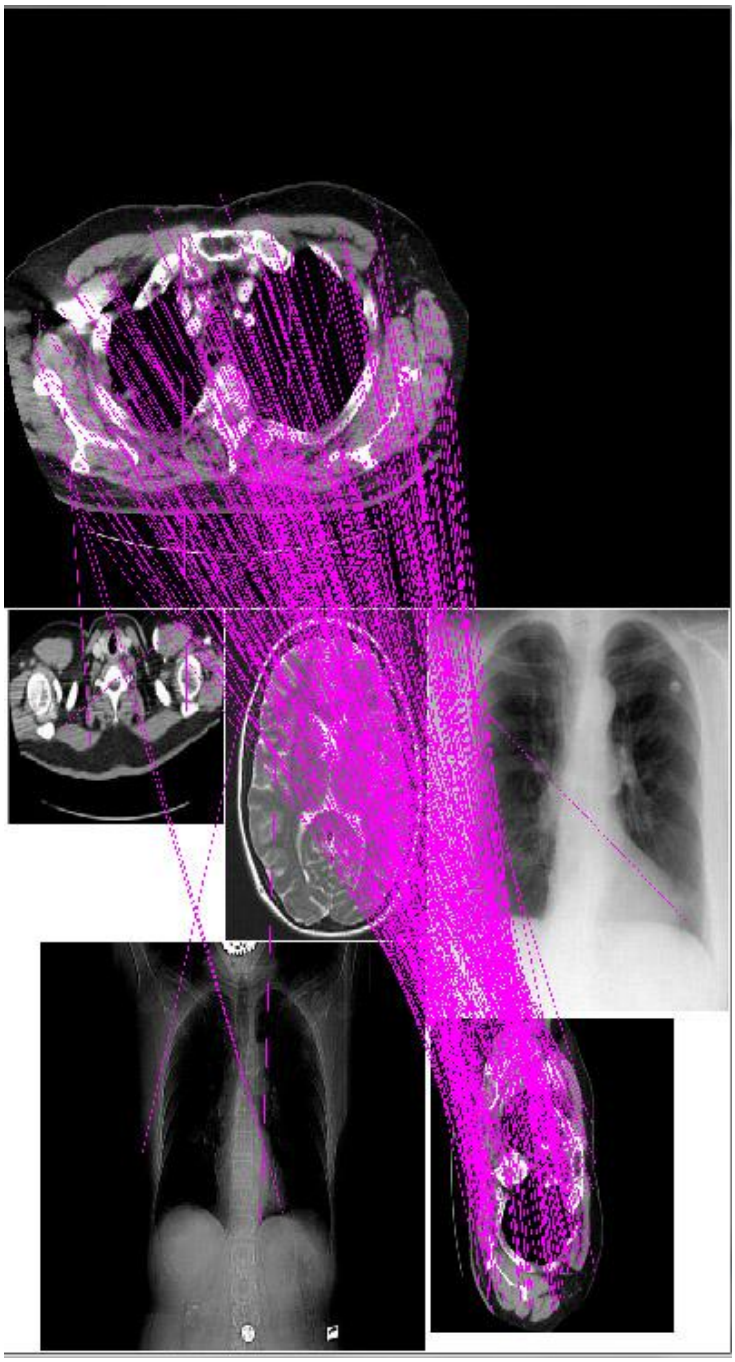

Fig 3.16: Biomedical image matching between SIFT key-points using KD-tree functions

Fig.3.16 depicts SIFT key-point matches between two images using the open SIFT library's KD-tree functions.

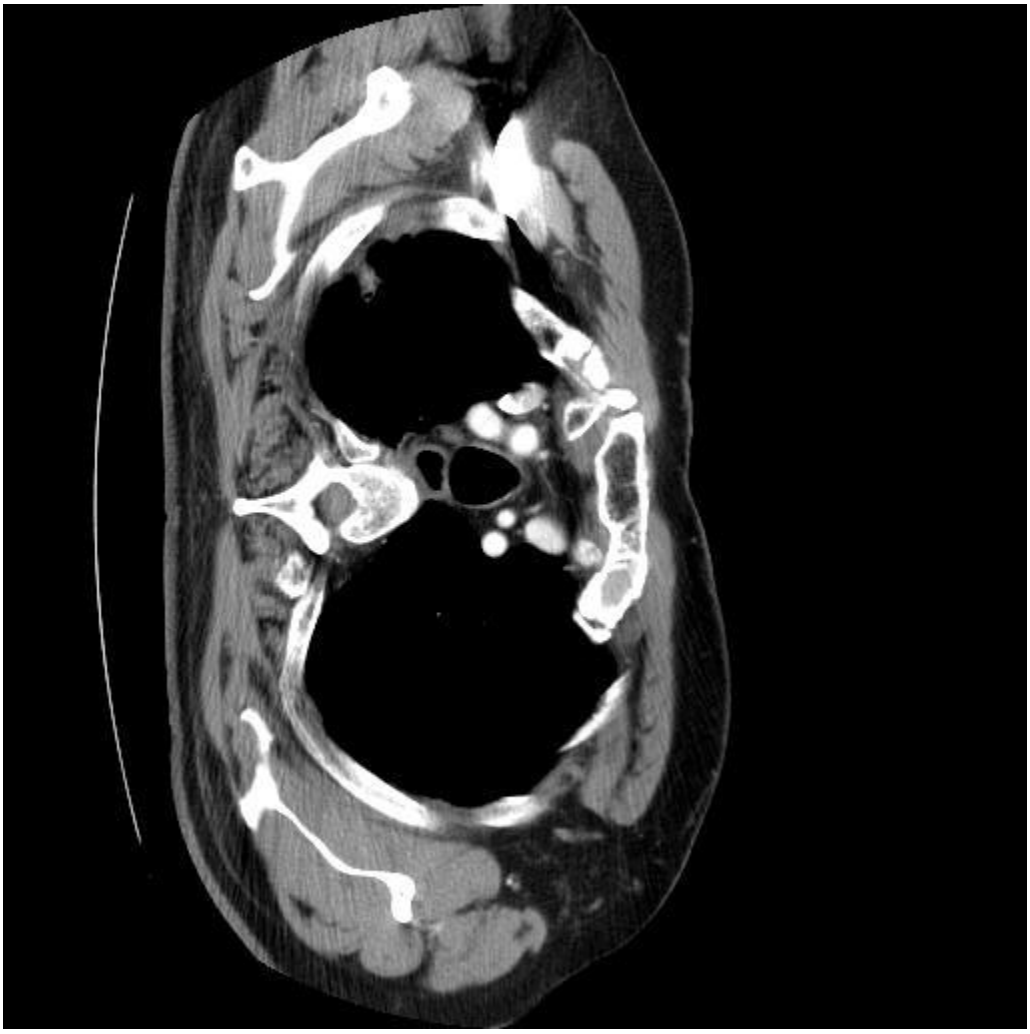

Fig. 3.17: A transform between two images using RANSAC functions

Fig. 3.17 shows the transformations based on the matched key-points using the SIFT library's RANSAC functions.

CHAPTER 4

APPLICATIONS

4.1 Food application

In this thesis, we proposed an image matching application for food images on the iPad. Food images are stored in the database. Each image has 500 by 500 pixels. A fast food dataset was obtained from the PFID (Pittsburgh Fast-food Image Dataset). They introduced the first visual dataset of fast food with total of 4,545 still images. The dataset is public available at http://pfid.intel-research.net/. They researched many restaurants on fast food recognition for dietary assessment. This dataset was collected by 11 popular fast food chains such as Arbys, Dunkin Donuts, KFC, and McDonalds. Using a set of image collection, this thesis investigated image matching programs for food images.

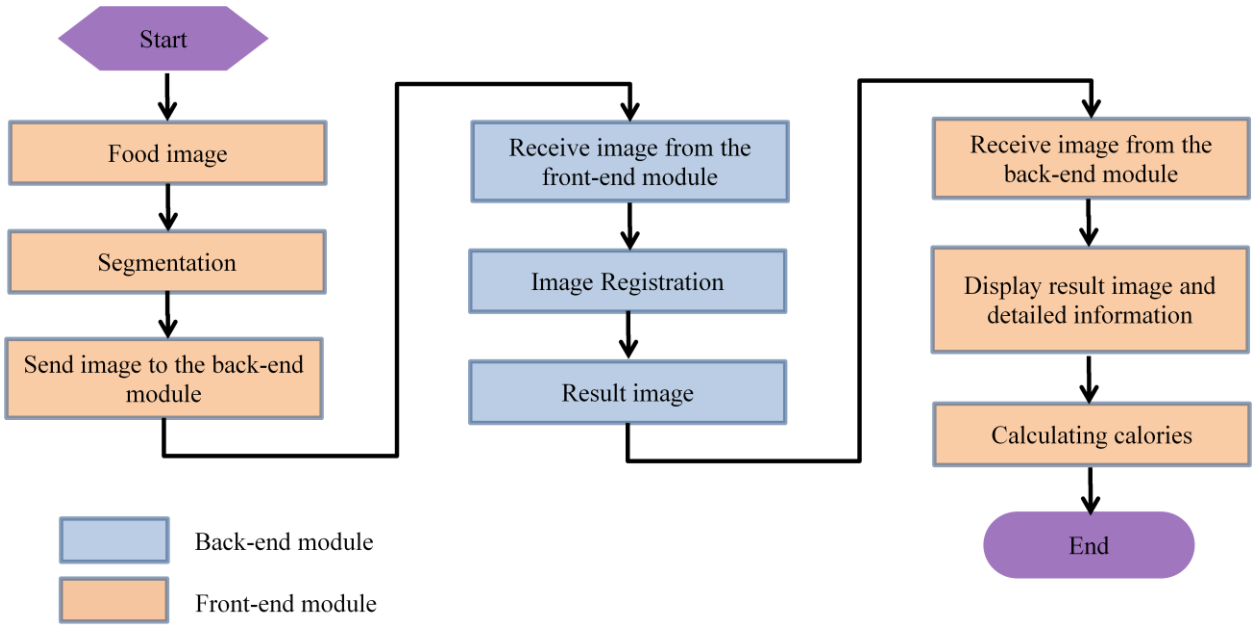

Fig. 4.1: The flowchart of food object tracing

Fig. 4.1 describes the flowchart of image information processing. This application will help people know how many calories they are going to consume from one food. This application is a smarter way to count its calories. An iPad application takes a picture of the meal and sends image to the server. The server performs the SIFT algorithm and finds the same image.

Afterwards the server sends the image data with image information to the iPad. On the iPad, the image information will be displayed such as calories, nutrition, and detailed information. This application makes it easier to track the calories in a customized food diary. Additionally, it should help people meet their health and fitness goals. The detailed information includes company name, food name, calories, and nutrition.

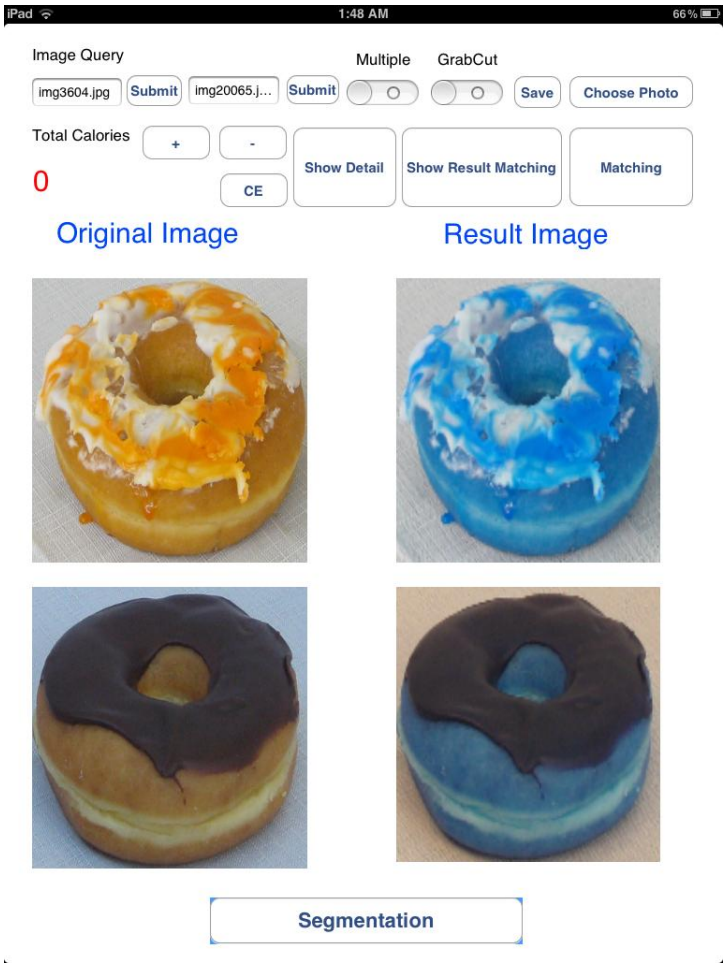

Fig. 4.2: Snapshot for food image after the SIFT algorithm

Fig. 4.2 shows an overall screenshot for food images. A user sends a food image (Original Image, left) to the server and gives the resulting food image (right) after finishing the image matching algorithm.

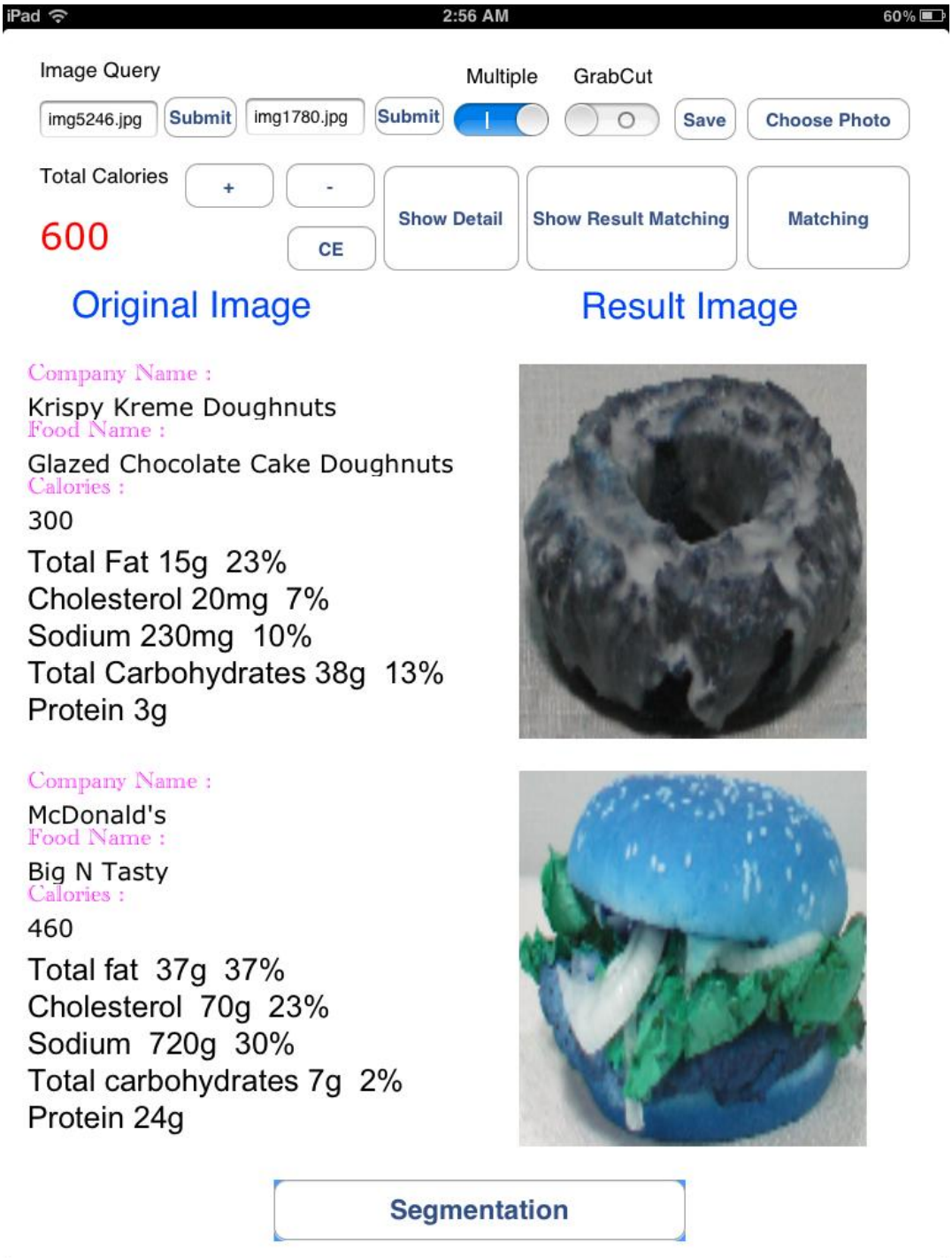

Fig. 4.3: Displays detailed information of the resulting image

The snapshot shows the detailed information for the resulting image which contains company name, food name, calories, and nutrition. Furthermore, this application can calculate the calories based on the resulting image. For instance, a calorie counter is a way to count our daily calorie intake. This model provides nutritional food information for calorie counter.

Fig. 4.4: Detecting key-points by using the SIFT algorithm for food images

As we can see in Fig. 4.4, it performs the SIFT algorithm to find key-points in different food images. The SIFT algorithm runs on the server side. This module provides a multiple image matching. This algorithm detects key-points by comparing shapes, scales, and locations in two images. In this example, the first image size is 500 by 500 pixels and finds the 403 key-points out of 430 candidates in their images. We quantified their accuracy as 93.7% at identifying the correct matches. The average area of the images tested was 250000 sq. pixels. If the image is larger than 500 by 500, the matching rate will be increased. Hence the number of key-points rises with increased sampling of scales. However, the speed of the image matching will be decreased when using the large size image.

The cost of computation rises when using the large image. In this thesis, we utilize the multithreaded methods in order to avoid heavy computation on the back-end module.

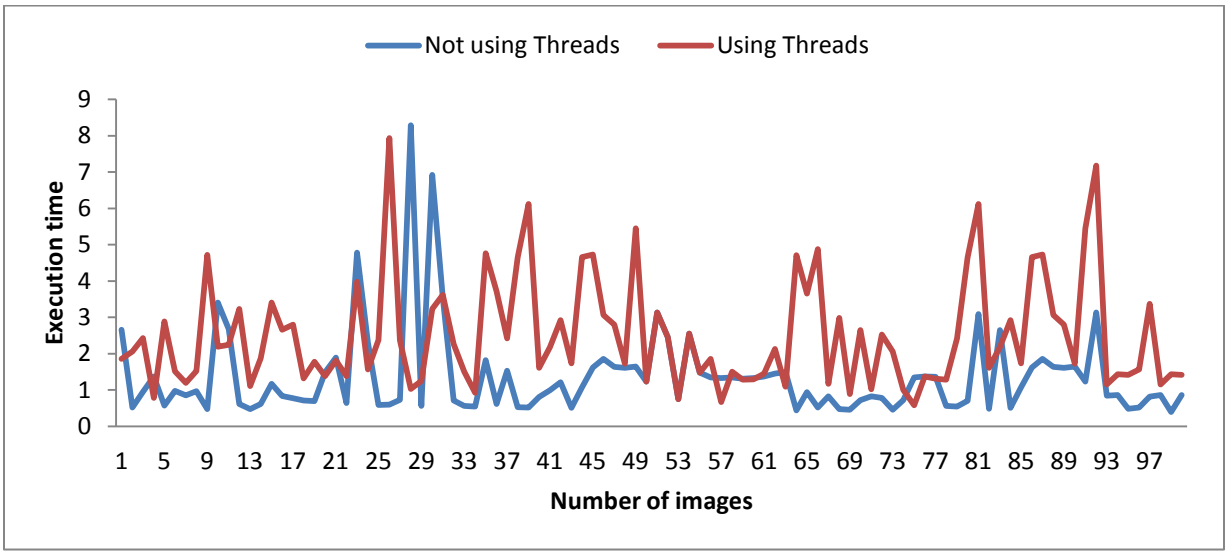

Fig. 4.5: Execution time between using threads and not using threads

Fig. 4.5 illustrates the estimation of the runtime of the multiple threads between a query image and a reference image. The performance of the image matching depends on the hardware environment. If it runs on the high-level equipments, the performance will be increasingly improved. However, we tested on the laptop computer which has Intel core 2 Duo processor 2.13

GHz, 4GB memory, and Windows 7 64-bit OS. Unfortunately, the running time of a single thread could be faster than multiple threads.

Table 4.1: Overall runtime comparison for food images

	Not using multiple threads	Using multiple threads
Overall Runtime	104.937 sec	53.857 sec

The table 4.1 compares the overall runtime for image matching either using a single thread or using multiple-threads. It is evident that a multi-threaded runtime is two times faster than a single processor.

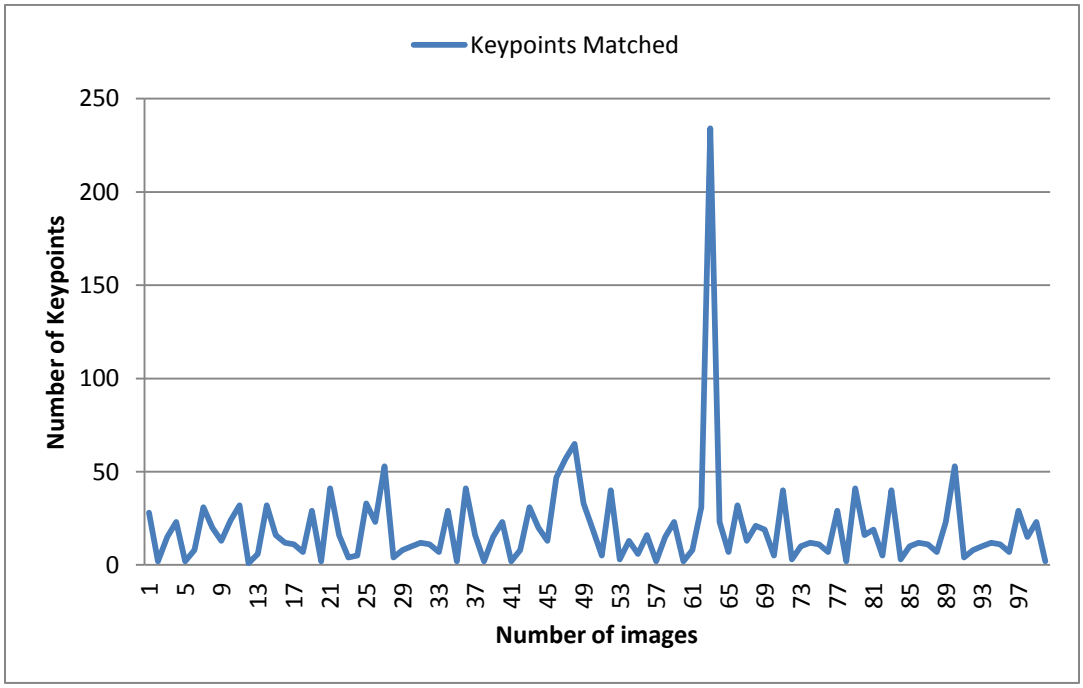

Fig. 4.6: A comparison of key-point matching between a query and a reference image

Fig. 4.6 shows the comparison of the key-point matching to compute a perspective transform based on these matches. This result is acquired by detecting key-points to each of 100 images from the databases. The result of the key-point will be different depending on the images. The accuracy can be measured by using Equation 4.1 is shown in Fig. 4.7. This result could be drawn from the Fig. 4.6 as follows.

Equation 4.1 Accuracy = [(# of Detected key-points*100) / (# of Candidate features)]

For instance, if we found the 234 matched key-points among the 240 candidate key-points, the accuracy would be 97.5%.

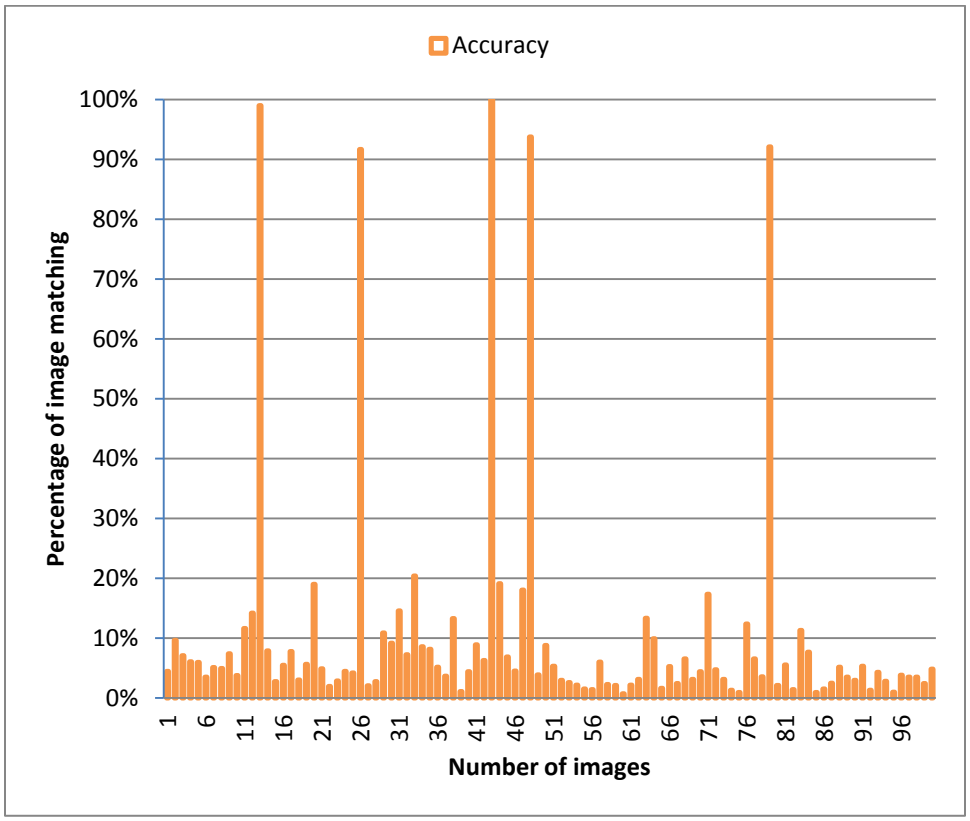

Fig. 4.7: The result of accuracy for 5 food images

Fig. 4.7 shows the result of accuracy for searching food image cases. This application can match the multiple food images in 100 images. The result of the SIFT algorithm describes key-points between images. Eventually, we detected 5 images using the SIFT method. In some case, we found 784 key-points out of 810 candidate features and its accuracy will be 96.7%. The accuracy of the 5 images would be over 92%.

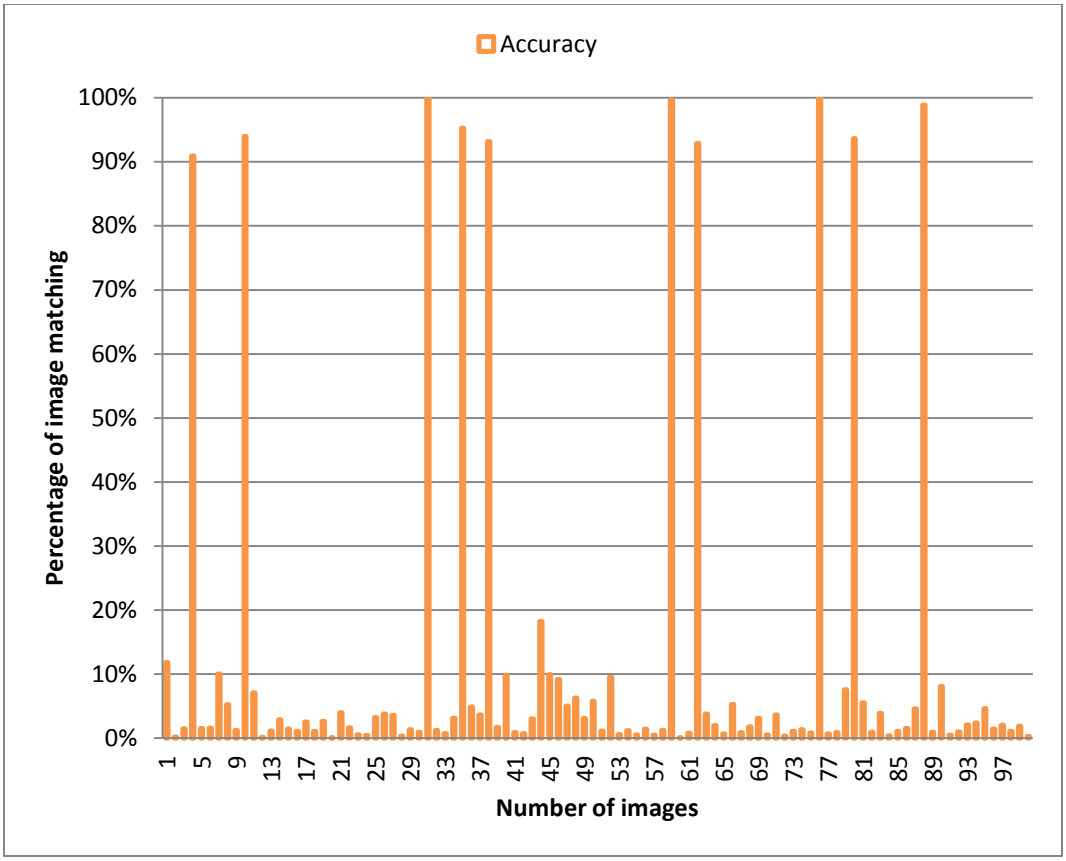

Fig. 4.8: The result of accuracy for 10 food images

Fig. 4.8 shows the result of the accuracy in 10 food image cases. This module performs the image matching for multiple food images. This application found 10 matched corresponding food images using the SIFT algorithm.

4.2 Bioimage application

We are applied an interactive image matching for MRI chest images [69, 70]. The biomedical images are provided by National Biomedical Imaging Archive (NBIA). We downloaded 100 biomedical MRI images of that are stored in the NBIA database. We can access and acquire the sample images at https://imaging.nci.nih.gov/ncia/login.jsf.

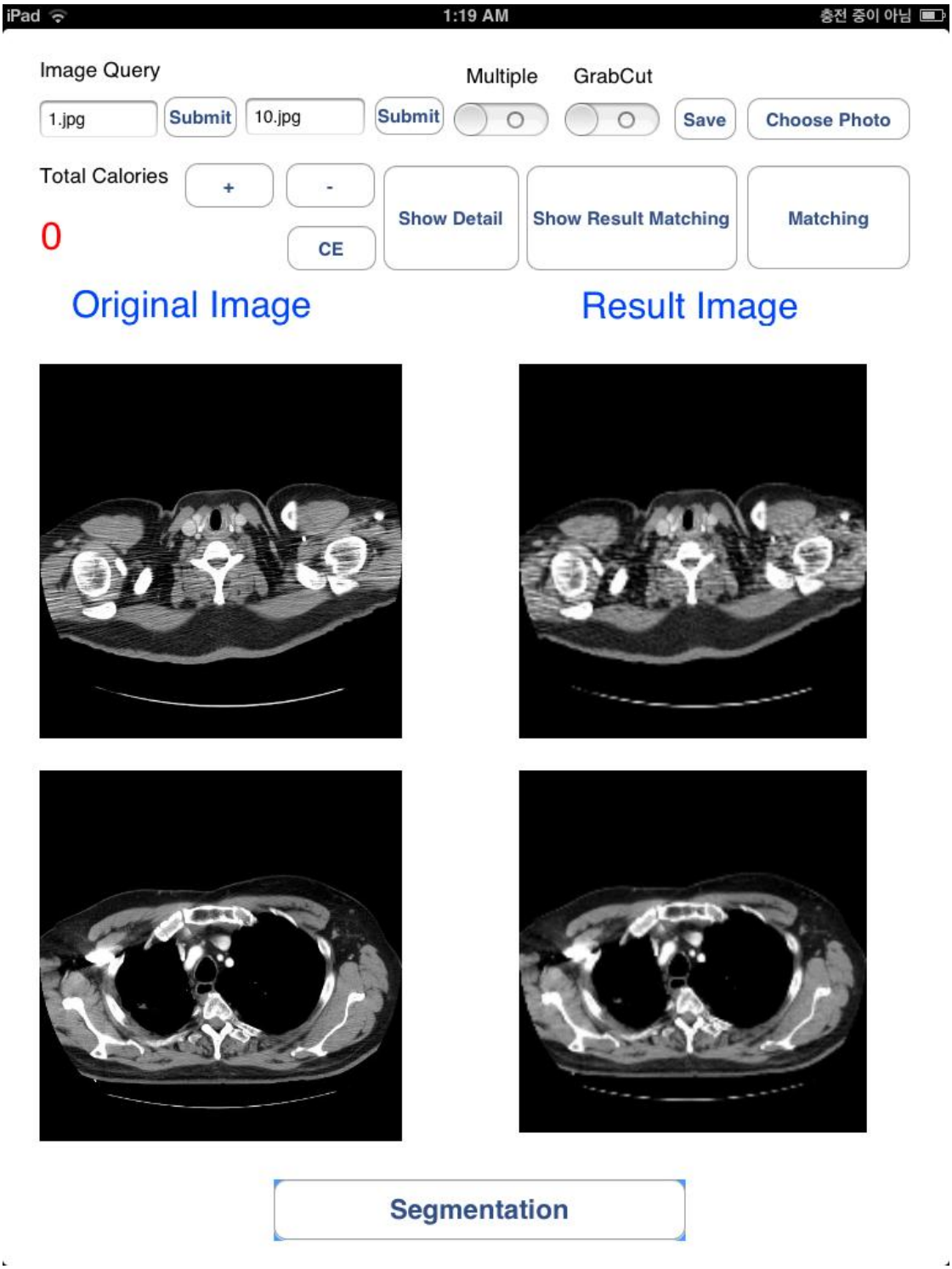

Fig. 4.9: Snapshot the GUIs on the iPad using the SIFT algorithm for bioimages

This snapshot above shows the overall GUI interface for biomedical images. The user can transfer the image to the server and perform the image matching by using the KD-tree and

RANSAC algorithms. The resulting image (right) is received from the server after finishing image matching [68]. At the time, the image will be converted from IplImage to UIImage when sending image to the client while image will be converted from UIImage to IplImage when receiving image from the server. The picture on the right side displays the information of the bioimages.

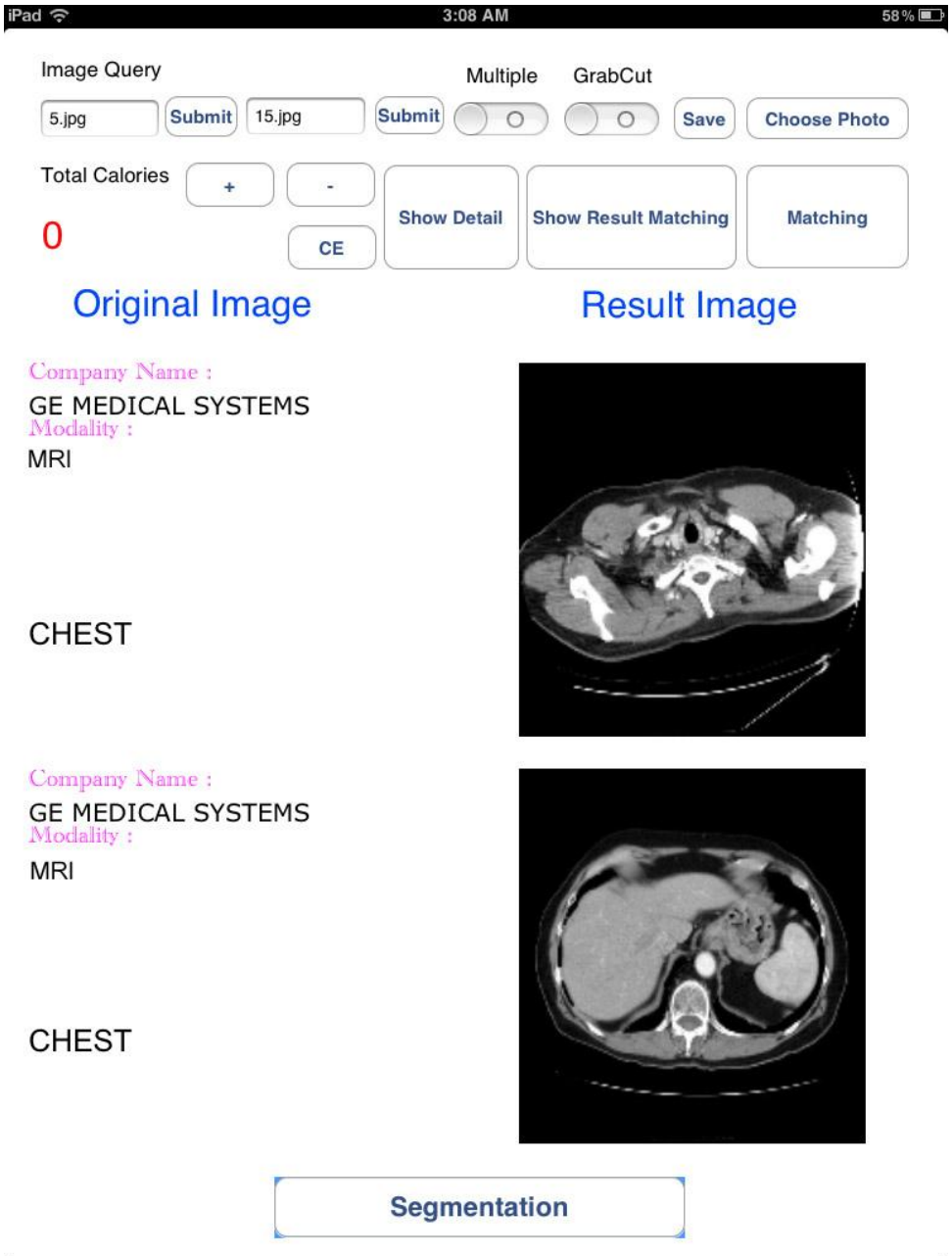

Fig. 4.10 Displays detailed information of the resulting image

Fig. 4.10 shows the detailed information for the resulting image which contains information like imaging system company name, modality, and organ name.

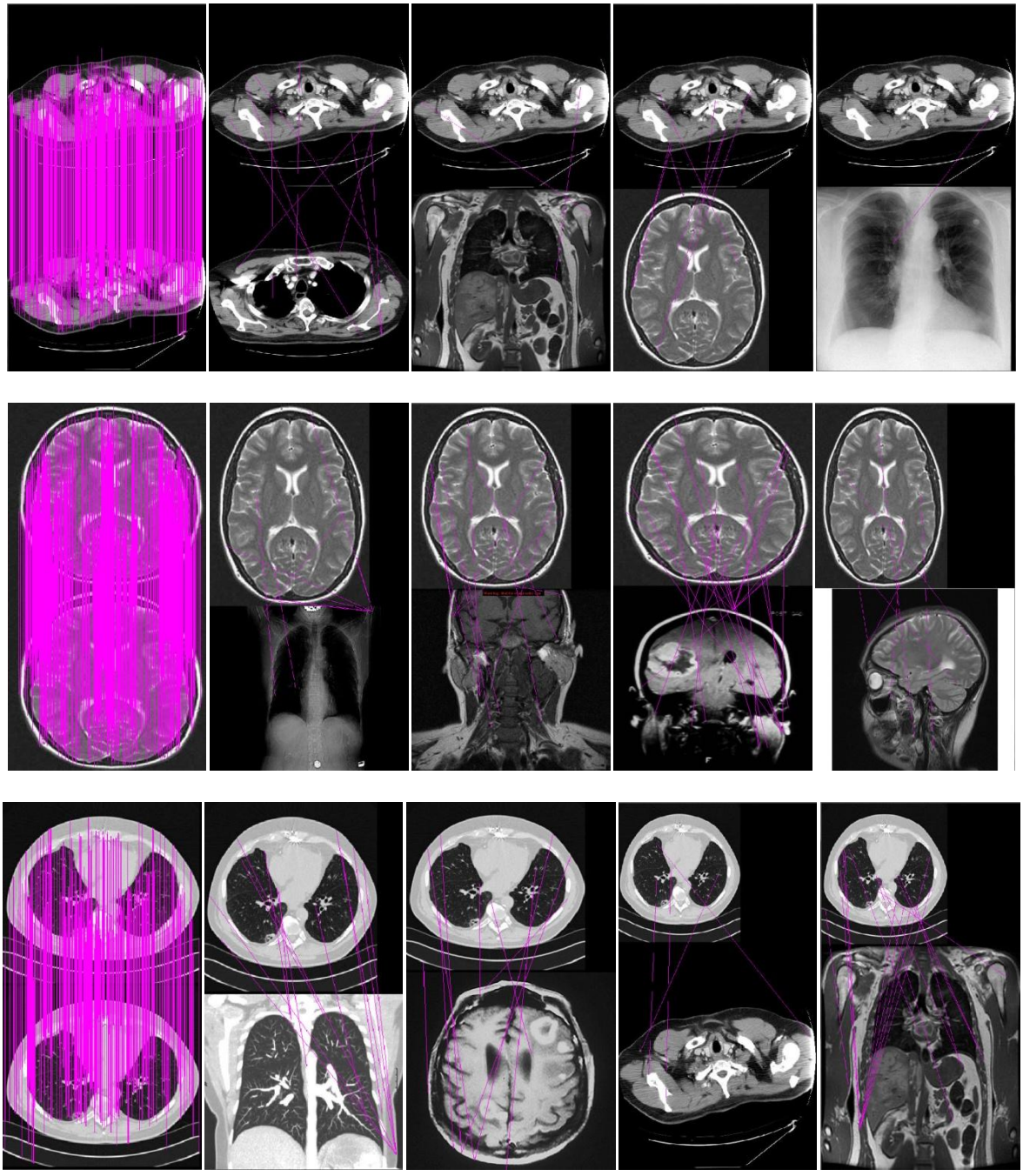

Fig. 4.11: Detecting key-points by using the SIFT algorithm for bioimages

Key-points of the SIFT algorithm are selected based on the measures of their localization for biomedical images, as shown in Fig. 4.11. These images are from the MRI imaging device.

The key-points can be detected by comparing image features based on shapes, scales, and locations. This image size is 500 by 500 pixels. For example, the first case, we detected the 1079 key-points in the 1091 candidate features of their images. It means that this accuracy will be 98.9% by using the Equation 4.1.

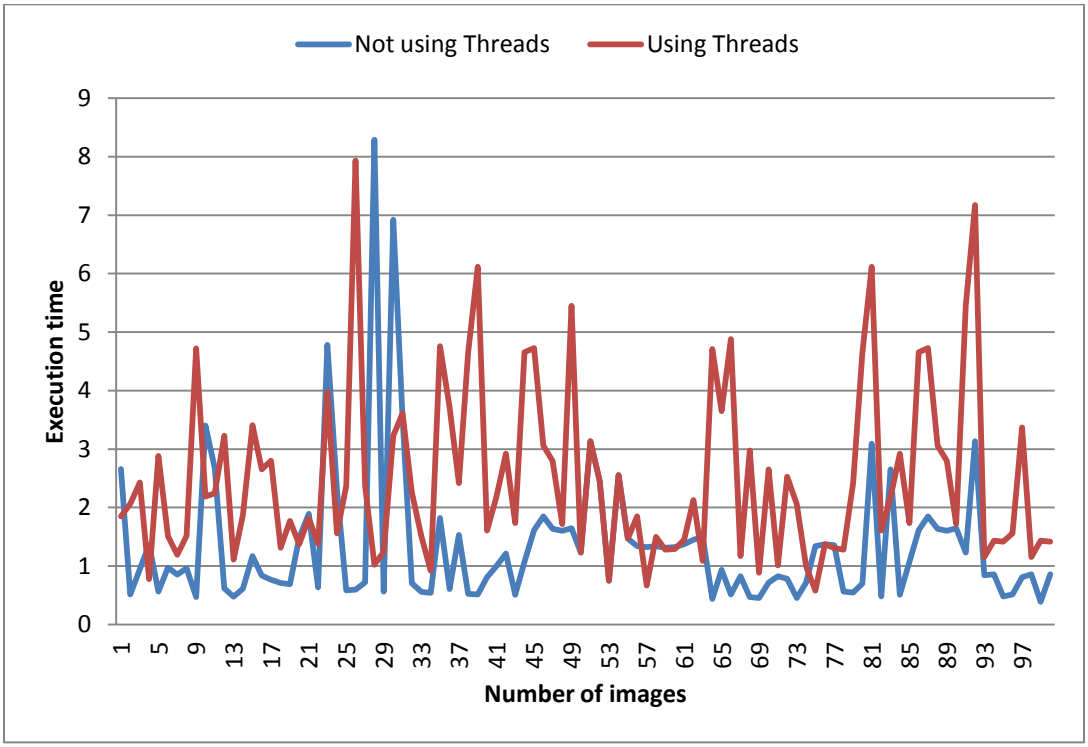

Fig. 4.12: Execution time between using threads and not using threads

Fig. 4.12 shows the runtime when the back-end module performs the SIFT algorithm between a query image and a reference image. As we can see the picture, the average runtime of multithreaded methods for image matching is higher than that using a single processor. When the server performs the image matching, the system needs more time to manage multiple threads.

Table 4.2: Overall runtime comparison for biomedical images

	Not using multiple threads	Using multiple threads
Overall Runtime	113.041 sec	57.295 sec

As we can see the Table 4.2, the overall runtime of the SIFT algorithm is increasingly improved when using multiple threads.

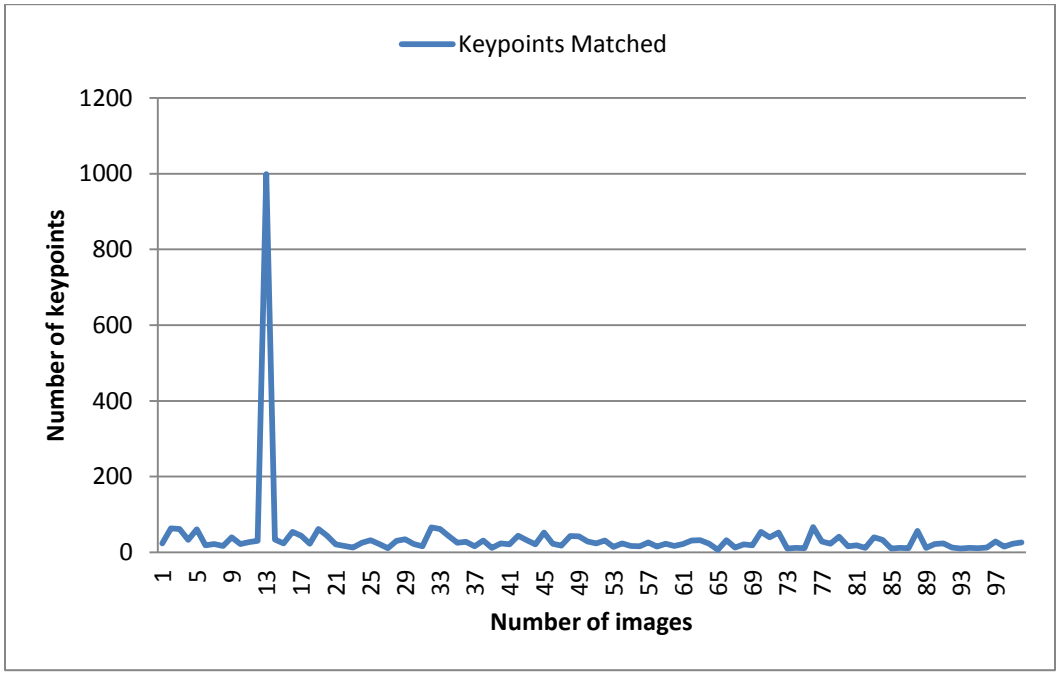

Fig. 4.13: Overall comparisons of key-points for biomedical images

Fig. 4.13 states that the SIFT algorithm detects key-points by using KD-tree and RANSAC algorithms. In the experiment, we used the MRI images from biomedical resources. Each image is very similar to others. As a result, the SIFT algorithm finds more key-points than food images. Since food images are not similar to each other.

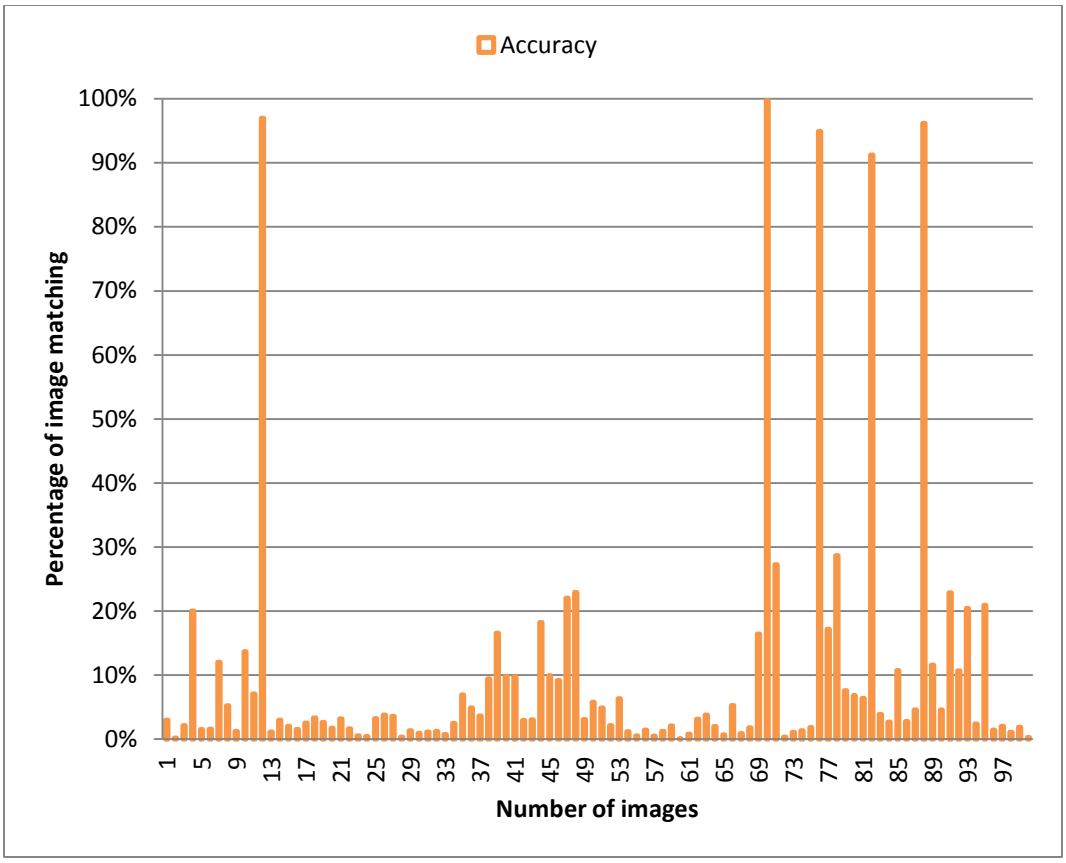

Fig. 4.14: The result of accuracy for 5 biomedical images

The result of the SIFT algorithm demonstrates key-points in their images as shown in Fig. 4.14 above. Fig. 4.14 shows the accuracy of the image matching for 5 biomedical image cases. This module performs to match images for multiple biomedical images. Eventually, we detected 5 images using the SIFT method. In some case, we found 999 key-points out of 1009 candidate features and its accuracy will be 99.0%. The accuracy of the 5 images would be over 90%.

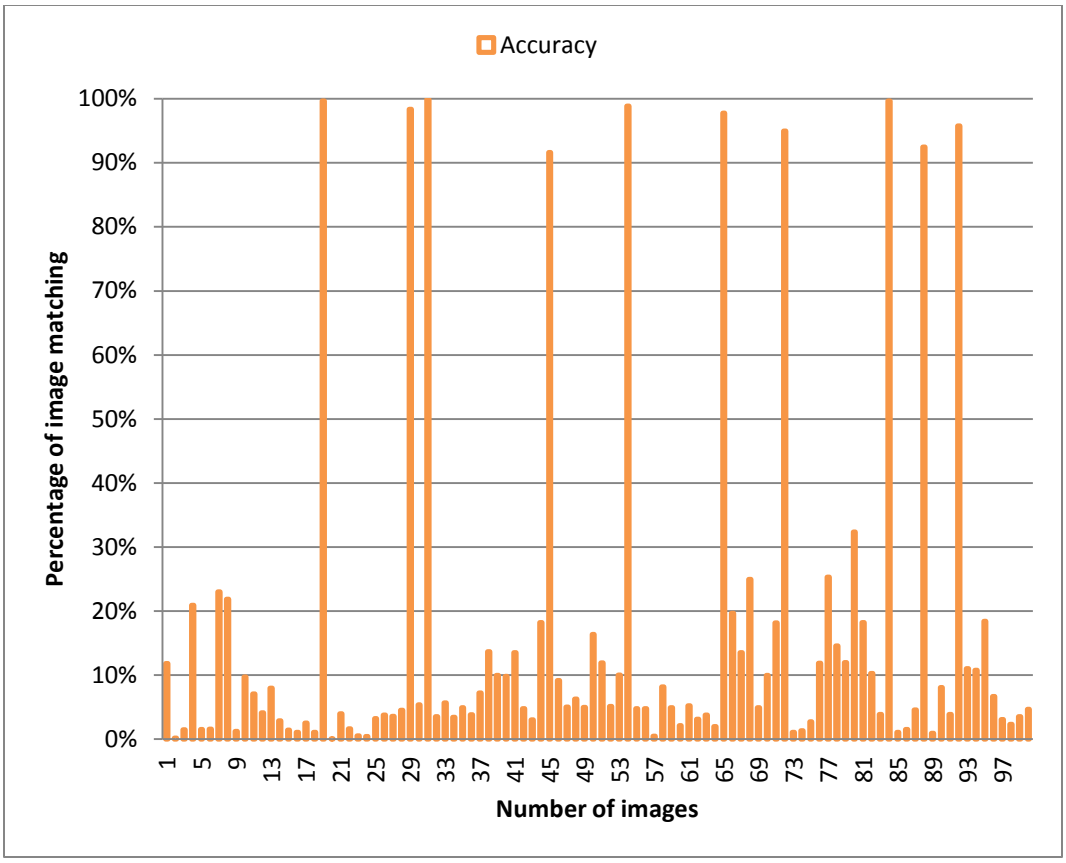

Fig. 4.15: The result of accuracy for 10 biomedical images

Fig. 4.15 shows the result of the accuracy for searching biomedical images. As we can see, the key-points of 10 matched images would be higher in 100 biomedical images. This application performs the SIFT algorithm for multiple biomedical images. This application found 10 matched corresponding biomedical images using the SIFT algorithm. For instance, if we found the 124 matched key-points among the 135 candidate key-points, the accuracy would be 91.8%.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this chapter, we describe the conclusion to summarize our work and propose the future work.

5.1 Conclusion

The open-source SIFT library is widely used in the computer vision field, which is a powerful algorithm to match objects in two different images. Based on the SIFT algorithm, Rob Hess released the SIFT implementation in 2006 as an open-source library. As it is introduced, many papers are published by using the SIFT algorithm and tried to improve its algorithm. Apple released a portable device, iPad 2, which provided a solid platform with faster graphics, WI-FI networks, high resolution screen, and touch technology. With this idea, we have designed and implemented the front-end and the back-end modules by using the SIFT algorithms for food and biomedical images.

The front-end module can perform to take an input image, to manipulate images using either GrabCut or rectangle-cut methods. This segmentation image will transfer to the back-end module using the CFNetwork based on the iOS platform. On the back-end module, the SIFT algorithm performs the image matching using the key-points can be extracted from a typical image. The image data and information will transfer to the front-end module using the Winsock library. Finally, all the information will display on the iPad.

We have investigated two applications for image matching: food and biomedical images. These applications are mainly based on the SIFT algorithm for comparing and finding between

different images. An application for food images will help people to manage their health and fitness. Moreover if they have a one photo, they can calculate calories of food how many calories they consume in a day. Also, they can control the food nutrient components using detailed information on the iPad. The application of the biomedical images provides an efficient and convenient way to compare between two images for biologists.

In conclusion, many applications will appear by using image matching algorithms relative to the food and biomedical images. With the iPad, an application will improve using the SIFT algorithms.

5.2 Future Work

Our approach is not supported by image control methods and manipulations. It only performs to mach in two images and displays the resulting image on the iPad. We need to add more functionalities and abilities to the manipulation of an object such as zoom-in, zoom-out, rotations, and movements. These functionalities will help users to control an image and gives detailed information. Moreover, they can help to easily recognize images. Assume that an image is damaged, these functionalities can help to identify the image by using zoom-in and rotation functions.

The application has to provide the image matching for video streaming image. Our module only can match pictures. The video can provide to collect a variety of images at one time while photo camera needs more time to collect pictures. This application can play a video, stop a play, and segment a specific image.

Finally, we propose the functions of multi-segmentation methods. Assume that four images are displayed on the iPad. We can segment numerous images in a short time by using the multi-segmentation functions. It will be an efficient and convenient way for users.

REFERENCES

1. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proc. in Alvey Vision Conference. (1988) pp. 147–151.
2. M. Zuliani, C. Kenney and B. S. Manjunath, A Mathematical Comparison of Point Detectors, Workshop on Image and Video Registration (IVR/CVPR Workshop), Issue 2, pp. 172{172, June 2004. (Provides a generalization of Harris-Stephens, Noble, and Shi and Tomasi corner detectors).
3. Chaohong Wu, Sergey Tulyakov and Venu Govindaraju, Robust point-based feature fingerprint segmentation algorithm, 2010.
4. Ryo Takei, A New Grey-Scale Template Image Matching Algorithm, July 2003.
   Using the Cross-Sectional Histogram Correlation Method Rakic, P., *Specification of cerebral cortical areas.* Science, 1988. **241**(4862): p. 170-176.
5. D. G. Lowe, Distinctive image features from scale-invariant keypoints. Intl. Journal of Computer Vision, 60(2):91,110, 2004.
6. OpenCV. http://opencv.willowgarage.com/.
7. Faraj Alhwarin, Chao Wang, Danijela Risti -Durrant, Axel Gr äser, *Improved SIFT-Features Matching forObject Recognition*, 2008.
8. Yuning Hua, *An improved SIFT feature matching algorithm,* WCICA, 2010. **7**: p. 6109-6113.
9. Soyel,H., Improved SIFT matching for pose robust facial expression recognition.FG, 2011. **3**: p. 585-590.

10. Y. Boykov, M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In Proc. IEEE Int. Conf. on Computer Vision, CD–ROM. 2001.

11. C. Rother, V. Kolmogorov, A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics (TOG). 2004.

12. Barbara Zitova, Jan Flusser. Image registration methods: a survey. Image and Vision Computing 21 (2003) 977–1000. 2003.

13. J. Liu, B.C. Vemuri, J.L. Marroquin, Local frequency representations for robust multimodal image registration, IEEE Transactions on Medical Imaging 21 (2002) 462–469.

14. L. Lucchese, G. Doretto, G.M. Cortelazzo, A frequency domain technique for range data registration, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2002) 1468–1484.

15. B.C. Vemuri, J. Ye, Y. Chen, C.M. Leonard, Image registration via level-set motion: Applications to atlas-based segmentation, Medical Image Analysis 7 (2003) 1–20.

16. http://blogs.oregonstate.edu/hess/code/sift/.

17. http://robwhess.github.com/opensift/.

18. Hanchuan Peng. Bioimage informatics: a new area of engineering biology. Bioinformatics.Vol. 24, No. 17, pp. 1827-1836. 2008.

19. Erica Sadun, The iPhone Developer's Cookbook second edition, Addison-Wesley.

20. https://developer.apple.com/xcode/.

21. http://developer.apple.com/library/ios/#documentation/uikit/reference/UITouch_Class/Reference/Reference.h

22. http://developer.apple.com/library/ios/#documentation/uikit/reference /UIResponder_Class/Reference/Reference.html.

23. C. Rother, V. Kolmogorov, A. Blake. GrabCut: interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics (TOG). 2004.

24. Gang Li, Tianming Liu, N. Jingxin, L. Guo, M. Andrew, S. Holley, J. Zhu, J. Chen, S.T.C. Wong. Segmentation of touching cell nuclei using gradient flow tracking. Journal of Microscopy, 231(1): 47-58. 2008.

25. http://www.apple.com/ipad/.

26. http://opencv.org/.

27. Gary Bradski, Adrian Kaehler. Learning OpenCV Computer Vision with the OpenCV Library. Publisher: O'Reilly Media. 2008

28. OpenCV 2.1 C++ Reference. http://opencv.willowgarage.com/documentation/cpp/index.html.

29. Computer Vision. http://www.computervisiononline.com/

30. J Goet, S Kiesler, A Powers. Matching Robot Appearance and Behavior to Tasks to Improve Human-Robot Cooperation. The 12th IEEE International Workshop on Robot and Human Interactive Communication, Vol., IXX, Oct. 31-Nov. 2, Milbrae, CA. pp. 55-60. 2003.

31. B. Leibe, A. Leonardis, and S. Bernt. Robust object detection with interleaved categorization and segmentation. IJCV, 77(1{3), 2008.

32. A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. IEEE TPAMI, 28(3), 2006.

33. S. Se, D. G. Lowe, and J. J. Little. Vision-based global localization and mapping for mobile robots. IEEE T-RO, 21(3), 2005.

34. Rob Hess. An Open-Source SIFT Library. MM'10 October 25–29. 2010

35. http://developer.apple.com/library/IOs/#documentation/CFNetwork/Reference/CFNetwork_Framework/_index.html.

36. https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaObjects/CocoaObjects.html.

37. https://developer.apple.com/library/mac/#documentation/CoreFOundation/Reference/CFSocketRef/Reference/reference.html.

38. Jurek Czyzowicz, Evangelos Kranakis, Jorge Urrutia. Rectilinear Glass-Cut Dissections of Rectangles to Squares.

39. Prosenjit Bose,Jurek Czyzowicz,Dominic Lessard.Cutting Rectangles Into Equal Area Pieces.

40. BOSE, P.K., CYZOWICZ , J., K RANAKIS , E., K RIZANC, D. and M AHESHWARI. Cutting circles and squares in equal area pieces to appear in FUN98. 1988.

41. Shoudong Han, Wenbing Tao, Desheng Wang Xue-cheng Tai, Xianglin Wu. Image Segmentation Based on GrabCut Framework Integrating Multi-scale Nonlinear Structure Tensor.

42. Multithreaded Programming Guide. SunSoft. 1994. http://www4.ncsu.edu/~rhee/clas/csc495j/MultithreadedProgrammingGuide_Solaris24.pdf.

43. http://developer.apple.com/library/ios/#documentation/uikit/reference/UIImagePickerController_Class/UIImagePickerController/UIImagePickerController.html.

44. William T. Freeman, David B. Anderson. Computer Vision for Interactive Computer Graphics. IEEE Computer Graphics and Applications. Volume 18, Number 3. 1998.

45. iOS Cocoa Touch. https://developer.apple.com/technologies/ios/cocoa-touch.html.

46. BLAKE, A., ROTHER, C., BROWN, M., PEREZ, P., AND TORR, P. Interactive Image Segmentation using an adaptive GMMRF model. In Proc. European Conf. Computer Vision.2004.

47. BOYKOV, Y., AND KOLMOGOROV, V. Computing Geodesics and Minimal Surfaces via Graph Cut. In Proc. IEEE Int. Conf. on Computer Vision. 2003.

48. KWATRA, V., SCHODL ¨ , A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. Proc. ACM Siggraph, 277–286.

49. MySQL Connector/C++ 5.1. http://dev.mysql.com/doc/refman/5.1/en/connector-cpp-getting-started-examples.html.

50. Winsock Reference. http://msdn.microsoft.com/en-us/library/windows/desktop/ms741416(v=vs.85).aspx.

51. MySQL 5.1 Reference Manual. http://dev.mysql.com/doc/refman/5.1/en/.

52. Windows Socket 2. http://msdn.microsoft.com/en-us/library/windows/desktop/ms740673(v=vs.85).aspx.

53. Mustafa Özuysal, Pascal Fua, Vincent Lepetit.Fast Keypoint Recognition in Ten Lines of Code.

54. Y. Amit. 2D Object Detection and Recognition: Models, Algorithms, and Networks. The MIT Press, 2002.

55. L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(4):594–611, 2006.

56. MFC Reference. http://msdn.microsoft.com/en-us/library/d06h2x6e(v=vs.80).aspx.

57. Carsten Rother, Vladimir Kolmogorov, Yuri Boykov, Andrew Blake. Microsoft Technical Report: MSR-TR-2011-46. Interactive Foreground Extraction using graph cut.

58. D. Singaraju, L. Grady, and R. Vidal. P-brush: Continuous valued mrfs with normed pairwise distributions for image segmentation. In Proc. IEEE Conf. Computer Vision and Pattern Recog., 2009.

59. M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. In Proc. European Conf. Computer Vision, 2008.

60. M. Unger, T. Pock, D. Cremers, and H. Bishop. Tvseg - interactive total variation based image segmentation. In Proc. British Machine Vision Conf., 2008.

61. H. Nickisch, P. Kohli, and C. Rother. Learning an interactive segmentation system. In Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP), 2010.

62. V. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-andmincut. In Proc. European Conf. Computer Vision, 2008.

63. Ryutarou Ohbuchi, Takahiko Furuya. Accelerating Bag-of-Features SIFT Algorithm for 3D Model Retrieval. 2010.

64. C. Wu, SiftGPU: A GPU Implementation of David Lowe's Scale Invariant Feature Transform (SIFT) , http://cs.unc.edu/~ccwu/siftgpu/.

65. Ziv Yaniv. Random Sample Consensus (RANSAC) Algorithm, A Generic Implementation. 2010.

66. Octavian Procopiuc, Pankaj K. Agarwal, Lars Arge, and Jeffrey Scott Vitte. Bkd-tree: A Dynamic Scalable kd-tree.

67. Stefan Popov,Johannes Günther, Hans-Peter Seidel, Philipp Slusallek. Stackless KD-Tree Traversal for High Performance GPU Ray Tracing. 2007.

68. Derek L G Hill, Philipp G Batchelor, Mark Holden and David J Hawkes. Medical image registration. 2000.

69. Van Herk M and Kooy H M 1994 Automated three-dimensional correlation of CT–CT, CT–MRI and CT–SPECT using chamfer matching Med. Phys. 21 1163–78.

70. Thacker N A, Jackson A, Moriarty D and Vokurka E 1999 Improved quality of re-sliced MR images using re-normalized sinc interpolation J. Magn. Reson. Imaging 10 582–8.

71. Y.-Y. Chuang, B. Curless, D. Salesin, R. Szeliski. A Bayesian approach to digital matting. In Proc. IEEE Conf. Computer Vision and Pattern Recog., CD–ROM. 2001.

72. E. Mortensen, W. BARRETT. Intelligent scissors for image composition. Proc. ACM Siggraph, 191–198. 1995.

73. Y. Boykov, M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In Proc. IEEE Int. Conf. on Computer Vision, CD–ROM. 2001.