DATA EXTRACTION FOR SCALE FACTOR DETERMINATION USED IN 3D-PHOTOGRAMMETRY FOR PLANT ANALYSIS

by

LEELA VENKATA NAGA SATISH ACHANTA

B.Tech., Jawaharlal Nehru Technological University, Kakinada, India, 2011

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2013

Approved by:

Major Professor
Dr. Mitchell L. Neilsen

Abstract

ImageJ and its recent upgrade, Fiji, are image processing tools that provide extensibility via Java plug-ins and recordable macros [2]. The aim of this project is to develop a plug-in compatible with ImageJ/Fiji, which extracts length information from images for scale factor determination used in 3-D Photogrammetry for plant analysis [5]. Plant images when processed using Agisoft software, gives an image consisting of the images processed merged into a single 3-D model. The coordinate system of the 3-D image generated is a relative coordinate system. The distances in the relative coordinate system are proportional to but not numerically the same as the real world distances. To know the length of any feature represented in 3-D model in real world distance, a scale factor is required. This scale factor when multiplied by some distance in the relative coordinate system, yields the actual length of that feature in the real coordinate system. For determining the scale factor we process images consisting of unsharpened yellow colored pencils which are all the same shape, color and size. The plug-in considers each pencil as a unique region by assigning unique value and unique color to all its pixels. The distance between the end midpoints of each pencil is calculated. The date and time on which the image file gets processed, name of the image file, image file creation and modification date and time, total number of valid (complete) pencils processed, the midpoints of ends of each valid pencil, length (distance) i.e., the number of pixels between the two end midpoints are all written to the output file. The length of the pencils written to the output file is used by the researchers to calculate the scale factor. Plug-in was tested on real images and the results obtained were same as the expected result.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to take this opportunity to thank my major professor Dr. Mitchell Neilsen for his valuable guidance and support throughout my project, my course work and my Master's program.

I would like to express my appreciation to my committee members, Dr. Daniel Andresen and Dr. Torben Amtoft for accepting to serve on my committee.

I wish to place on record, my deep sense of gratitude to my Supervisor, Dr. Stephen Welch, Department of Agronomy, for giving me the opportunity to work on this project. His constant support and guidance has been inspiring and indefatigable in accomplishing the given task.

Finally, I would like to thank CIS administrative and technical support staff, my friends for their support throughout my graduate study.

Chapter 1 - Introduction

1.1 Motivation

In an experiment conducted at the University of California at Davis, plants are grown in growth chambers under suitable environmental conditions and in fields. The plant features such as the growth of leaves, stem, etc need to be analyzed periodically. It's hard for the researches to measure the growth of plant features using a ruler or any other physical methods. The only way to analyze is from images. Today techniques in image processing have been increased to a large extent in such a way that features of an object can be analyzed from its image. The science of making measurements from photographs is called Photogrammetry [5]. So, the plants are photographed periodically and the images are sent to remote laboratories to analyze the growth of plants. The plant pictures (330000 of them over a six month period) and the pencil pictures are taken in the experiment at the University of California at Davis are forwarded to the laboratory for processing. In the laboratory a 3D model for the plant images is generated. But to know the length of any feature in the 3D model image in real world measurements a scale factor is required. This motivated me to develop this plug-in which extracts the data required from the image for calculating the scale factor. This data is used by the researchers to calculate the scale factor used in 3D Photogrammetry for plant analysis.

1.2 Objective

The main objective of the project is to develop a plug-in, which extracts length information from images for scale factor determination used in 3-D Photogrammetry for plant analysis. This plug-in can be integrated to work with image processing tools like ImageJ/Fiji. The plug-in processes an image consisting of unsharpened yellow colored pencils, with a metal foil and an eraser tip at their ends, which are all the same shape, color and size. Pencils were selected as targets because they are highly uniform, readily available, and very inexpensive. Each input image to be processed contains a certain number of pencils placed randomly. Pencils are arranged such that there is no contact between them. The plug-in processes a series of images in a folder at a time. Each input image, the plug-in processes is under the surveillance of two cameras. So some images also contain pencils which fall over the edges of the image (incomplete pencils). The plug-in should not consider these pencils. Plug-in should consider only the yellow region of the pencil and output the midpoint coordinates of the ends of pencil. Using the midpoint coordinates of each pencil the distance between two end midpoints is calculated. This process continues for all the pencils in the image except the pencils which fall over the edges of the image (incomplete pencils). Plant images when processed using Agisoft software, gives an image consisting of the images processed merged into a single 3-D model. This coordinate system of the 3-D image is a relative coordinate system. The distances in the relative coordinate system are proportional to but not numerically the same as the real world distances. So, based on the distance between the ends midpoints of each pencil (length) obtained from the images processed by the plug-in, a scale factor is calculated. This scale factor determined from the pencil pictures, when multiplied by some distance in the relative coordinate system, yields the actual length of that feature in the real coordinate system. The scale factor can be applied to all the other distances in the 3-D model.

1.3 Modes of Operation

Plug-in is intended to have two modes of operation depending on the name of the folder specified. If the name of the folder is 'Calibration' the plug-in processes the images present in the folder 'Calibration'. The plug-in will also be used to develop scale factors for small 3D models of plant inflorescences ("Inflorescences" are stalks on plants containing their flowering parts) [7]. The calibration activity will only need to be done once but the flowering part will happen many times as different plants flower. If the name of the folder is 'Flower' then the plug-in processes all the sub folders in 'Flower' which are recently added to it. There is a chance that additional image files will be added to the subfolders created on the day the plug-in is invoked. So, plug-in does not process the folders which are created on the day the plug-in is invoked. Each time the plug-in completes processing the folders in 'Flower', it keeps track of the date and time at which 'Flower' is processed. Based on the tracked time and date, only the newly added folders in 'Flower' are processed for the next execution. The images in the folder 'Calibration' are processed only once before the experiment. All the images in the folder 'Calibration' contains only pencils where as images in the folder/subfolders of 'Flower' contain both plants and pencils.

Chapter 2 - Implementation

2.1 Working Environment

Image processing tools: This plug-in works with Image Processing tools like ImageJ and Fiji. ImageJ is a public domain java based image processing program developed by National Institute of Health [2]. Fiji is an Open Source image processing package based on ImageJ. Fiji is described as a distribution of ImageJ together with Java, Java 3D and lot of plug-ins organized into a coherent structure [1]. ImageJ/Fiji plugins are java classes that implements two different kinds of plug-in interfaces namely, PlugIn and PlugInFilter. PlugIn requires no image to be opened to start the plug-in. With PlugInFilter the currently active image is passed to the plug-in when started. These interfaces must implement at least two methods: setup() and run(). These methods have the following signatures:

- int setup (String arg, ImagePlus im)

When the plug-in is started this method is called first to verify that the capabilities of this plug-in match the target image. ImagePlus object represents an image that can be displayed on screen

- void run (ImageProcessor ip)

Actual implementation for the plug-in is done by this method. All the code to be implemented is written/called in this method. It takes an object of type ImageProcessor which contains the image to be processed and all relevant information about it [3].

Operating System Used: Windows 7

Technologies Used: Java

IDE used: Eclipse

Valid Formats of Images the plug-in processes: BMP, JPEG, TIFF

For the plug-in to work, the plug-in should be placed in Plugins folder of ImageJ/Fiji.

2.2 Image Data

This is data of the image written to the output file. An output file with the name as the name of the folder is created in the folder the plug-in currently access. For each image file in the folder accessed the following data, date and time on which the image file gets processed, name of the image file, image file creation and modification time and date, total number of valid pencils processed, the midpoints of ends of each complete pencil, length (distance) i.e., the number of pixels between the two end midpoints are written to the output file.

2.3 Implementation

The plug-in is implemented in java. The processing of all the images is automated and the plug-in displays a message "Task Completed" upon completion of execution. Plug-in is implemented using the four modules.

- Scale Calibration
- Pencil Detection
- Connected Regions
- Length Extraction

2.3.1 Scale Calibration

This is the first module which gets executed after invoking the plug-in. This module implements PlugIn interface. The plug-in is intended to process only images of specified format. It does not process the files other than the specified format. Initially a JFileChooser dialog box is displayed [8].

If the user selects the folder 'Calibration' to process then initially a text file with the name of folder selected (Calibration) is created in the folder selected (Calibration). This text file contains the outputs of all the images processed in the folder 'Calibration'. Each file in the folder is first checked whether it is in the valid scope or not. If it is a valid file then the date and time on which the image gets processed, name of the image, image creation and modification time and date are written to the output file and the image is passed as an argument to the module Pencil Detection as an ImageProcessor object for further processing. The date and time are in Excel date format [9]. Pencil Detection passes the ImageProcessor object of image after processing to

Length Extraction. In Length Extraction the image is processed and total number of valid pencils processed the midpoints of ends of each complete pencil, length (distance) i.e., the number of pixels between the two ends midpoints are written to the output file. Once processing of the image is completed all the windows of the processed image are closed automatically and the next file in the folder is chosen for processing.

If the user selects the folder 'Flower', this folder contains several subfolders and may also contain individual image files. In the folder 'Flower' a text file named 'LastAccessed.txt' is created which keeps track of the date and time on which the folder 'Flower' is last accessed. Each time 'Flower' is accessed 'LastAccessed.txt' is updated with the date and time of recent access. Each subfolder/file is processed one by one. The plug-in should not process the subfolders which are created on the same day the plug-in is invoked i.e., the date difference (days) should be at least one. Because, there may be a chance that additional image files can be added to the subfolders created on the day the plug-in is invoked. 'Flower' will not contain the file 'LastAccessed.txt' when it is accessed for the first time. Whenever 'Flower' is accessed, if 'Flower' does not contain 'LastAccessed.txt' then each subfolder/file is processed only if the creation date of the subfolder/file is less than the day on which the plug-in is invoked i.e., the date difference(days) should be at least one. If 'Flower' contains 'LastAccessed.txt' then each subfolder/file is processed only if the creation date of the subfolder/file is less than the day on which the plug-in is invoked and greater than or equal to the date of last access stored in 'LastAccessed.txt'. This process repeats for all the subfolders/file.

For each subfolder selected to process, a text file with the name of subfolder selected is created in it. This text file contains outputs of all the images processed in the subfolder selected. Each file in the subfolder is first checked whether it is in the valid scope or not. If it is a valid file then the date and time on which the image gets processed, name of the image, creation and modification time and date of the image are written to the output file and the ImageProcessor object of the image is passed as an argument to the module Pencil Detection for further processing. Pencil Detection sends the ImageProcessor object after processing to Length Extraction. In Length Extraction the image is processed and the total number of valid pencils processed the midpoints of ends of each complete pencil, length (distance) i.e., the number of pixels between the two ends midpoints are written to the output file. Once processing of the image is completed all the windows of the processed image are closed automatically and the next file in the folder is considered for processing.

For the individual image files in 'Flower' a text file named 'Flower.txt' (name of the folder) is created in 'Flower'. Each image file is sent to Pencil Detection, Length Extraction for further processing and Image data is written to the output file "Flower.txt". Once processing of all the files/folders in 'Flower' is done 'LastAccessed.txt' is updated with the date and time of the recent access of 'Flower'.

Plug-in also has the capability to process a single image files selected manually by the user from the JFileChooser dialog box. The plug-in responds with a message "Invalid File. Select only valid Image Files" when any individual file which is not in the valid format is selected. If the user selects a folder and the folder is neither 'Calibration' nor 'Flower' or not any folder in the folder 'Flower' then the plug-in responds with a message "Invalid File".

2.3.2 Pencil Detection

This module implements PlugInFilter interface. Here for the image obtained from Scale Calibration, all the yellow pixels in the image are converted to white and the remaining pixels are replaced by black pixels. The resulting image is converted to grey scale image and the reference of the image is sent as an argument to Connected Regions. The image obtained as an output from Connected Regions is sent as an argument to Length Extraction for performing length extraction. Figure 2.1 shows the HSV color space [3].

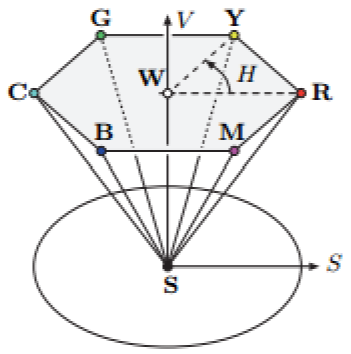
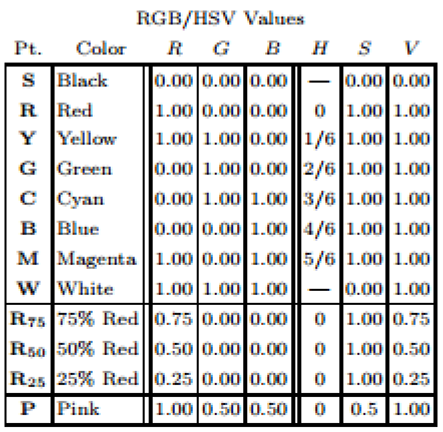


Figure 2.1 HSV Color Space [3]



RGB/HSV Values

Pt.	Color	R	G	B	H	S	V
S	Black	0.00	0.00	0.00	—	0.00	0.00
R	Red	1.00	0.00	0.00	0	1.00	1.00
Y	Yellow	1.00	1.00	0.00	1/6	1.00	1.00
G	Green	0.00	1.00	0.00	2/6	1.00	1.00
C	Cyan	0.00	1.00	1.00	3/6	1.00	1.00
B	Blue	0.00	0.00	1.00	4/6	1.00	1.00
M	Magenta	1.00	0.00	1.00	5/6	1.00	1.00
W	White	1.00	1.00	1.00	—	0.00	1.00
$\mathbf{R_{75}}$	75% Red	0.75	0.00	0.00	0	1.00	0.75
$\mathbf{R_{50}}$	50% Red	0.50	0.00	0.00	0	1.00	0.50
$\mathbf{R_{25}}$	25% Red	0.25	0.00	0.00	0	1.00	0.25
P	Pink	1.00	0.50	0.50	0	0.5	1.00

Figure 2.2 RGB-HSV Values [3]

Code snippet for converting yellow pixels to white: This code snippet is executed for all the pixels in the image. Pixels other than yellow are converted to black.

```
int[] RGB = new int[3];
float[] HSV = new float[3];
ip.getPixel(a, b1, RGB);
Color.RGBtoHSB(RGB[0], RGB[1], RGB[2], HSV);
if((HSV[0]>=0.07 && HSV[0]<=0.18) && (HSV[1]>=0.25 && HSV[1]<=1.0)
&& (HSV[2] >=0.26 && HSV[2]<=1.0)  )
{
RGB[0]=255;
RGB[1]=255;
RGB[2]=255;
ip.putPixel(a, b1, RGB);
}
else
{
RGB[0]=0;
RGB[1]=0;
RGB[2]=0;
ip.putPixel(a, b1, RGB);
}
}
```

2.3.3 Connected Regions

This module implements PlugIn interface. ImageJ/Fiji has a plug-in called "Find Connected Regions" which finds clusters of voxels in images based on some simple criteria and separates out those clusters into images [4]. "Find Connected Regions" is available in VIB-lib.jar file. We make a call to "Find Connected Regions" in Connected Regions. The resulting image contains each pencil identified as a separate region by assigning a unique color to each region and a unique value for all the pixels of that region. This image is sent to Length Extraction as an ImageProcessor object for extracting the length information.

"Find Connected Regions" takes 12 values as input arguments [4]. The 12 arguments are:

- ImagePlus: Image to be processed is processed as an ImagePlus object.
- Allow diagonal Connections: This argument is sent as a Boolean value. If this argument is set to true then two voxels that meet all the other criteria will be considered to be connected even if they only touch at the corners or along one edge. In our program we set the value of this Boolean variable to true.
- Display an image for each region: This argument is sent as a Boolean value. If this argument is set to true then each pencil which is considered as a separate region is displayed in a separate image. As our plug-in function in the background we do not require a separate image for each pencil. So, we set this argument to false indicating to show all the pencils as separate regions in a single image.
- Display one image for all regions: This argument is sent as a Boolean value. This argument is set to true indicating to show all the pencils as separate regions in a single image.
- Display results table: This argument is sent as a Boolean value. This argument is set to false as the user doesn't require information about each region.

- Regions must have same value: This argument is sent as a Boolean value. As all the pencils are of same color and dimensions(unique), to classify them as separate pencils(region) this argument is set to false.
- Start from point Selection: This argument is sent as a Boolean value. By setting this argument to true we can specify that the search for the first cluster should start at a particular point. But the execution process is automated, the plug-in should not ask for inputs from the user during the execution. So this argument is set to false. The plug-in starts its search from the first pixel in the image.
- Auto subtract discovered regions from original image: As the name indicates if this argument is set to true then the regions discovered will be removed from the original image. So, this argument is set to false.
- Regions for values over: This argument is sent as a double value. This value is set as 100D.
- Minimum number of points in a region: This argument is sent as a double value. To avoid unwanted regions other than pencil regions we need to specify the minimum number of points(pixels) in a region in order for the plug-in to consider the cluster as a separate region. This argument is set to 20000D.
- Stop after this number of regions are found: This argument is sent as a integer value. The user doesn't know the number of complete pencils in the image. So this argument is set to -1. The plug-in stops only after finding all the valid regions based on the values of other arguments.
- Default: This argument is sent as a Boolean value with its default predefined value set to false.

2.3.4 Length Extraction

This module implements PlugInFilter interface. The image obtained from Pencil Detection is taken as an ImageProcessor object for extracting length information for each pencil in the image. This image contains each pencil identified as a separate region by assigning a unique color to each region and a unique value 'V' for all the pixels of that region. Incomplete pencils in the image are not considered for length extraction. The midpoint coordinates of the two ends of each complete pencil and the distance between the two end midpoints i.e., length is written to the output file. Length of the pencil is the total number of pixels along the axis between the two end midpoints. Based on the lengths of all the complete pencils in the image a scale factor is calculated. This scale factor, when multiplied by some distance in the relative coordinate system, yields the actual length of that feature in the real coordinate system. The process of Length Extraction is as follows:

For each region (complete pencil) 'R' selected in the image, the following steps are used to extract length information.

Step 1. Centroid of 'R' is calculated. Centroid is calculated using the following equations:

cx = u/count
cy = v/count

u - Sum of x-coordinates of all the pixels in the region R
v - Sum of y-coordinates of all the pixels in the region R
count - Total number of all the pixels in the region R
cx - x-coordinate of Centroid
cy - y-coordinate of Centroid

Step 2. For each pixel 'P' in the region Central moments(μ) are calculated. Central moments are calculated using the following equation [3]:

$$\mu_{pq}(R) = \sum (P_x - cx)^p \cdot (P_y - cy)^q \quad \text{for all } (P_x, P_y) \text{ belongs to R}$$

P_x = x-coordinate of the pixel 'P'
P_y = y-coordinate of the pixel 'P'
p,q are the orders for the central moment μ

We need to calculate central moments $\mu_{11}, \mu_{20}, \mu_{02}$

$\mu_{11} = (P_x - cx) \cdot (P_y - cy)$

$\mu_{20} = (P_x - cx)^2$

$\mu_{02} = (P_y - cy)^2$

Sum of Central moments of all the pixels 'P' in a region 'R' are calculated. Sum of central moments μ_{11} of all the pixels in 'R' is again taken as μ_{11}. Sum of central moments μ_{20} of all the pixels in 'R' is again taken as μ_{20}. Sum of central moments μ_{02} of all the pixels in 'R' is again taken as μ_{02}.

Step 3: Orientation vectors specify the direction of the major axis which is the axis that runs through the Centroid and along the widest part of the region. Orientation vectors (x_d, y_d) for the region 'R' are calculated using the following equations [3]:

$$x_d = \cos\theta = 0 = \begin{cases} 0 & \text{for } A = B = 0 \\ \left[\frac{1}{2}\left(1 - \frac{B}{\sqrt{A^2+B^2}}\right)\right]^{\frac{1}{2}} & \text{otherwise} \end{cases}$$

$$y_d = \sin\theta = 0 = \begin{cases} 0 & \text{for } A = B = 0 \\ \left[\frac{1}{2}\left(1 - \frac{B}{\sqrt{A^2+B^2}}\right)\right]^{\frac{1}{2}} & \text{for } A \geq 0 \\ \left[\frac{1}{2}\left(1 - \frac{B}{\sqrt{A^2+B^2}}\right)\right]^{\frac{1}{2}} & \text{for } A < 0 \end{cases}$$

where,

$A = 2 \cdot \mu_{11}$

$B = \mu_{20} - \mu_{02}$

$C = \sqrt{A^2 + B^2}$

Step 4: In this step we take a constant called 'λ' (lambda). Using the orientation vectors, find a point 'P' at a distance 'λ' from Centroid. From point 'P', using the orientation vectors back trace towards Centroid along the major axis of the region until we find a pixel 'M' whose value is equal to unique value 'V' of the region 'R' and its next pixel along the major axis is a black pixel. 'M' is midpoint of one end of the region.

'λ' can be taken as the value of major axis of region 'R' calculated using the following equations [3]. For every region 'λ' is approximately equal to the length of major axis of region 'R'. So, in order to reduce the computation time in calculating the length of major axis for every region, a constant value of '800' is taken as 'λ' as this value is approximately equal to the length of major axis of region 'R'.

$$D = \mu 20 + \mu 02 + \sqrt{(\mu 20 - \mu 02)^2 + 4 \cdot \mu 11^2}$$

$$Major\ axis = \left(\frac{2 \cdot D}{count}\right)^{\frac{1}{2}}$$

Arithmetic operator '+' indicates the direction along the major axis. The following equations are used to traverse along the major axis in one direction [3]. lambda is initialized to 800. By decrementing the value of lambda by '1' every time, (x, y) and (x1, y1) are calculated. (x, y) is the midpoint 'M' of the end of the pencil when the value of the pixel at (x, y) is equal to 'V' and the pixel value at (x1, y1) is '0'(black pixel). (x, y) is written to the output file.

$$x = cx + \lambda \cdot x_d$$

$$y = cy + \lambda \cdot y_d$$

$$x1 = cx + (\lambda + 1) \cdot x_d$$

$$y1 = cy + (\lambda + 1) \cdot y_d$$

By using the operator '-' we traverse the region along the major axis in the opposite direction to get the midpoint of other end of the region 'R'. The following equations are used to traverse along the major axis in the opposite direction [3]. lambda is initialized to 800. By decrementing the value of lambda by '1' every time, (x, y) and (x1, y1) are calculated. (x, y) is the midpoint 'M1' of the end of the pencil when the value of the pixel at (x, y) is equal to 'V' and the pixel value at (x1, y1) is '0'(black pixel). (x, y) is written to the output file.

$$x = cx - \lambda \cdot x_d$$

$$y = cy - \lambda \cdot y_d$$

$$x1 = cx - (\lambda + 1) \cdot x_d$$

$$y1 = cy - (\lambda + 1) \cdot y_d$$

Next, the distance(length) between the two points 'M' and 'M1' is calculated. Length of the pencil is the total number of pixels along the axis between the two end midpoints. Coordinates of the midpoints and length are written the output file.

Chapter 3 - Testing

3.1 Unit Testing

In unit testing, various modules have been tested individually. This has been done manually to find bugs and to test the functionality of the plug-in. The testing was done with the help of the test cases below.

Sr. No	Test Module	Expected	Result
1	Invoke the plug-in	As soon as the plug-in is invoked a JFileChooser dialog box should be displayed	Pass
2	User opens the folder Calibration/Flower from the dialog box and the "Files of type" option is set to Image Files only	Only the image files and the folders with image files in Calibration/Flower should be visible to user	Pass
3	User opens the folder Calibration/Flower from the dialog box and the "Files of type" option is set to All Files	All the files in Calibration/Flower should be visible to user	Pass
4	On clicking Cancel button in the dialog box	Plug-in should not execute	Pass
5	Selected folder contains image files of different formats other than required formats and also contains	Plug-in process only valid image files	Pass

	text files		
6	A folder is selected for execution	A text file with name of the selected folder is created and image data of all the valid image files executed is written to the to the output file	Pass
7	An individual image file is selected	If the image is valid then a text file with name of the image file is created and image data is written to the to the output file	Pass
8	Plug-in takes an image file for processing	All the windows opened during processing of the image are closed automatically by the plug-in	Pass
9	User selects a folder and the folder is neither 'Calibration' nor 'Flower' or not any folder in the folder 'Flower'	Plug-in responds with a message "Invalid File"	Pass
10	Formats of processed date, creation date, modification date	Dates are written to the output file in excel date format and are valid	Pass
11	Image contains incomplete pencils	Plug-in will not process/consider incomplete pencils	Pass
12	Image data written to the output file	This data is in match with the results tested manually	Pass
13	Exception	No exceptions caught	Pass
14	Upon completion of plug-in execution	Output file for each folder/file contains image data of all the images the plug-in processes in the folder/file	Pass
15	When execution of plug-in is aborted	Output file created for the selected folder/file is empty	Pass

16	Upon completion of plug-in execution	"Task Completed" message is displayed on screen	Pass

Table 3.1 Unit Testing

3.2 Integration Testing

After testing of the individual modules, all the modules are made to communicate in collaboration and tested if they work successfully on integration or not. The integration testing was done with the help of the following test cases.

Sr. No	Test Case	Expected	Result
1	Plug-in is invoked and 'Flower' is selected for the first time	'LastAccessed.txt' is created. Plug-in processes all the file/folders in 'Flower' and the text file is updated with the date and time of recent access	Pass
2	Plug-in is invoked and 'Flower' is selected. Plug-in has already processed 'Flower' before.	'LastAccesssed.txt' is not empty. Plug-in process only the new added folders/files whose creation date is less than the day on which the plug-in is invoked and greater than or equal to the date of last access stored in 'LastAccessed.txt'. 'LastAccessed.txt' is updated with the date and time on which plug-in is invoked	Pass
3	Plug-in is invoked and user makes a selection	For each image processed, plug-in makes sure that all the four modules of implementation are executed automatically and all images are processed automatically without user intervention. When execution is completed "Task completed" message is shown on screen	Pass

Table 3.2 Integration Testing

3.3 Performance Factors

Factors that influence the performance of a java program are:

1. Memory Consumption of Java Program
2. Total runtime of the Java Program

On executing plug-in on Local Machine, the values of performance factors are:

Memory consumption: 4MB

Total runtime: depends on the number of pencils in the image file selected

The graph plotted with Number of pencils in an image on x-axis and Total runtime on y-axis is shown in figure 3.1

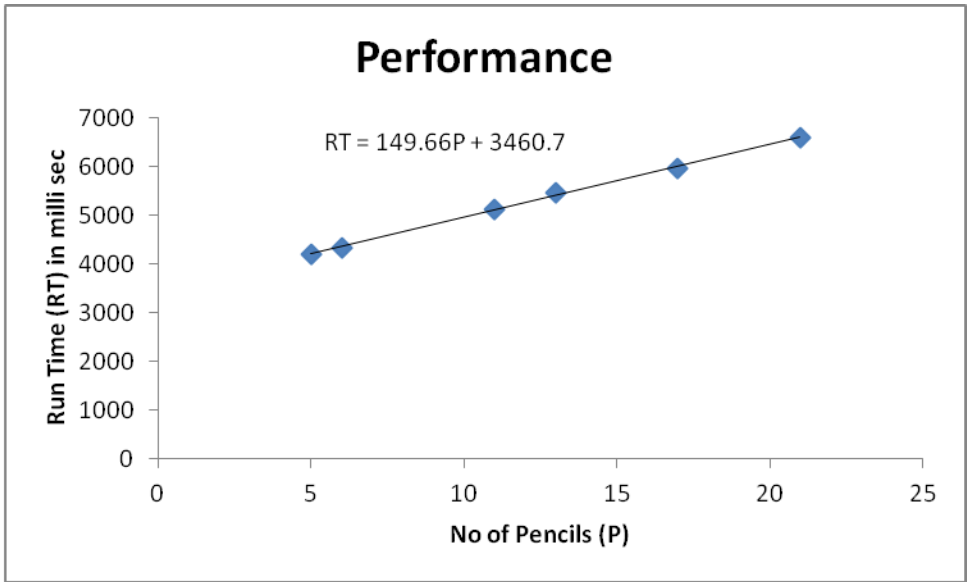
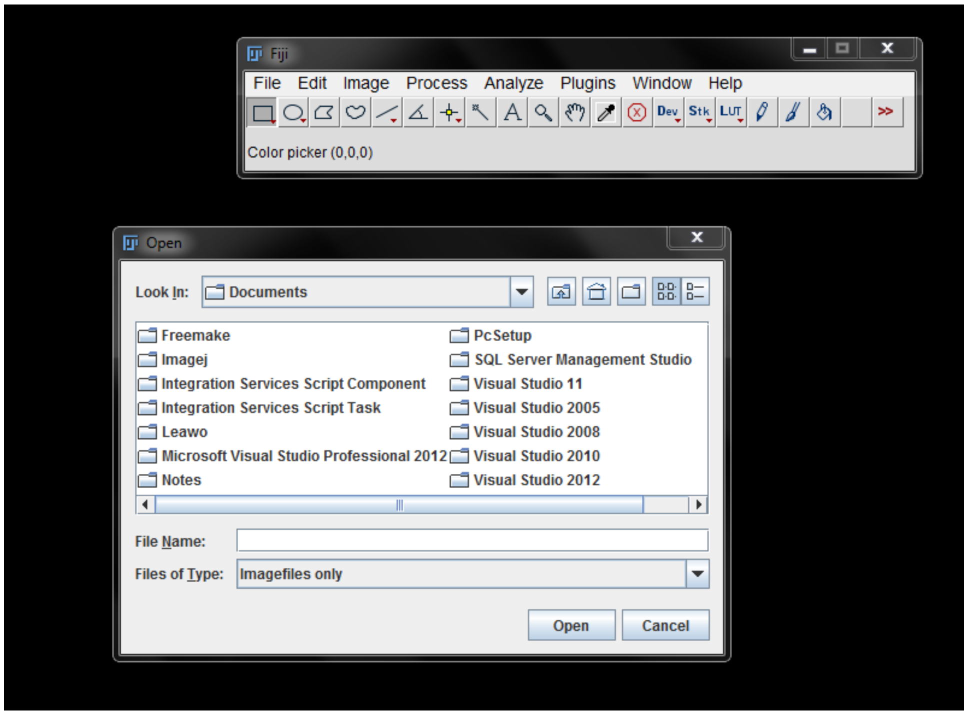


Figure 3.1 Performance

Chapter 4 - Results and Analysis

4.1 Processing Steps

JFileChooser Dialog Box(GUI):



Color picker (0,0,0)

Look In: Documents

Freemake	PcSetup
Imagej	SQL Server Management Studio
Integration Services Script Component	Visual Studio 11
Integration Services Script Task	Visual Studio 2005
Leawo	Visual Studio 2008
Microsoft Visual Studio Professional 2012	Visual Studio 2010
Notes	Visual Studio 2012

File Name:

Files of Type: Imagefiles only

Figure 4.1 JFileChooser Dialog Box

When the plug-in is invoked JFileChooser dialog box is displayed. User can browse through the directories to select a file/folder.

Step 1: Input Image

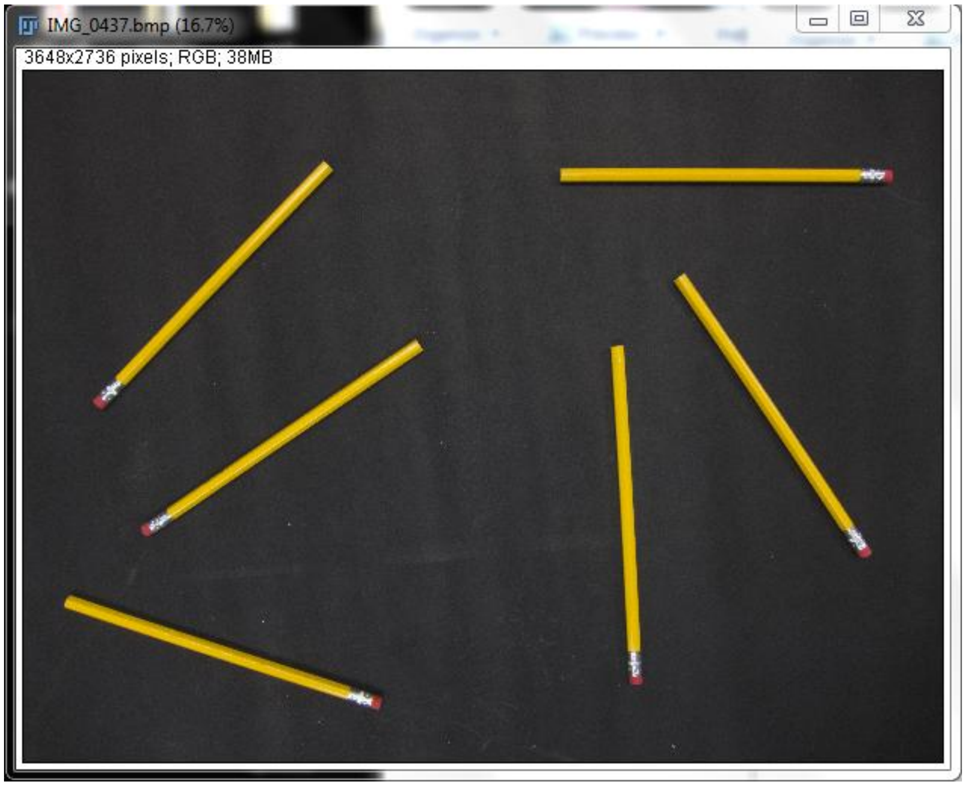


Figure 4.2 Input Image

Step 2: The input image from step1 is sent to Pencil Detection as an ImageProcessor object. In Pencil Detection yellow pixels are converted to white, and the other pixels are converted to black. The image is shown in figure 4.3. The resulting image is converted to gray scale image.

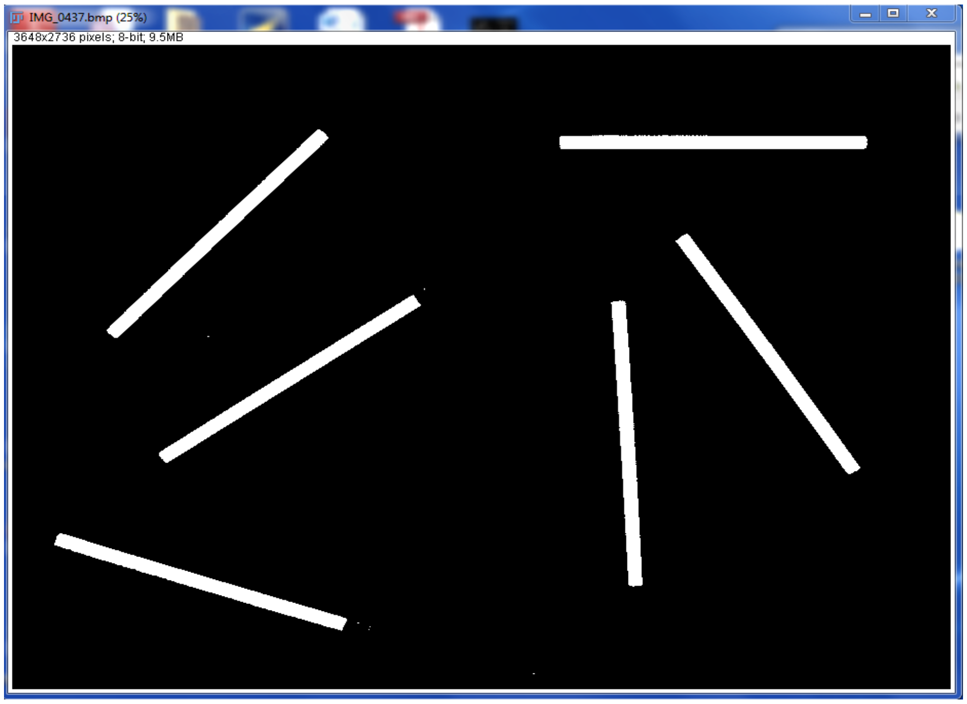


Figure 4.3 Pencil Detection

Step 3: Connected Regions is invoked which internally invokes Find Connected Regions plug-in. Find Connected regions takes the gray scale image obtained from step2 and produces the following image shown in figure 4.4. This image is passed to Length Extraction as an ImageProcessor object.

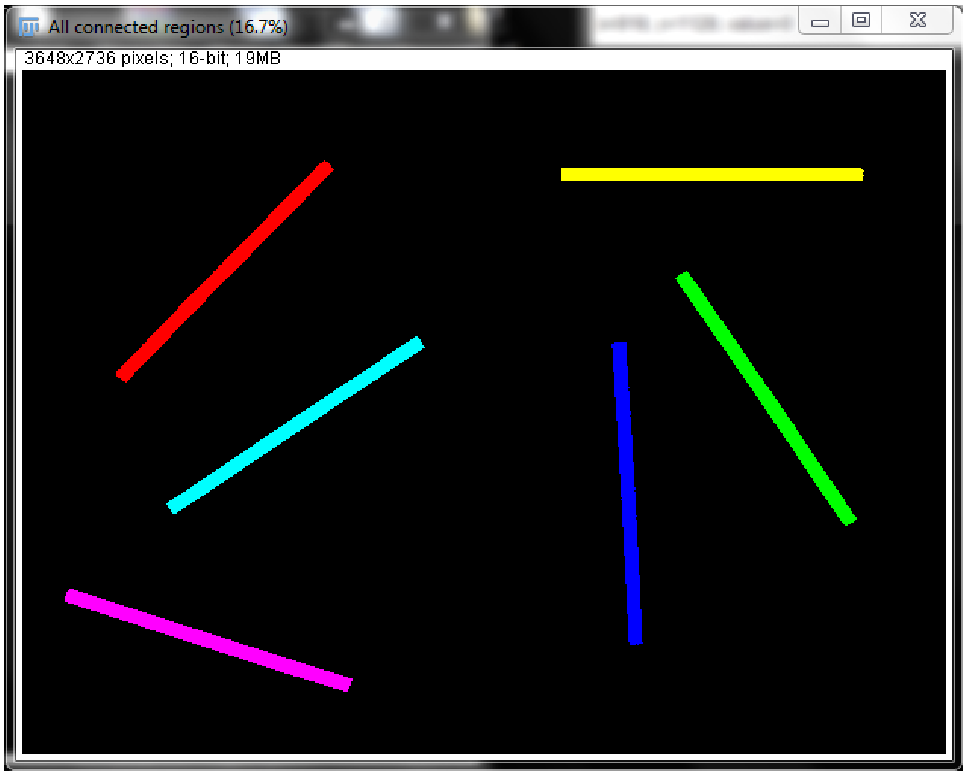
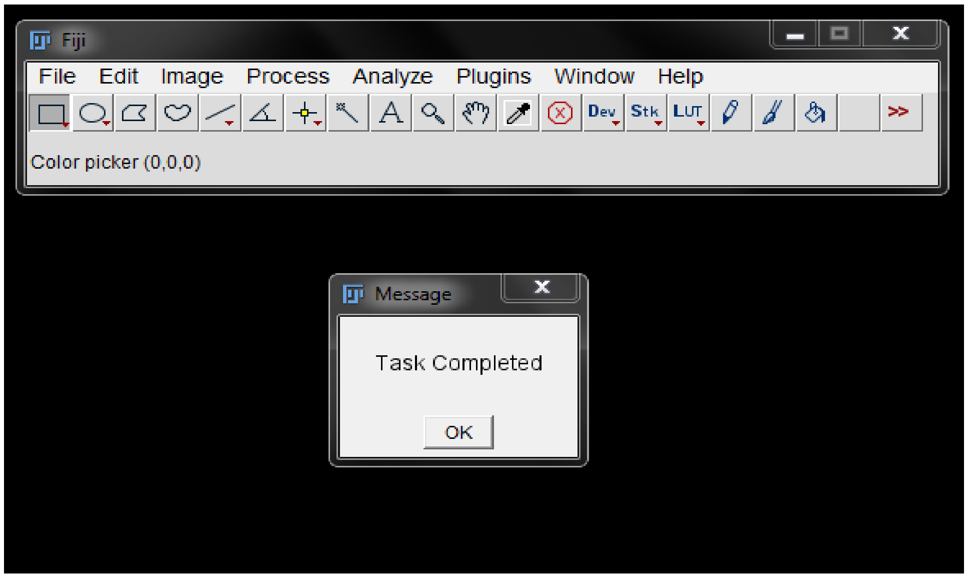


Figure 4.4 Connected Regions

Step 4: Length Extraction takes the image obtained from step 3 as an input argument and from the image the midpoints of the ends of each valid pencil, distance between the two midpoints are written to the output file. All the image windows shown during the process of length extraction are closed automatically and the next image is taken for execution which follows the process from step1 to step4. Finally "Task Completed" message is displayed on the screen.



Color picker (0,0,0)

Message

Task Completed

Figure 4.5 Task Completed

4.2 Analysis

Output:

The output file for the above image is shown below:

```
EPD,41371.83289247685
IMG_0437.bmp,ECD,41336.56490236111,EMD,41169.75460951389,6
1291,2466,168,2101,1180
1209,369,378,1229,1195
1572,1081,577,1757,1202
3325,411,2130,410,1195
2425,2301,2358,1087,1215
3279,1812,2603,811,1207
```

EPD - Excel Processed Date, date and time on which plug-in process the image. EPD is in format recognized by Excel.

ECD - Excel Creation Date, date and time on which plug-in process the image. ECD is in format recognized by Excel.

EMD - Excel Modification Date, date and time on which plug-in process the image. EMD is in format recognized by Excel.

Second line in the output file shows name of the image, Excel Creation Date, Excel Modification Date, Number of valid (complete) pencils in the image.

In the remaining lines first two values are the midpoint coordinates of one end of pencil, third and fourth coordinates are the midpoint coordinates of other end of pencil, fifth value is the distance between the two midpoints.

The white dot shown in the images below indicate the coordinates of midpoint of each end of the pencil that are written to the output file. The results checked manually are in exact match with the data written to the output file.

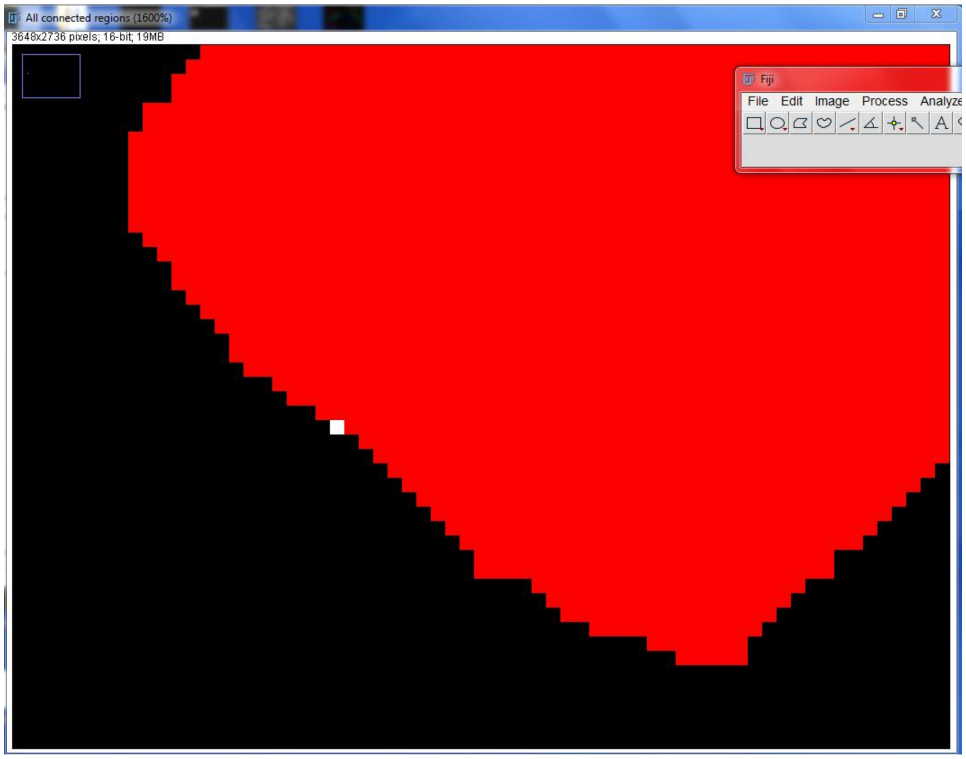

All connected regions (1600%)
3648x2736 pixels; 16-bit; 19MB
Fiji
File Edit Image Process Analyze

Figure 4.6 Pencil 1.a

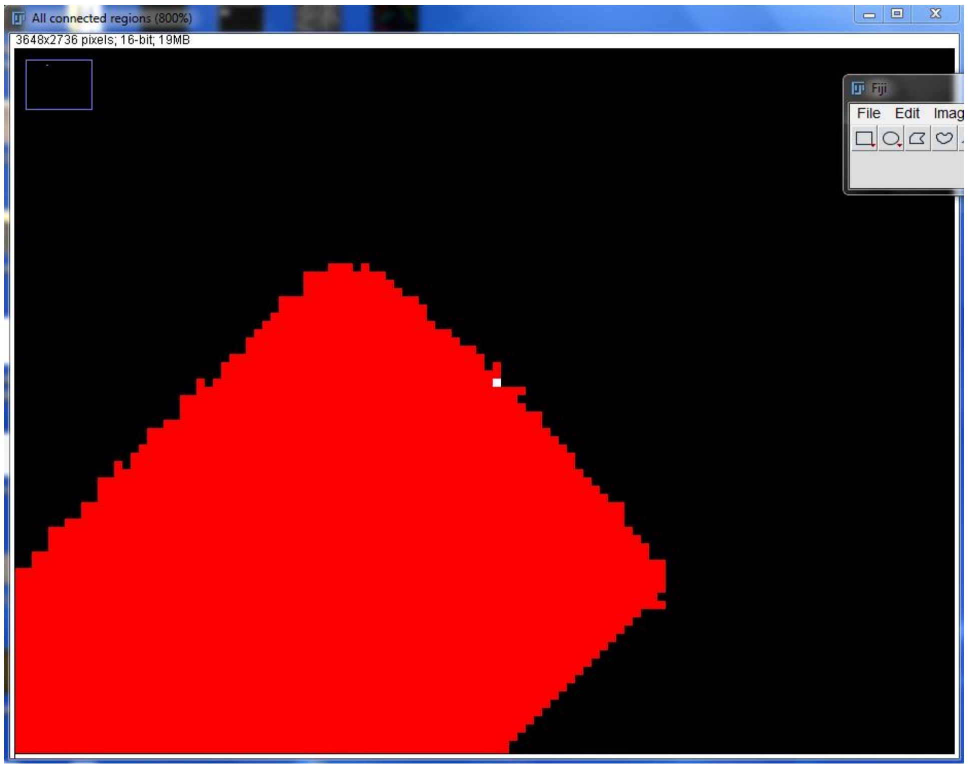


Figure 4.7 Pencil 1.b

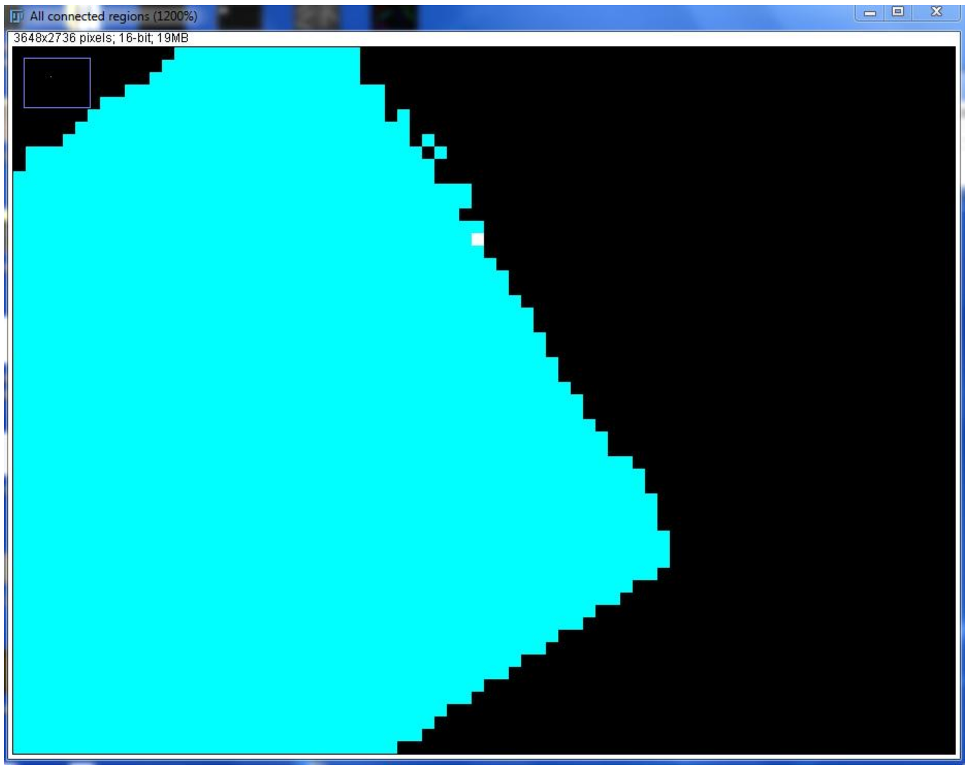


Figure 4.8 Pencil 2.a

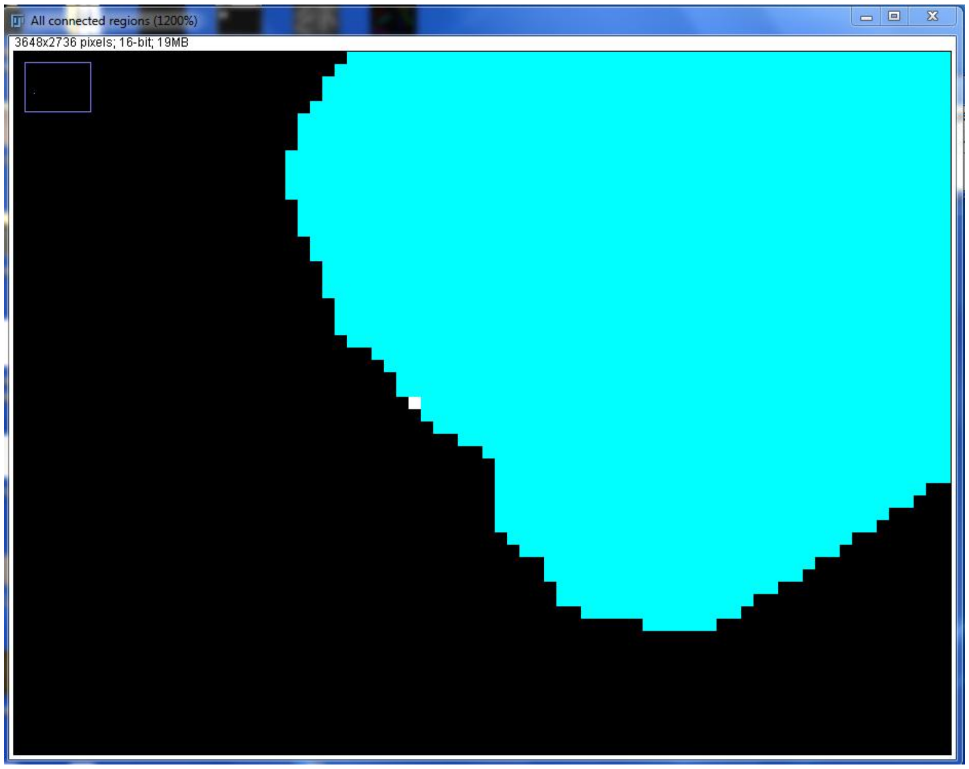


Figure 4.9 Pencil 2.b

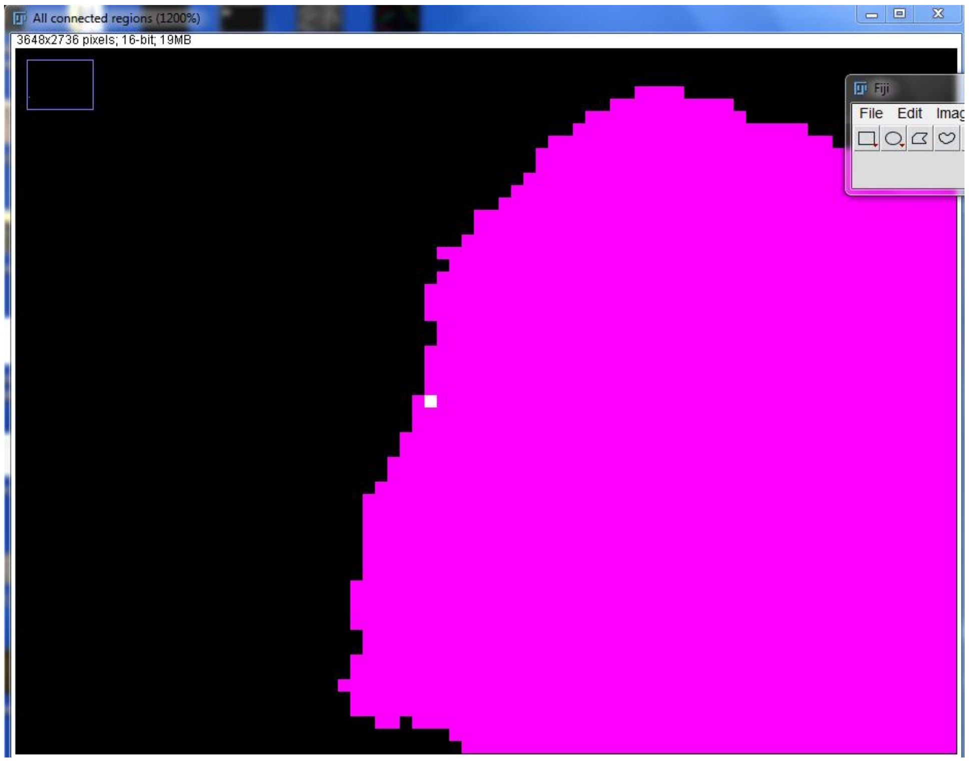


Figure 4.10 Pencil 3.a

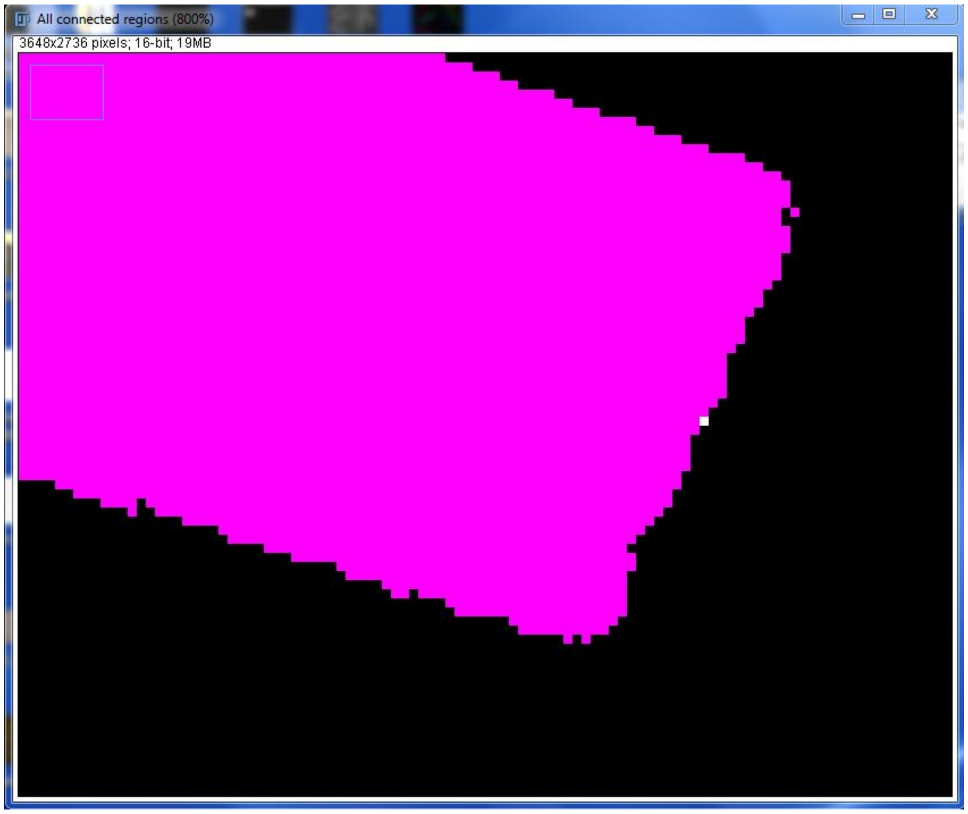


Figure 4.11 Pencil 3.b

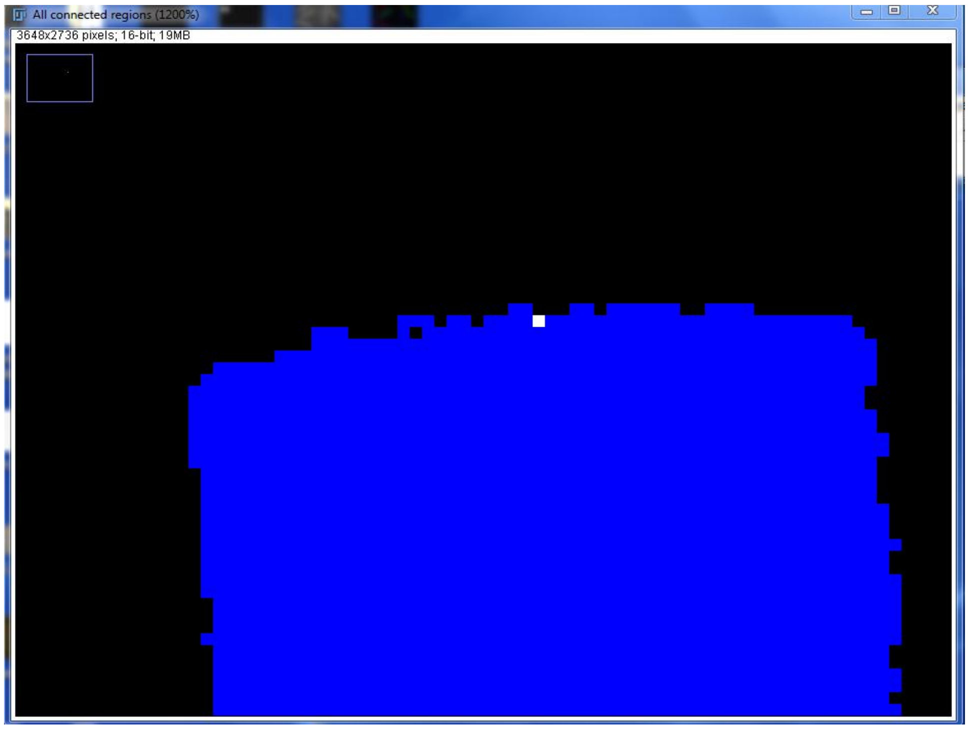


Figure 4.12 Pencil 4.a

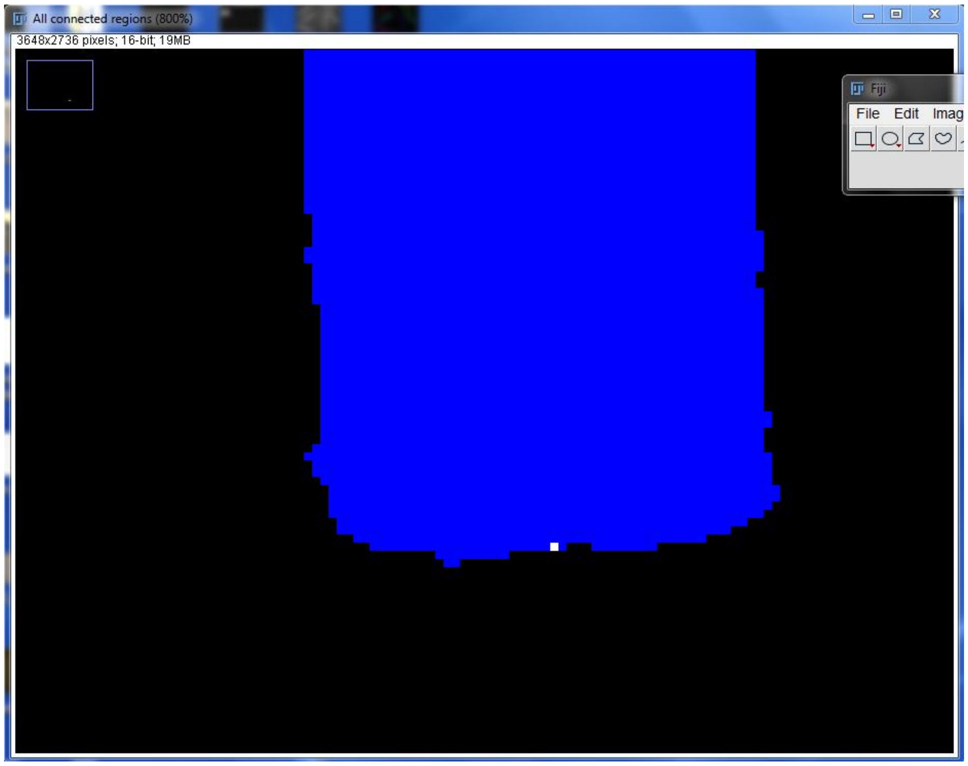


Figure 4.13 Pencil 4.b

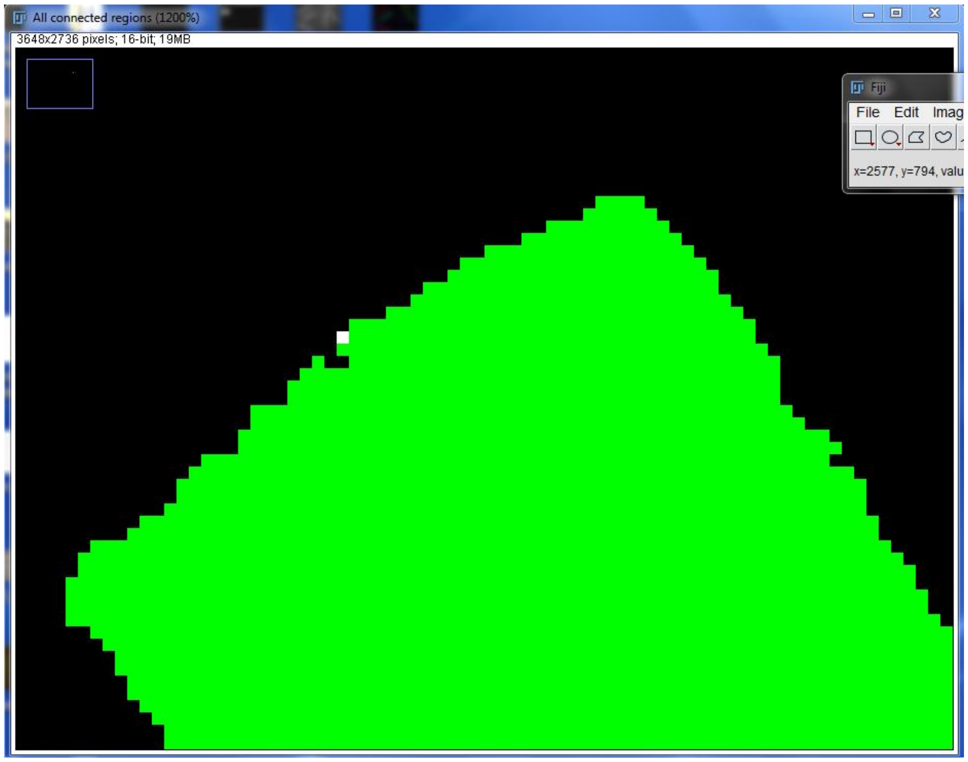


Figure 4.14 Pencil 5.a

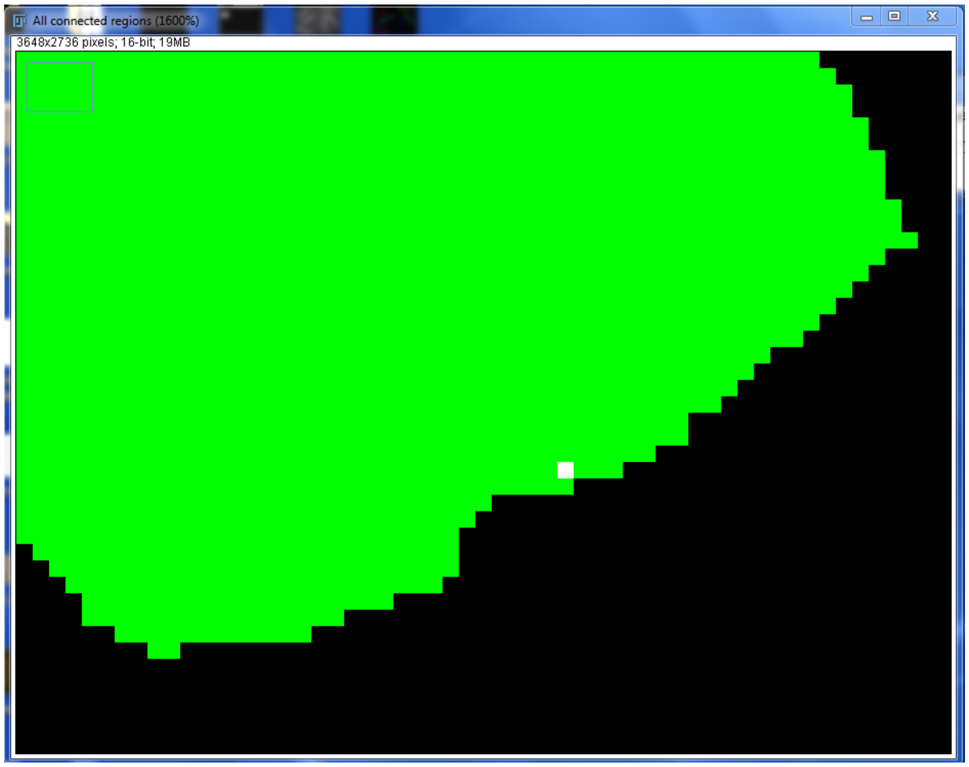


Figure 4.15 Pencil 5.b

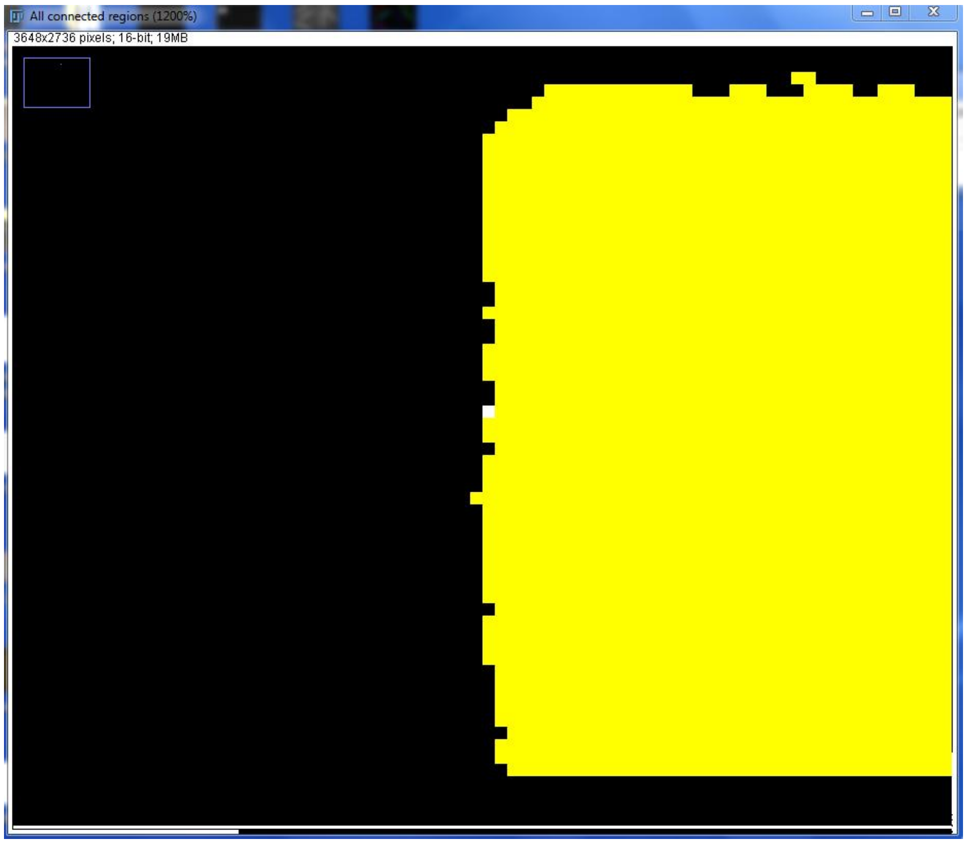


Figure 4.16 Pencil 6.a

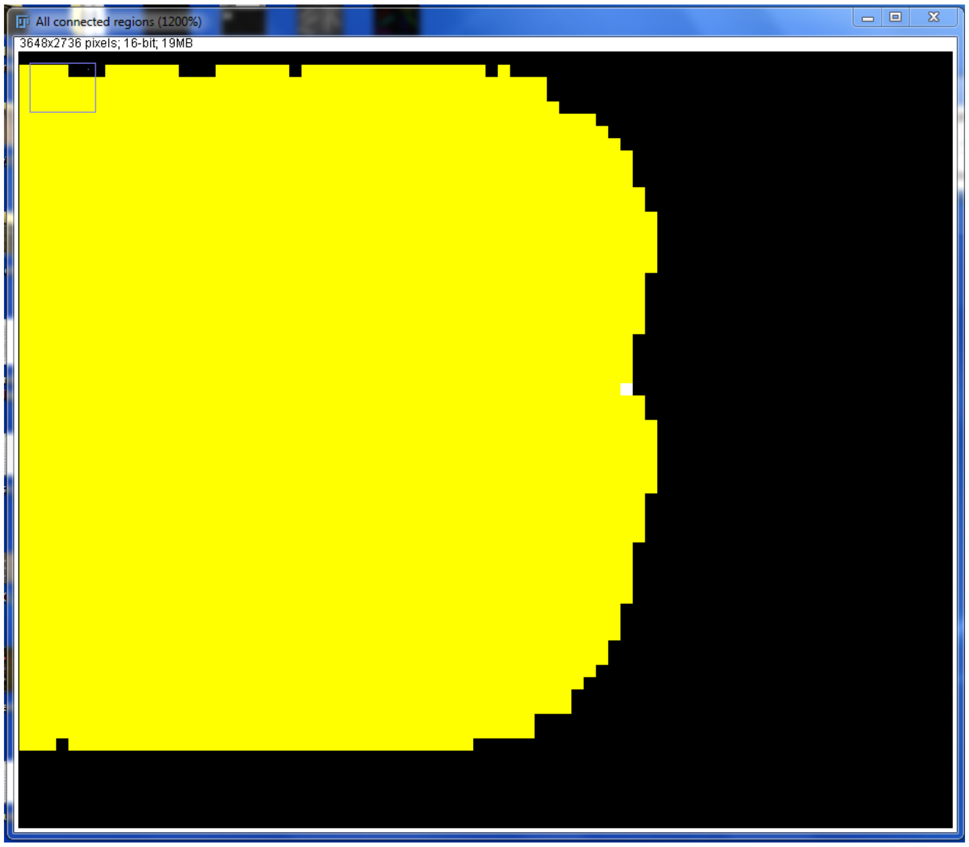


Figure 4.17 Pencil 6.b

Chapter 5 - Conclusion and Future Work

Working for the first time on Image Processing, it took a lot of hard work and time to understand the concepts and execution. The plug-in extracts the length information from the plant images and pencil images it processed. The length information is used by the researchers to calculate the scale factor. The 3D models we work with are generated for plant images. With the help of scale factor plant features such as length of leaves, length of stem, plant inflorescences, etc can be analyzed periodically. The plug-in has been implemented and tested on real images. Working on this project helped me to enhance my skill in image processing domain.

5.1 Challenges

- Considering only full pencils
- Considering each pencil into unique region as all pencils are unique

5.2 Future Work

- The plug-in can be enhanced to process the images in the folder concurrently, reducing the overall processing time.

Chapter 6 - References

[1] Fiji

http://fiji.sc/Fiji

[2] ImageJ

http://rsbweb.nih.gov/ij/

[3] Digital Image Processing: An Algorithmic Introduction using Java

http://books.google.com/books/about/Digital_Image_Processing.html?id=4gBUz_IkkSsC

[4] Find Connected Regions

http://www.longair.net/edinburgh/imagej/find-connected-regions/

[5] Photogrammetry

http://www.agisoft.ru/wiki/Photogrammetry#What_is_photogrammetry.3F

[6] Agisoft

http://www.agisoft.ru/wiki/Main_Page

[7] Inflorescences

http://en.wikipedia.org/wiki/Inflorescence

[8] JFileChooser

http://docs.oracle.com/javase/6/docs/api/javax/swing/JFileChooser.html

[9] JavaXT Core

http://www.javaxt.com/documentation/?jar=javaxt-core.jar

[10] Digital Image Processing (using Java)
http://www2.nkfust.edu.tw/~wenh/2008/multimedia/Java/Digital_Image_Processing_Java3.pdf

[11] Color Models
http://en.wikibooks.org/wiki/Color_Models:_RGB,_HSV,_HSL