PLOT2TABLE

TEAM 11

Contents

Part I

Software Requirement and Specifications

Chapter 1

Introduction

1.1 Purpose

This SRS describes the software functional and non-functional requirements for release of plot2table v0.1. This software is a standalone application is designed to read a set of scanned pages as input where each page has one or more plots embedded in text and convert each plot to table. Unless otherwise stated, all requirements specified here are of high priority and committed for release v0.1.

1.2 Product Scope

This software consists of following major functions:

- Import scanned pages as PDF which may contain one or more plots obtained from experimental results.The plots may or may not be embedded in text. Each plot has the X and Y axes with their labels and unit measurements marked in the plot (linear scale). Inside each figure are one or more plots, each with a different colour depicting a certain plotted entity.
- Generating an output set of pages each page containing a plot with its corresponding table
- Selecting one or multiple pages of the set of images to save as a PDF.
- Selecting one or multiple pages of the set of images to print
- Generate plot from the obtained output data

1.3 Definitions, Acronyms and Abbreviations Used

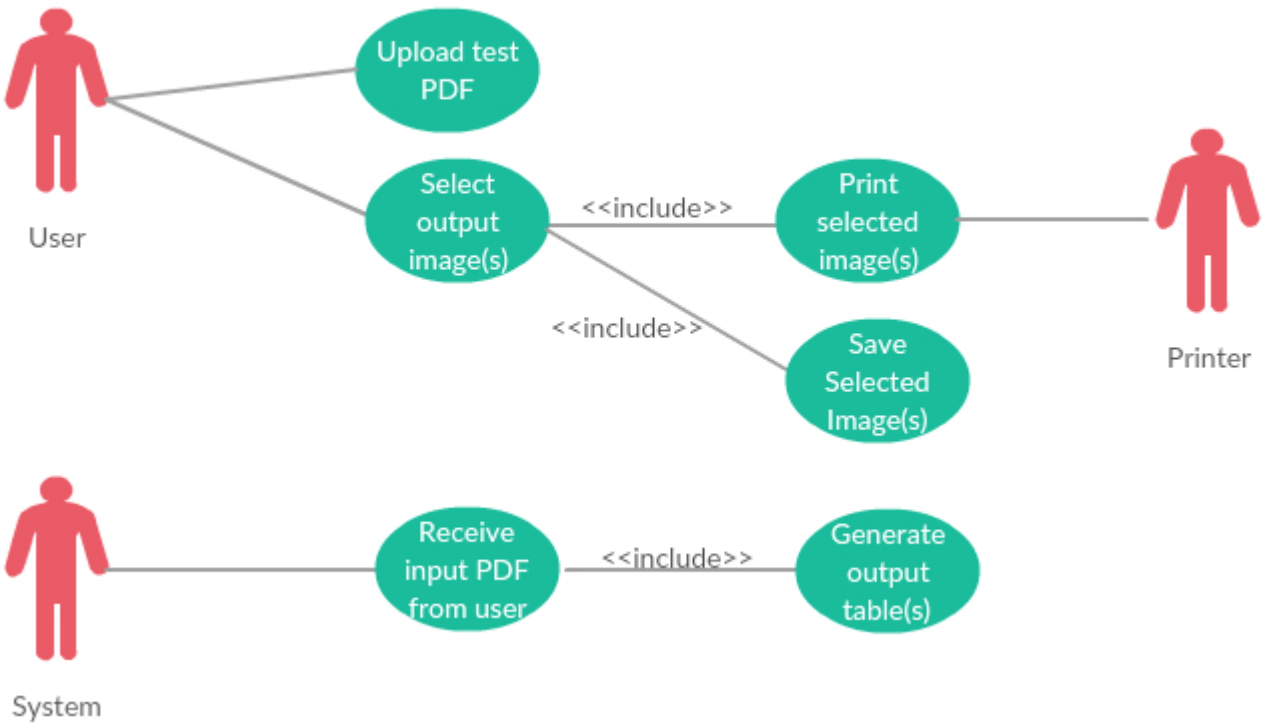
Input PDF: An input PDF is the PDF of the scanned pages which contain the 2-D plots that are to be converted to tables.
Output PDF: An output PDF is the PDF generated from the input PDF. It contains the tables corresponding to the 2-D plots in the input PDF.
GUI: Graphical User Interface

Chapter 2

Overall Description

2.1 Product Perspective

This software intends to take random scanned PDF containing one or more plots which may or may not be embedded in text.Each plot has the x and y axis with their labels and unit measurements marked in the plot . Inside each figure are one or more plots, each with a different colour depicting a certain plotted entity and their labels given separately within the plot as a caption. We generate tables corresponding to all the 2-D plots. The output PDF consisting of selected plots and tables can be saved in the desired destination for future reference.We may also print the plots and tables.Also there is an option to generate plots from the data tables obtained



2.2 Product Functions

The set of functionalities supported by this software are as follows :

- *Upload PDF*: The software allows the user to import a scanned input PDF and process it to extract plots and generate tables containing plot data.

- *Browse Plots*: Browse through the images extracted

- *Table to plot*: Select one or more input plots to visualize the extracted tables.

- *Save Output to PDF*: Select one or more plots to save as a PDF at a desired destination

- *Print Output Plots*: Select one or more plots to print, along with the table data.

2.3 User Classes and Characteristics

- __init__ : Starting point

- ui : Calls preprocessor to generate images and then calls generateTables to obtain tables of the generated images

- generateTables : Generates tables from plots by calling legend_detect,calibrator,axisD

- preprocessor : Generate images from imported PDF and enhance them

- calibrator : gets labels for axes

- garbageImageDetector : checks if axes is present in the cropped images
- legend_detect : detects legends of plots
- titleExtractor : obtains title of plots
- color_kmeans : detects entity colours
- utils : contains utility functions
- axisDetector : detects axes of plots

2.4 Operating Environment

This software is developed in Python 2.7, running on Debian-based Ubuntu 14.04 x64 Architecture. It should also be compatible with other Debian-Linux based x64 Operating Systems. This software is not compatible with any version of Python 3.

2.5 Design and Implementation Constraints

- The plots must have both X and Y axes
- If there are multiple entities in the same axes ,each plot must be with a different colour and none of the entity colours is black.
- The plot must be distinguishable from the the background
- Text and Tables in documents should be written in black color.

2.6 User Implementation

A user manual in PDF format would be made available along with the software.

2.7 Assumptions and Dependencies

This software has been targeted at Debian-based x64 Operating Systems. It depends on Open Source tools and Python libraries like:

- Tesseract OCR
- NumPy
- multicrop
- MatPlotlib
- Scikit-learn
- Report Lab

The required dependencies are all packaged with the installer, although super-user permissions might be necessary to install them. The software cannot be installed on any other operating system apart from Linux-Debian based x64 Architecture.

Chapter 3

External Interface Requirements

3.1 User Interfaces

The user interface is simplistic. It consists of a file menu within which is an option to open a PDF using file chooser and process it to generate the tables from plots. Then the plots are displayed in the image pane of the screen and can be selected to display the plot along with its table on the document pane.One or more pages may be selected by appropriately checking check boxes to save the selected plots with the tables as PDF or print them.There is also an option to generate plots from the output data,which when clicked opens a new window for each plot.

3.2 Hardware Interfaces

No special hardware is needed for the functioning of this software. A computer with a monitor, a keyboard and a mouse suffices. A printer must be connected to the computer to print the tables.

3.3 Software Interfaces

The software consists of a single user multitasking system which can open a PDF and generate tables from plots. For extracting plots command line tools like pdf2ppm,multicrop,imagemagick have been used. The images are enhanced using Python OpenCV. Tesseract OCR has been used for OCR functionalities.

3.4 Communication Interfaces

Internet connection is necessary for the installation of the software, although it is not needed for its working.

3.5 Memory Constraints

The software is quite memory efficient as it only requires minimal space to store the uploaded PDF in the project directory which can later be freed by the wish of the user. Any temporary files created by the software are erased upon exit.

Chapter 4

Functional Requirements

4.1 Open PDF

Select OPEN option from the file menu and choose the PDF using the file chooser.A PDF will be imported an it will be processed to detect all plots and generate tables for the plots.

4.2 Browse output pages

Selecting a plot on the image pane will display the plot and its table in the Document pane.

4.3 Plot from tables

Plot the extracted tables to compare results with the input plot.

4.4 Save

On selecting one or more plots from the image pane and clicking SAVE button the selected plots and their tables are saved as PDF. The user may modify the location where it is to be saved and rename it.

Chapter 5

Other Non-functional requirements

5.1 Performance Requirements

After uploading the required PDF and selecting the pages the software takes less than 5 seconds per PDF to process it. Thus the software is quite performance efficient in this regard.

5.2 Security and Privacy Requirements

The software does not require submission of sensitive information and hence no security and privacy requirements have been identified.

5.3 Reliability

As this software does not depend on any external factors for its processing it proves to be quite a reliable software.

Part II

Installation Instructions

Chapter 6

Installing dependencies

1. Open terminal. Press `Ctrl + Alt + T` on ubuntu to open it.
2. Execute `install.sh` file by typing `bash install.sh` and hitting enter.
3. Type `Y` and press enter whenever prompted during installation.

The dependencies are large and hence the complete installation can take time depending upon pre-installed packages. To avoid waiting for long, some of the large dependencies are packaged with the software itself. The sources are based on Ubuntu 14.04 LTS system with architecture amd64. The source codes are stored in `debs/` directory. Execute `install_partial_offline.py` script by typing `python install_partial_offline.py` and hitting enter. This installs all the dependencies using the packaged dependencies and installing some others from the online repository.

The following packages are supposed to be installed :

6.1 Aptitude dependencies

- build-essential
- python-dev
- python-pip
- imagemagick
- libopencv-dev
- python-opencv
- tesseract-ocr

6.2 Pip dependencies

- setuptools
- numpy
- scipy
- reportlab
- scikit-learn
- matplotlib
- PyQt4

Part III

User Manual

Chapter 7

Starting the software

7.1 Graphical User Interface

To start the GUI of the software, open terminal and type `plot2table` and press enter. This will open up the software. Further instructions are given down below.

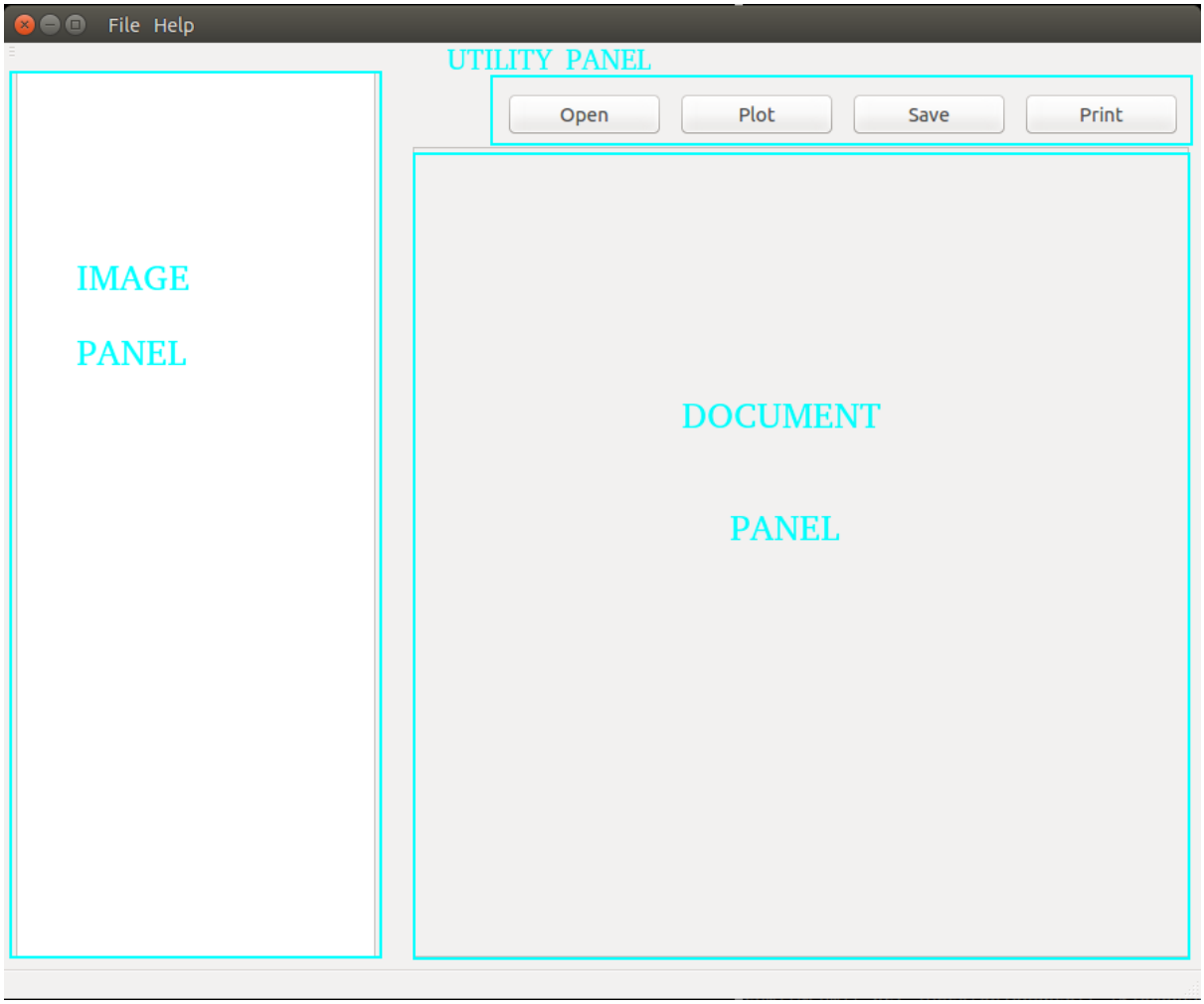
7.2 Command Line tool

This software can also be used without opening the GUI, using its Command Line Interface. Run `plot2table name_of_file.pdf` where the second argument is the name of the input pdf file. This will process the pdf and generate the output pdf on the same location.

Chapter 8

Main Window

Main Window appears first whenever you open the software. The following are the most important areas of the main window



8.1 Image Pane

After the user imports an input PDF and the processing is complete, the plots appear as individual images in the image pane. Each image has a corresponding check box which is useful to print and save the output PDF.

8.2 Document Pane

When user clicks on a particular picture box in the image pane, the enlarged version of the selected image along with the table appears on the document pane.

8.3 Utility Panel

This section has different buttons and menus to be used by the user. Description of buttons is given below:

8.3.1 Import

1. User can upload a PDF file by clicking on "OPEN" Button. A File Chooser Dialog appears after clicking on OPEN Button so that user can select a PDF file.

2. When user imports a PDF file another window appears which has a button "Process". When user clicks on this button processing starts. User can see the status in a progress bar in this window. After processing is done, user again reaches the Main Window having all the plots from the PDF file on the image pane and the first image selected by default appears on the document pane.

8.3.2 Plot

On checking the boxes of the images on the image pane an clicking on plot, a new window containing plot from data processed from input PDF appears for each of the selected images.

8.3.3 Save

User can save data for one or more than one number of plots at a time by clicking "SAVE" button. When user clicks the save button the a new window

will appear in which user can select any directory where he/she want to save the PDF file and rename it.

8.3.4 Print

When user clicks on the "PRINT" button,a PDF containing the data for the selected plots gets generated which can be printed.

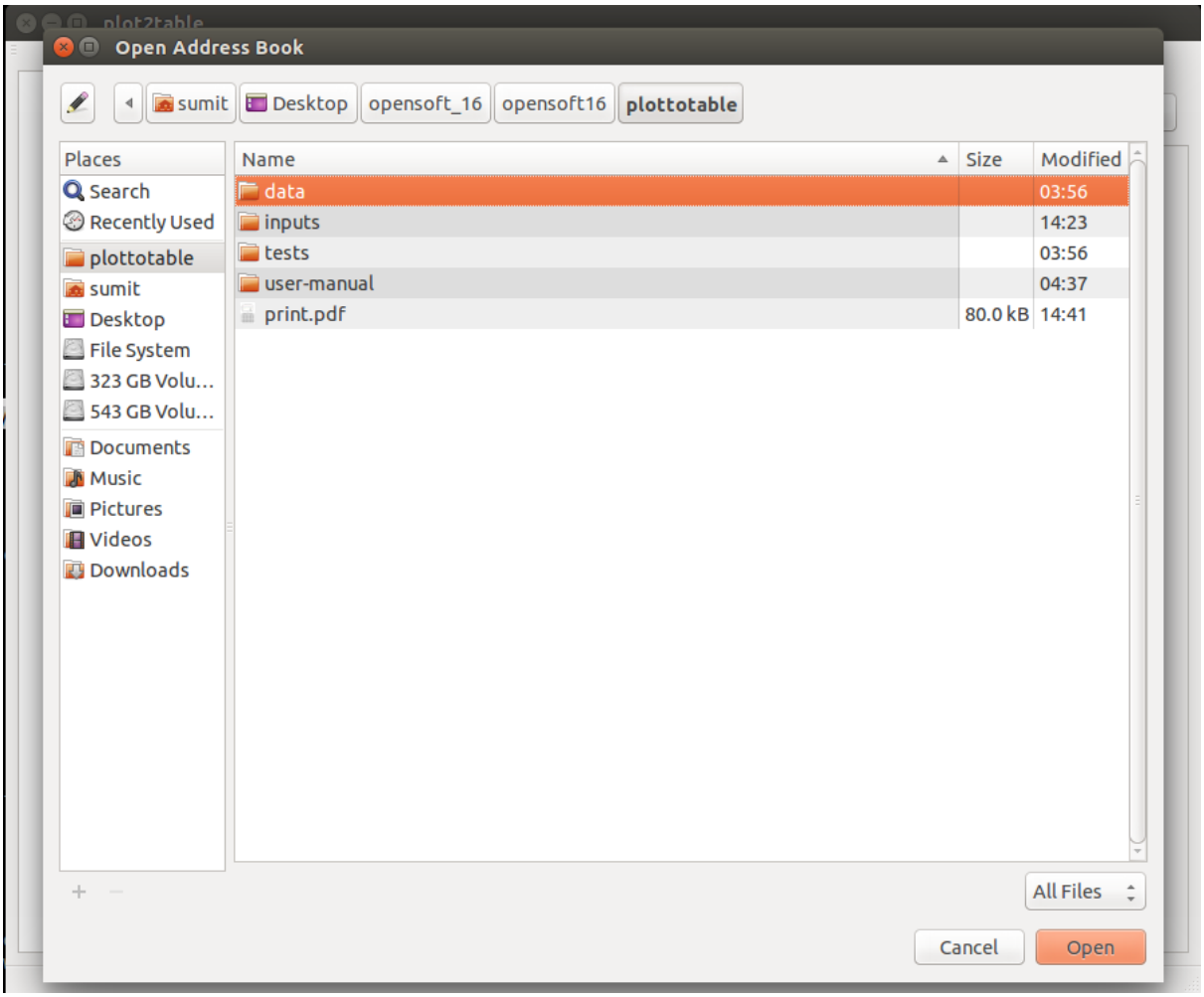
Chapter 9

File Operations

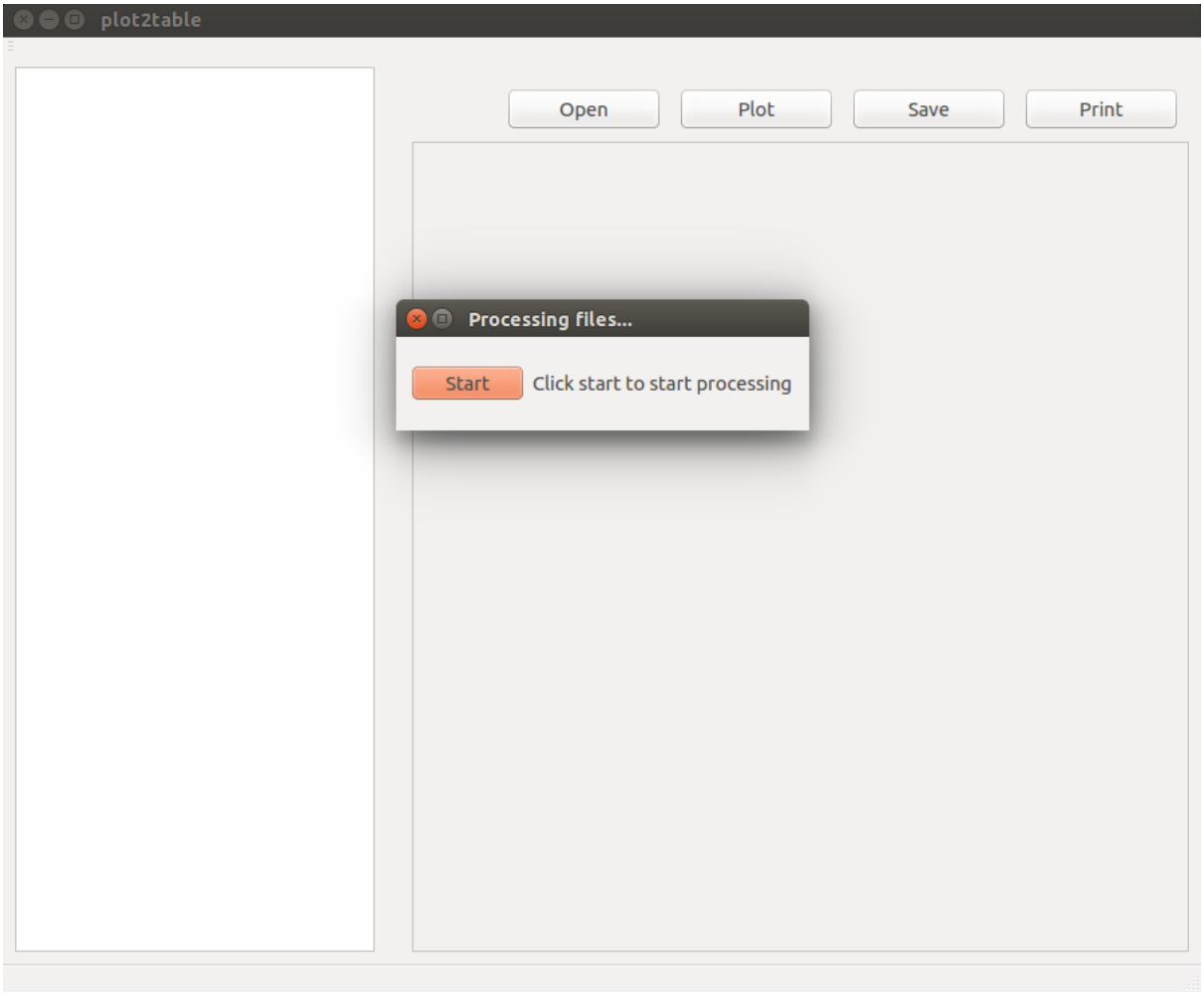
9.1 Import PDF File

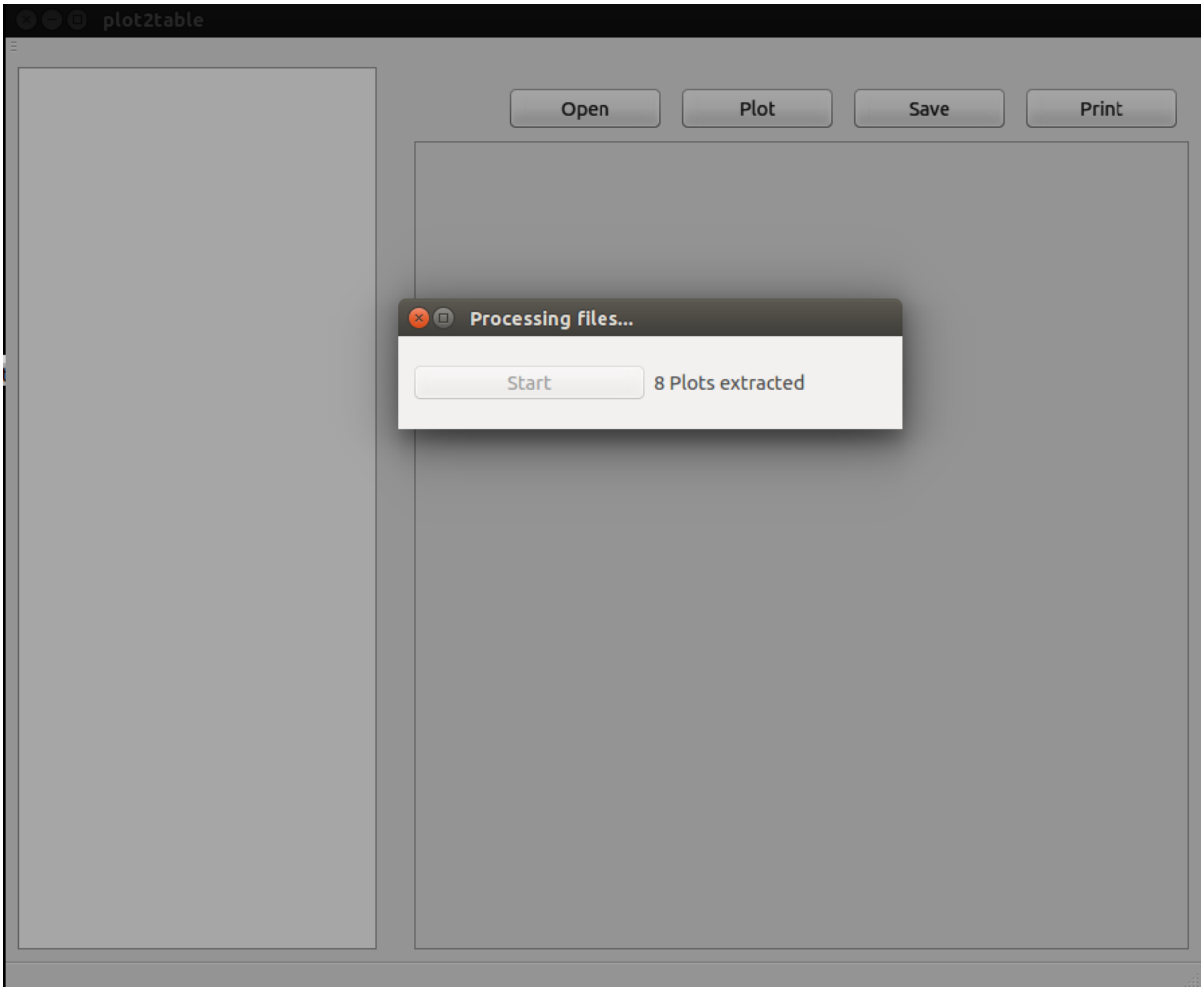
Steps to Import the PDF File are given below:

1. Click on the "Open" button.

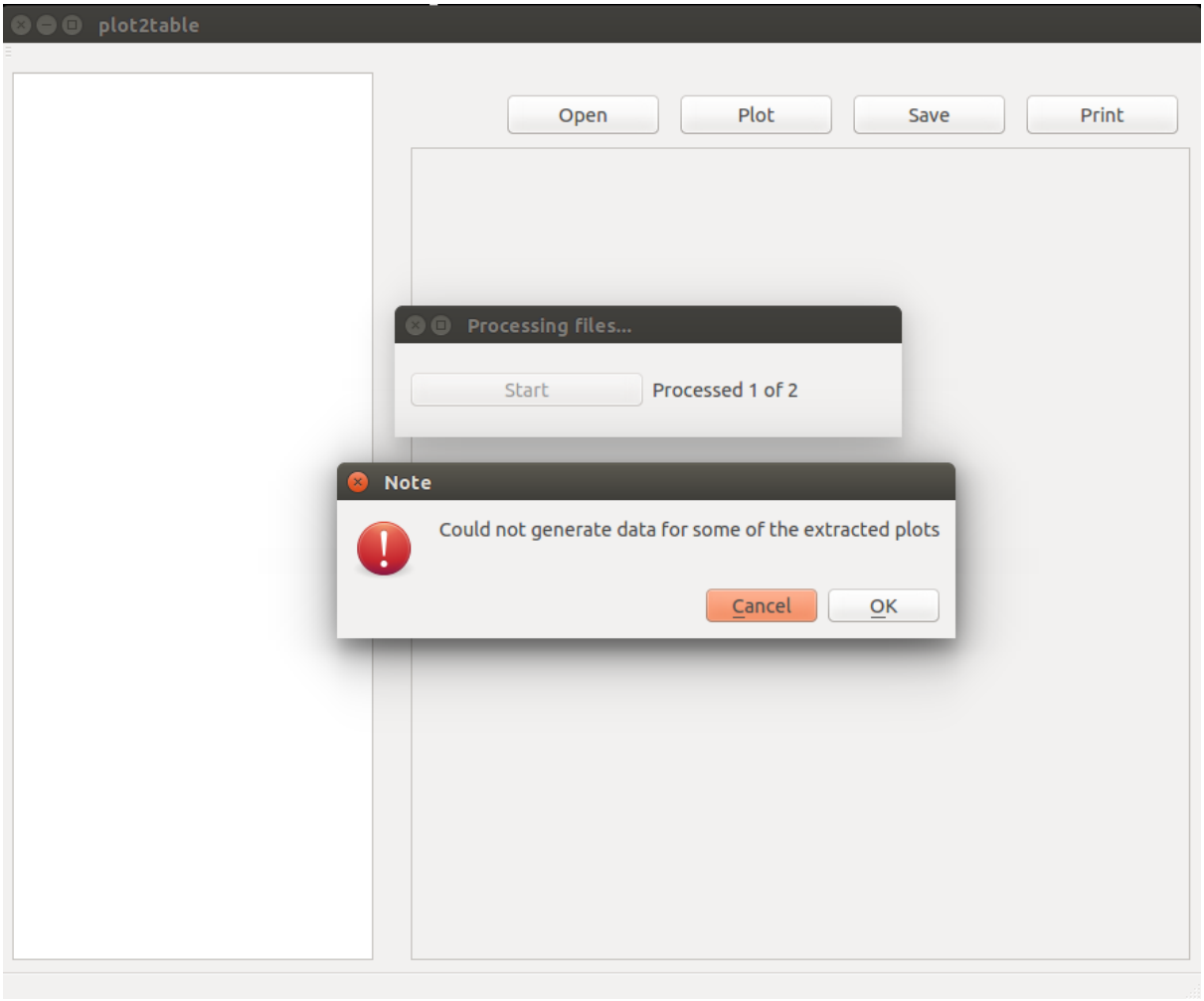


2. Select the file which you want to import and click on "Open" button.
3. Now, click on "Process" button to start processing the PDF file.

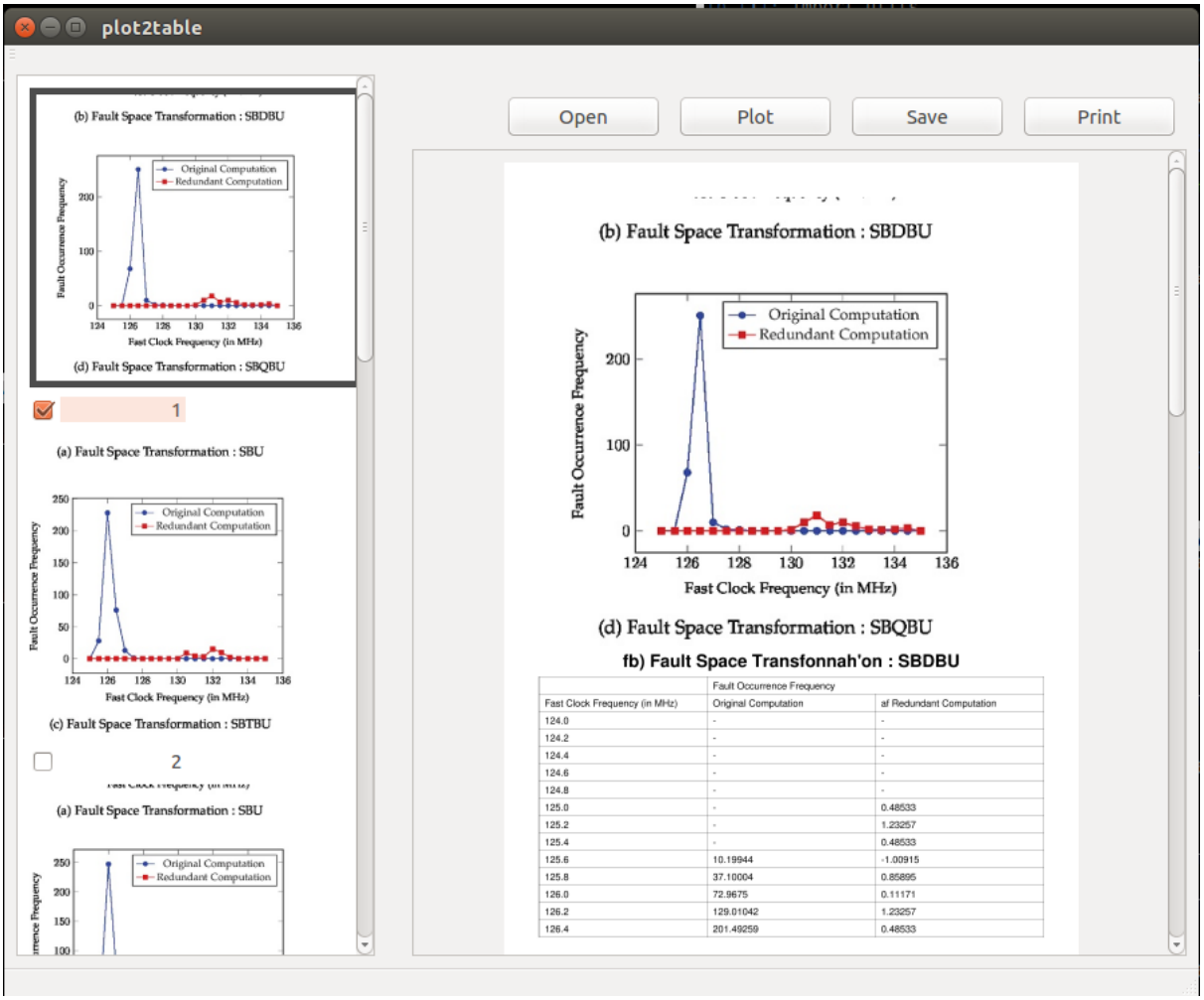




4. If we don not get data for some plot then it will show this warning.



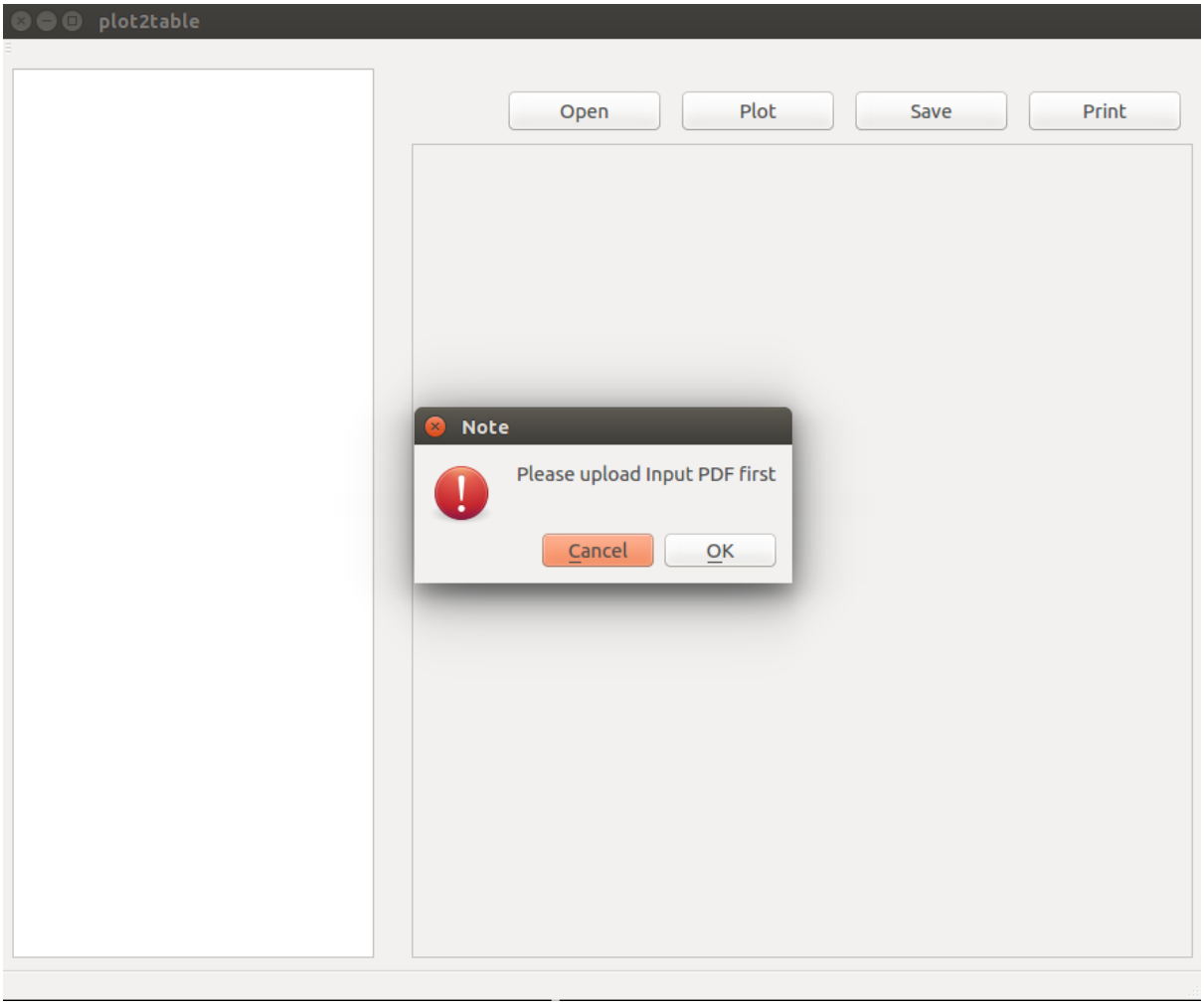
5. After processing is done you can see all the plots in the Image Pane on the left side



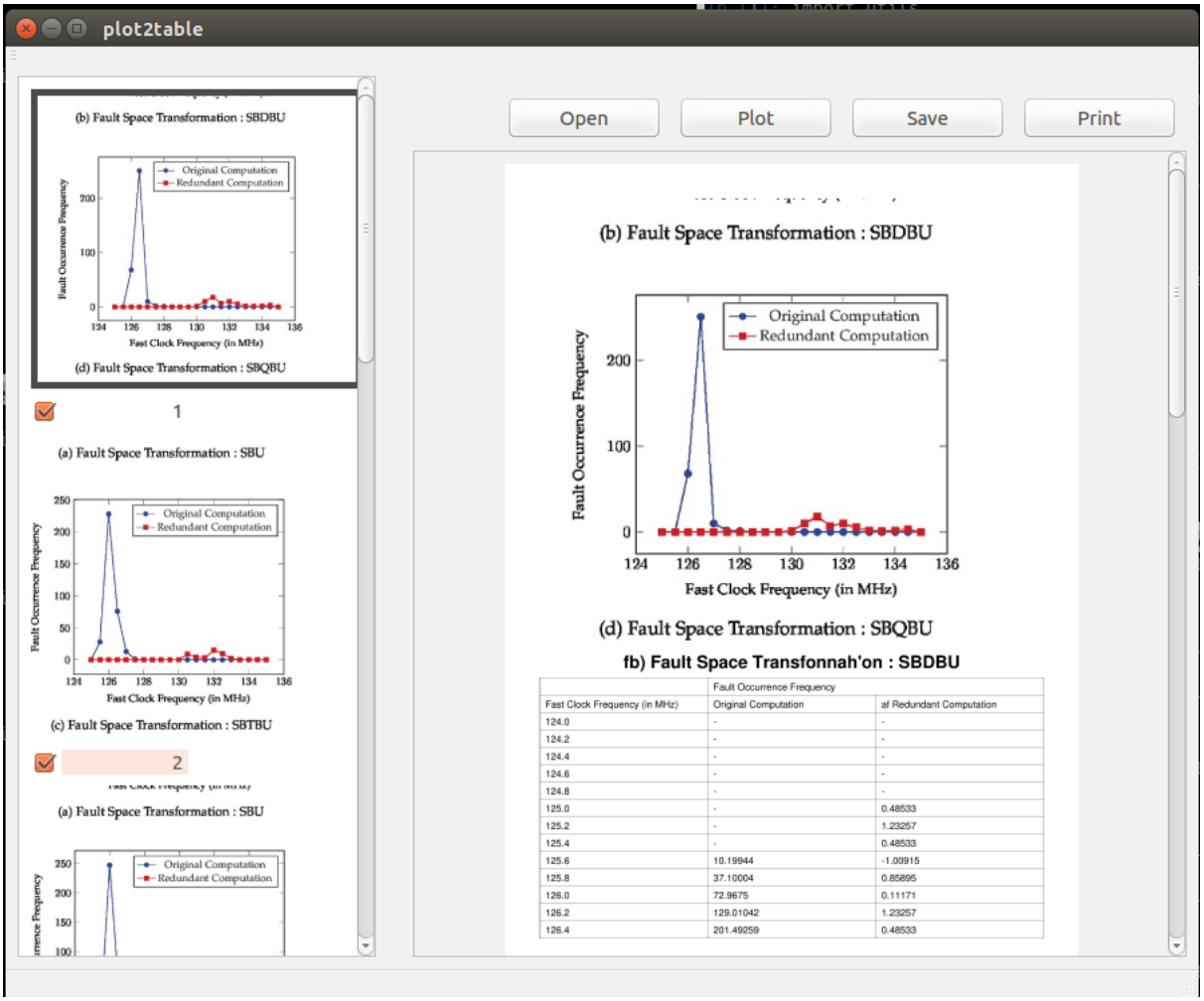
9.2 Plot Data

Steps to plot the extracted data are given below:

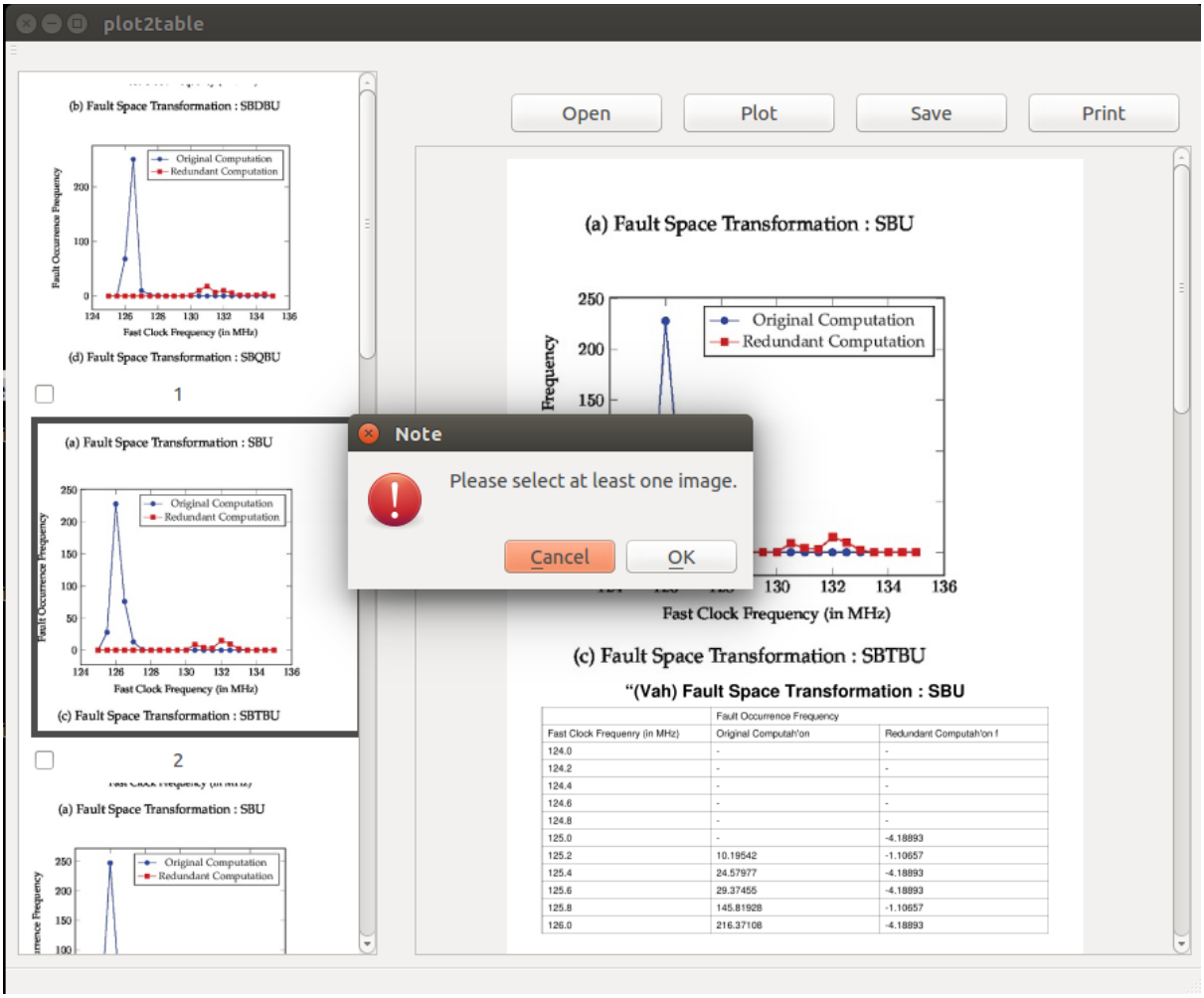
1. If you click on any button except "Open" without any input PDF File then you get this warning.



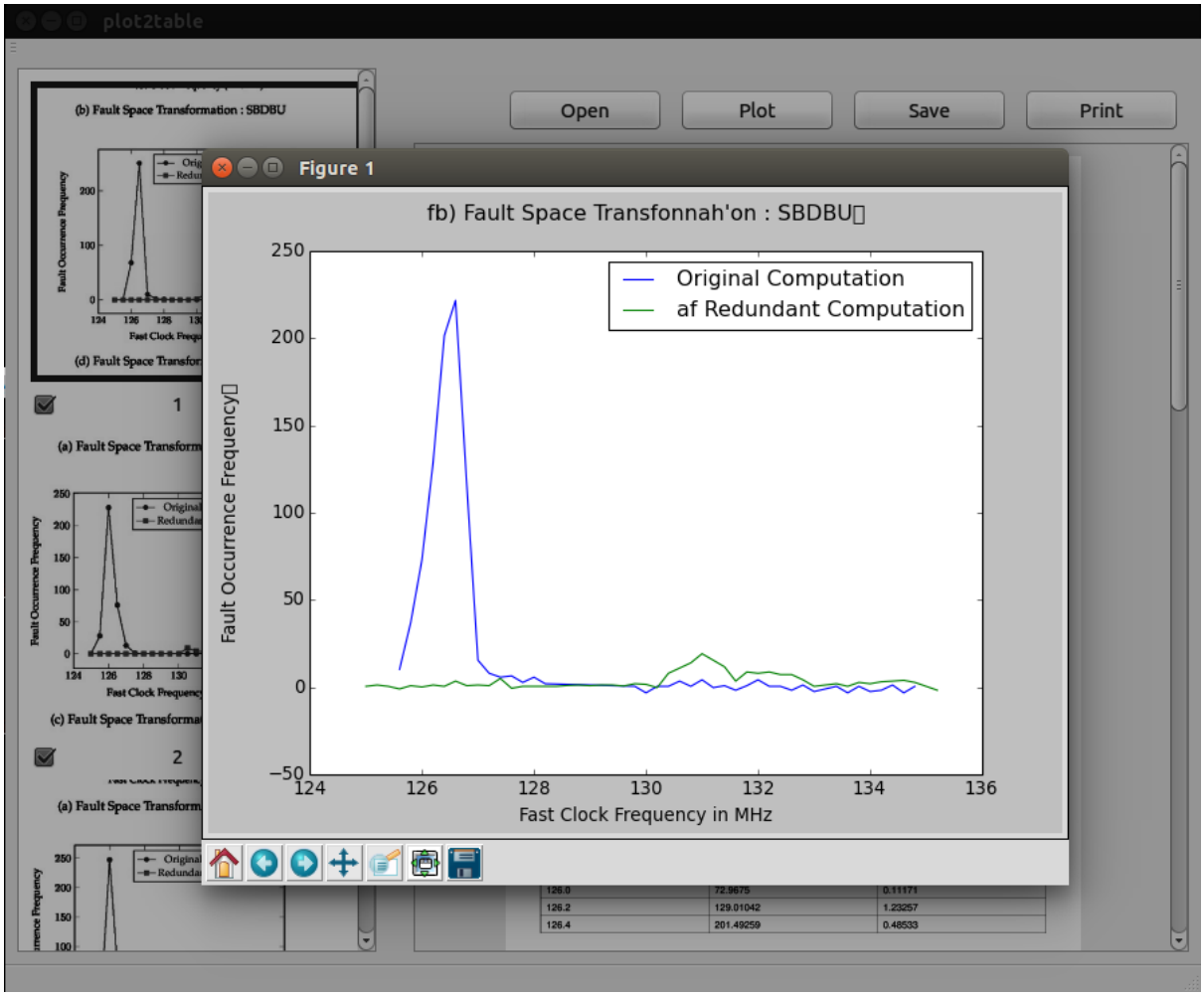
2. Select the image from Image Pane for which you want to see the plot.

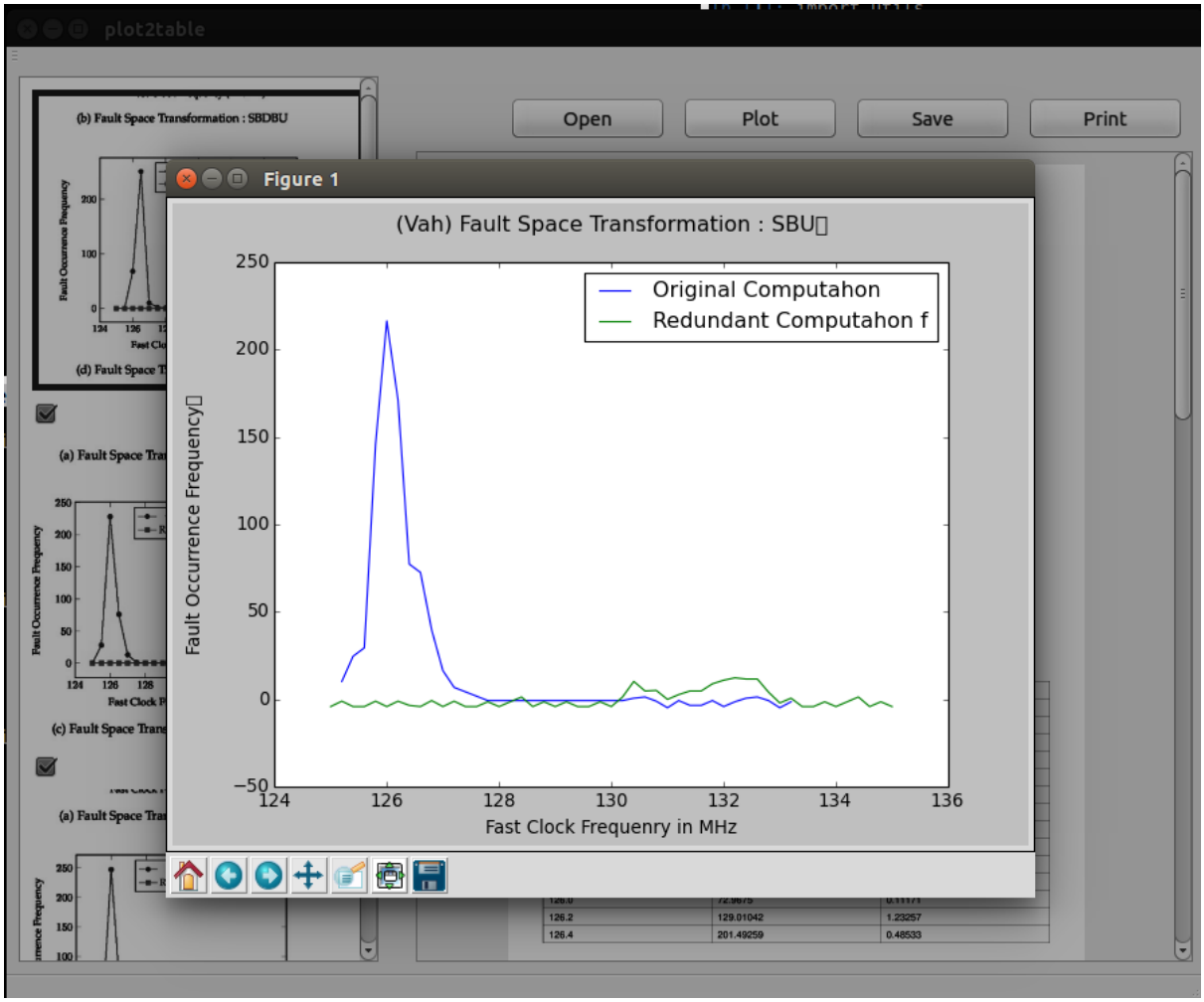


3. Click on the "PLOT" button.
   If you have not selected any plot



4. After clicking Plot button a graph appear which is made by plotting extracted data from selected plot.
   If you have selected multiple images than you have to close the first plot to see the next plot

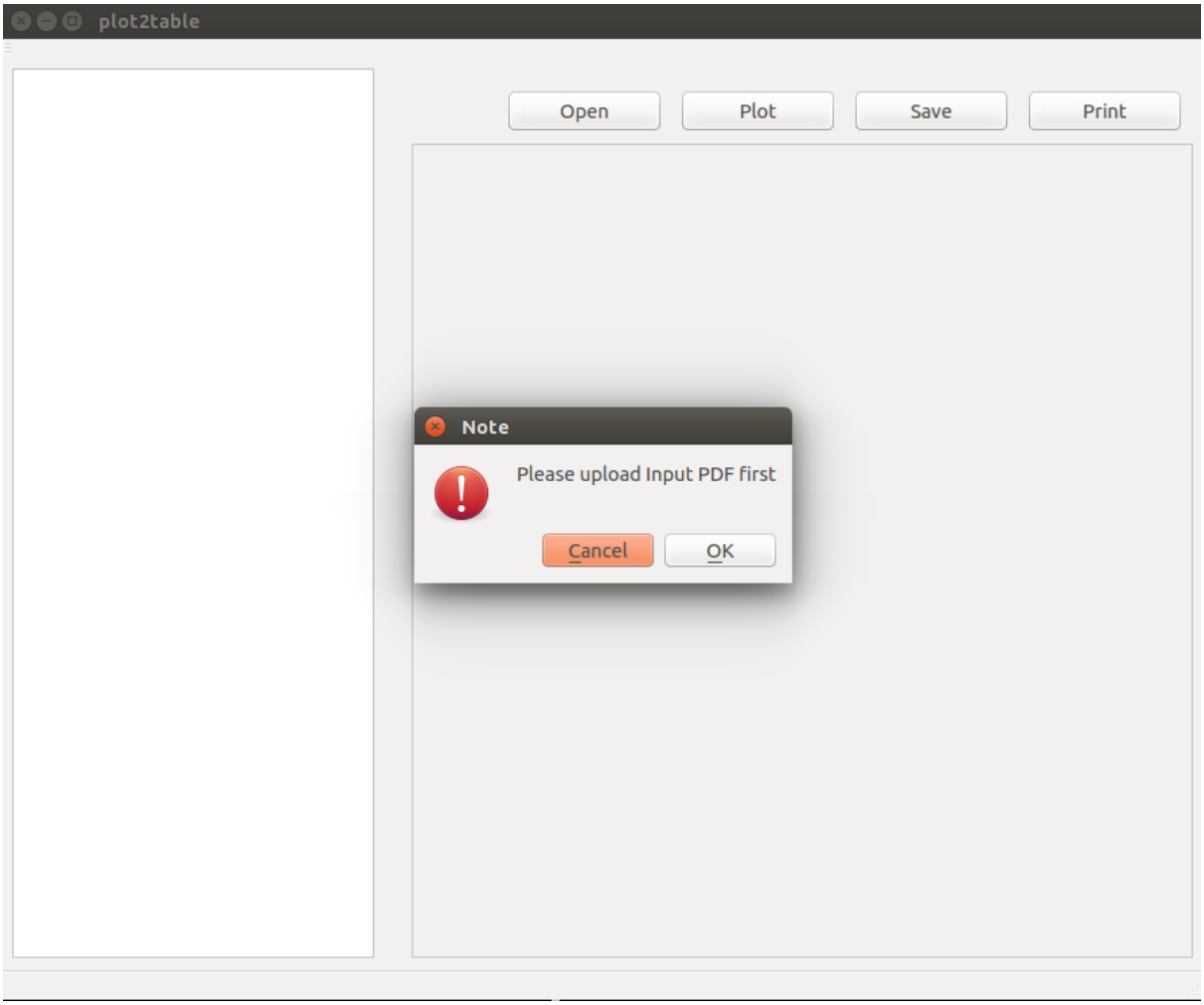

plot2table
Open
Plot
Save
Print
Figure 1
fb) Fault Space Transfonnah'on : SBDBU
Original Computation
af Redundant Computation
Fault Occurrence Frequency
Fast Clock Frequency in MHz



9.3 Save Output File

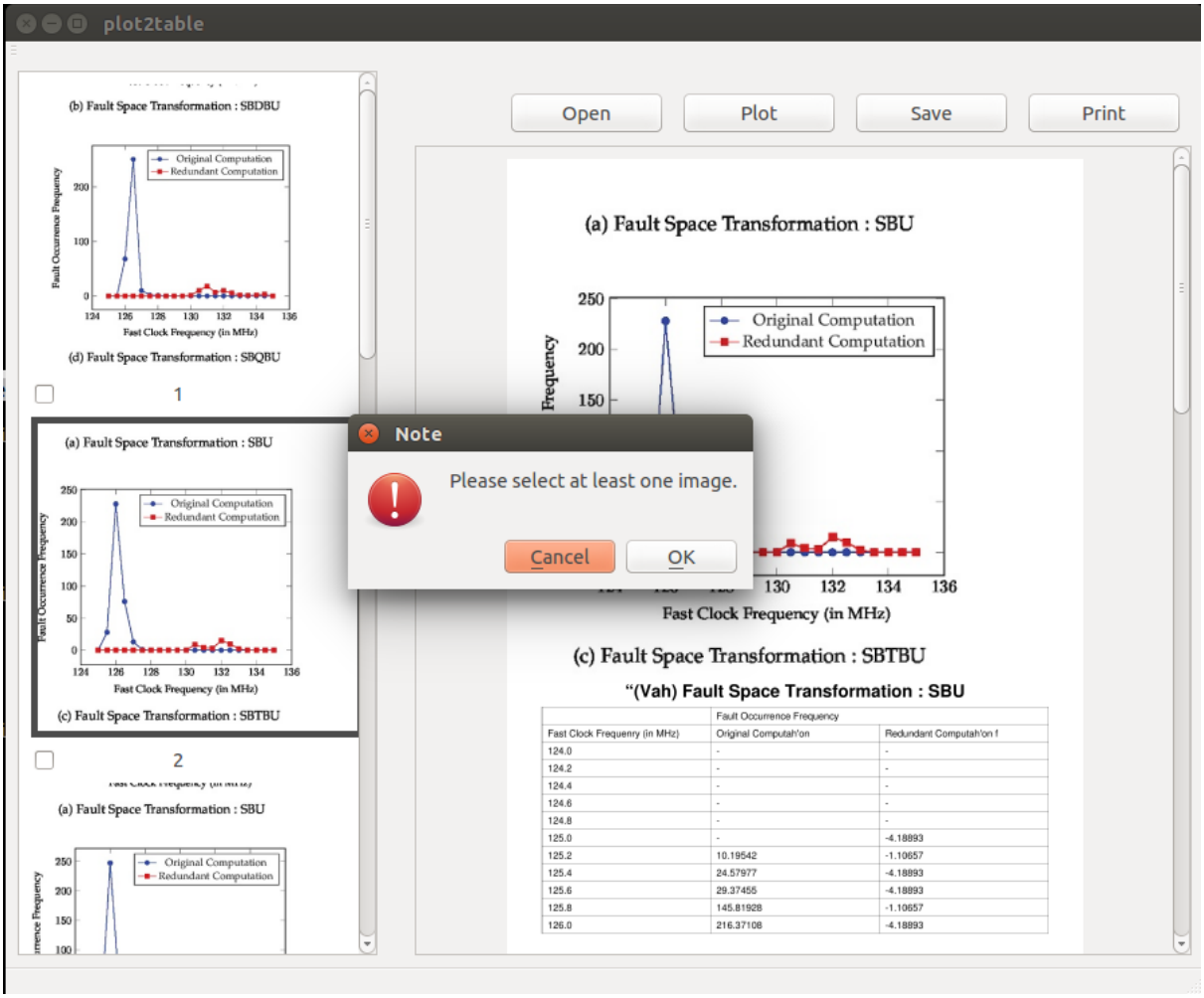
Steps to save the output file are given below:

1. If you click on any button except "Open" without any input PDF File then you get this warning.

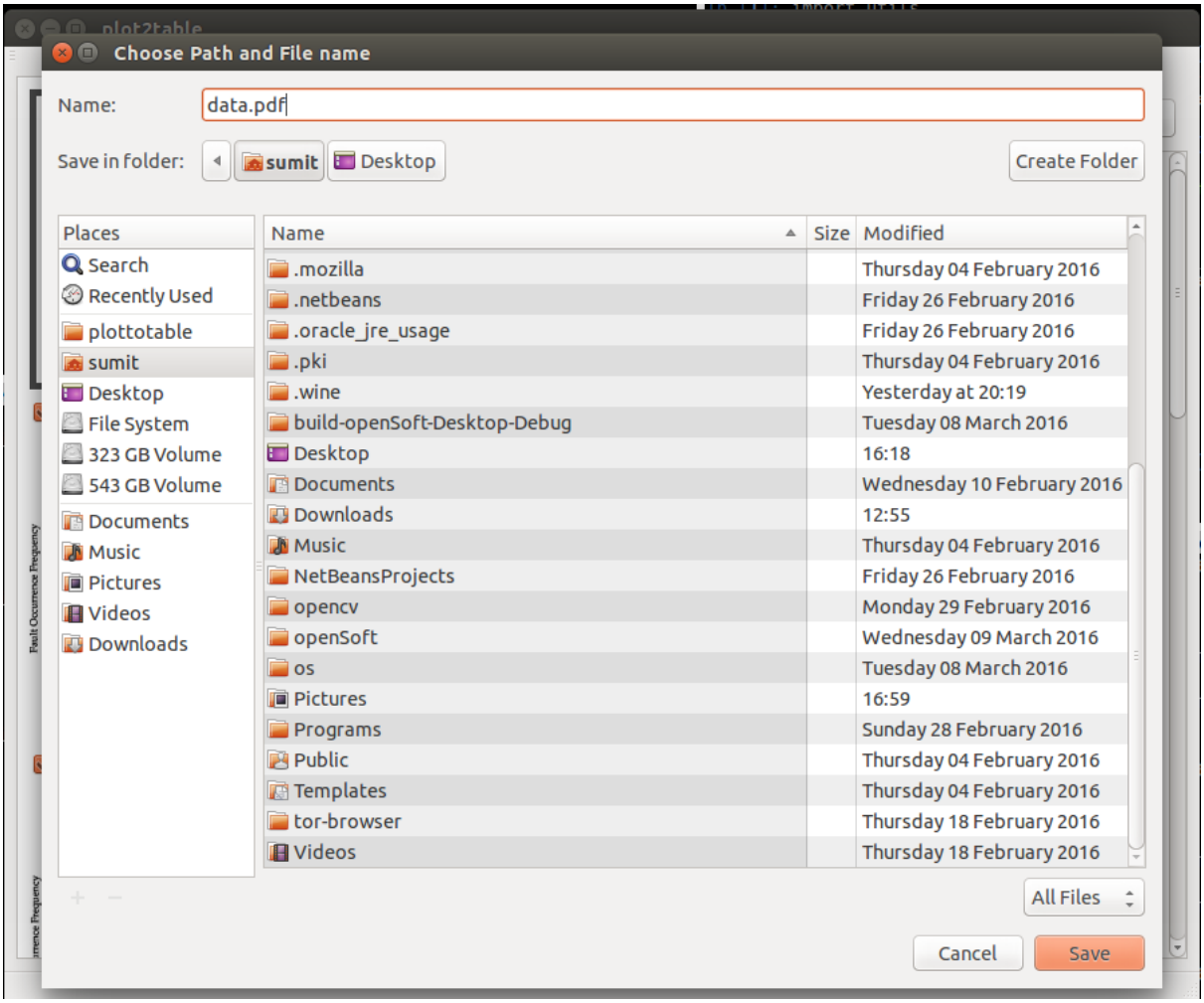


2. Select the plot(s) from the image panel for which you want to save the data.

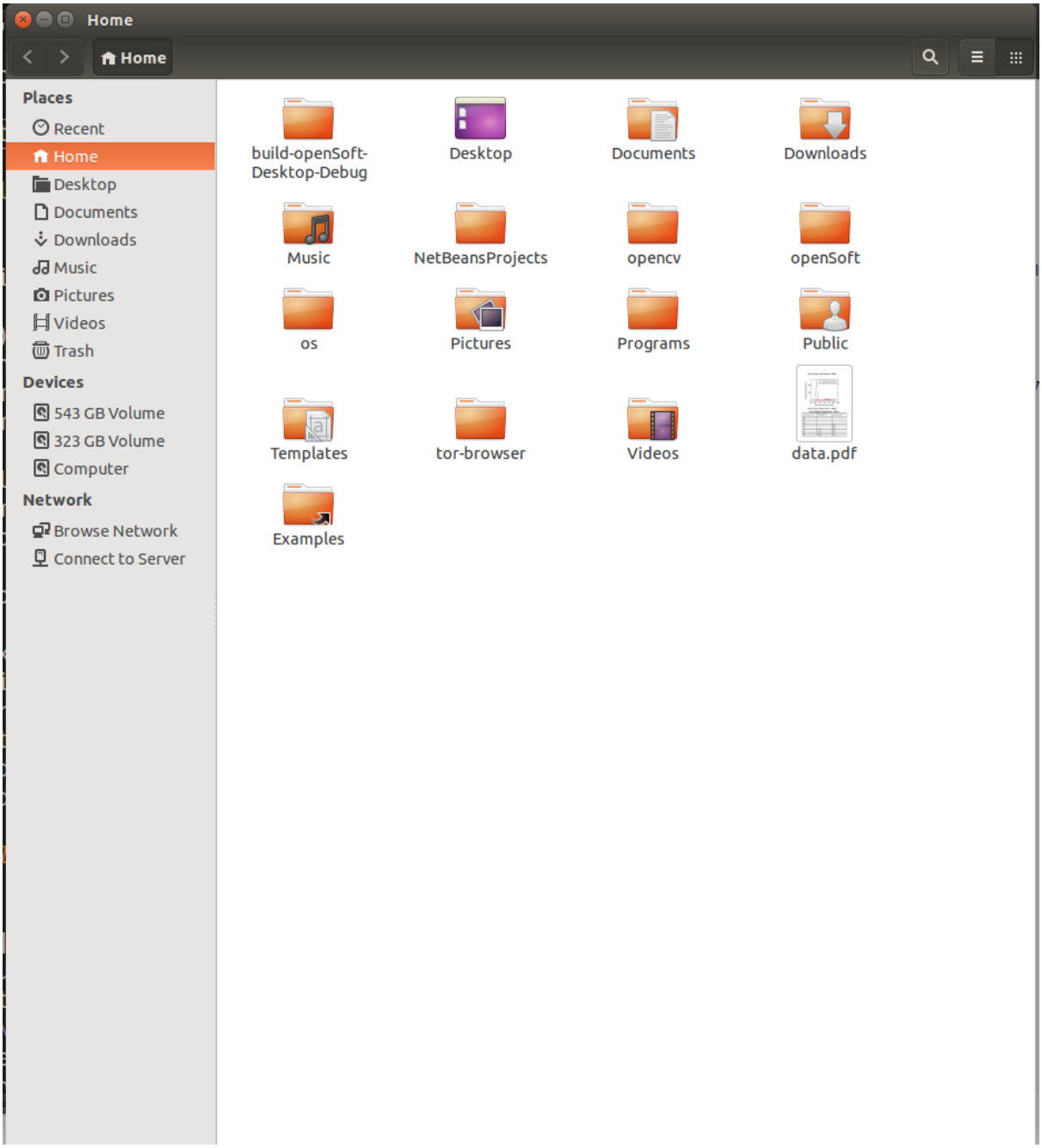
3. Click on the "Save" button.
   If you have not selected any plot



4. Enter the name of the output file and choose directory where you want to save file. Click on "Save" button.



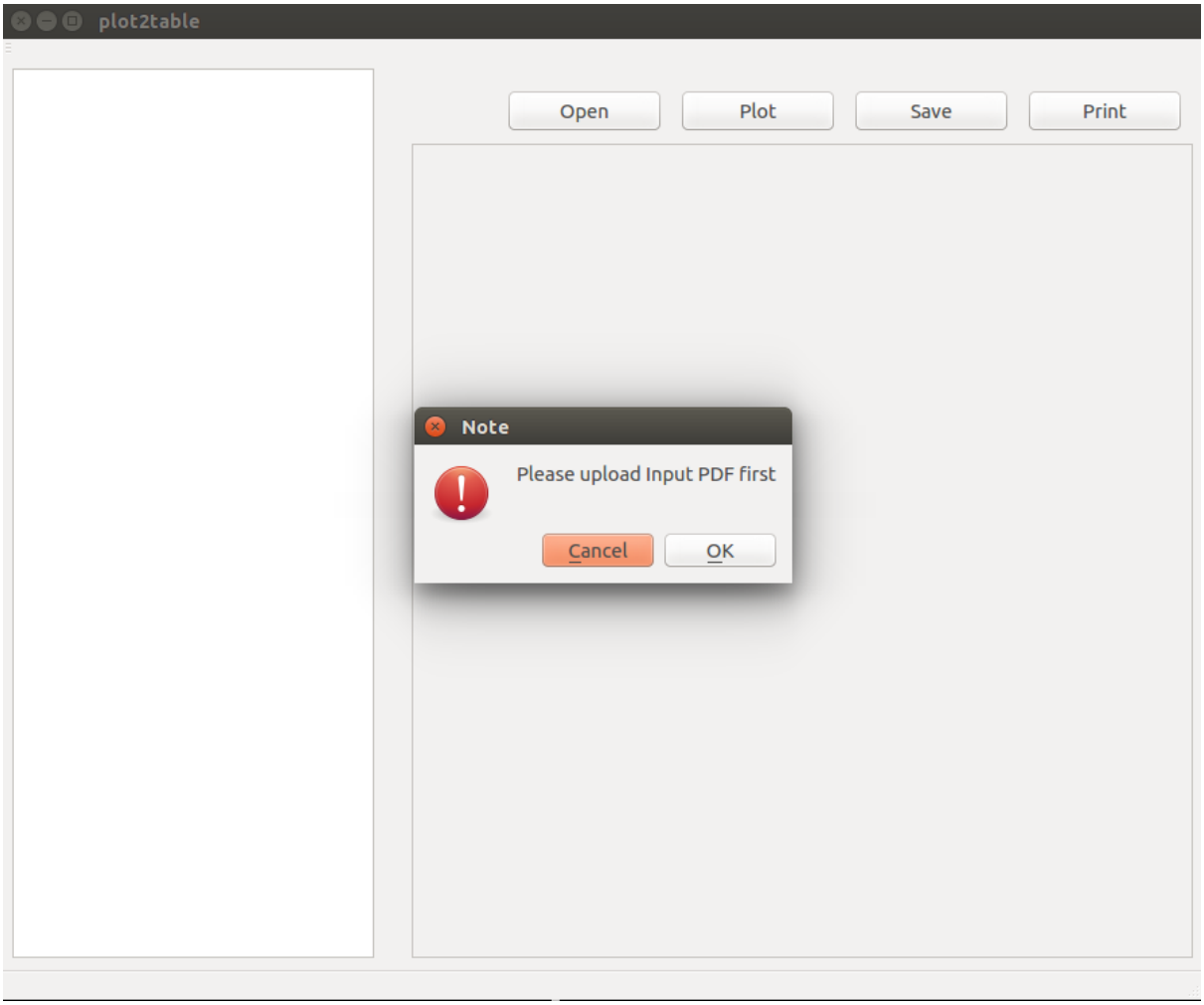
5. File is saved.



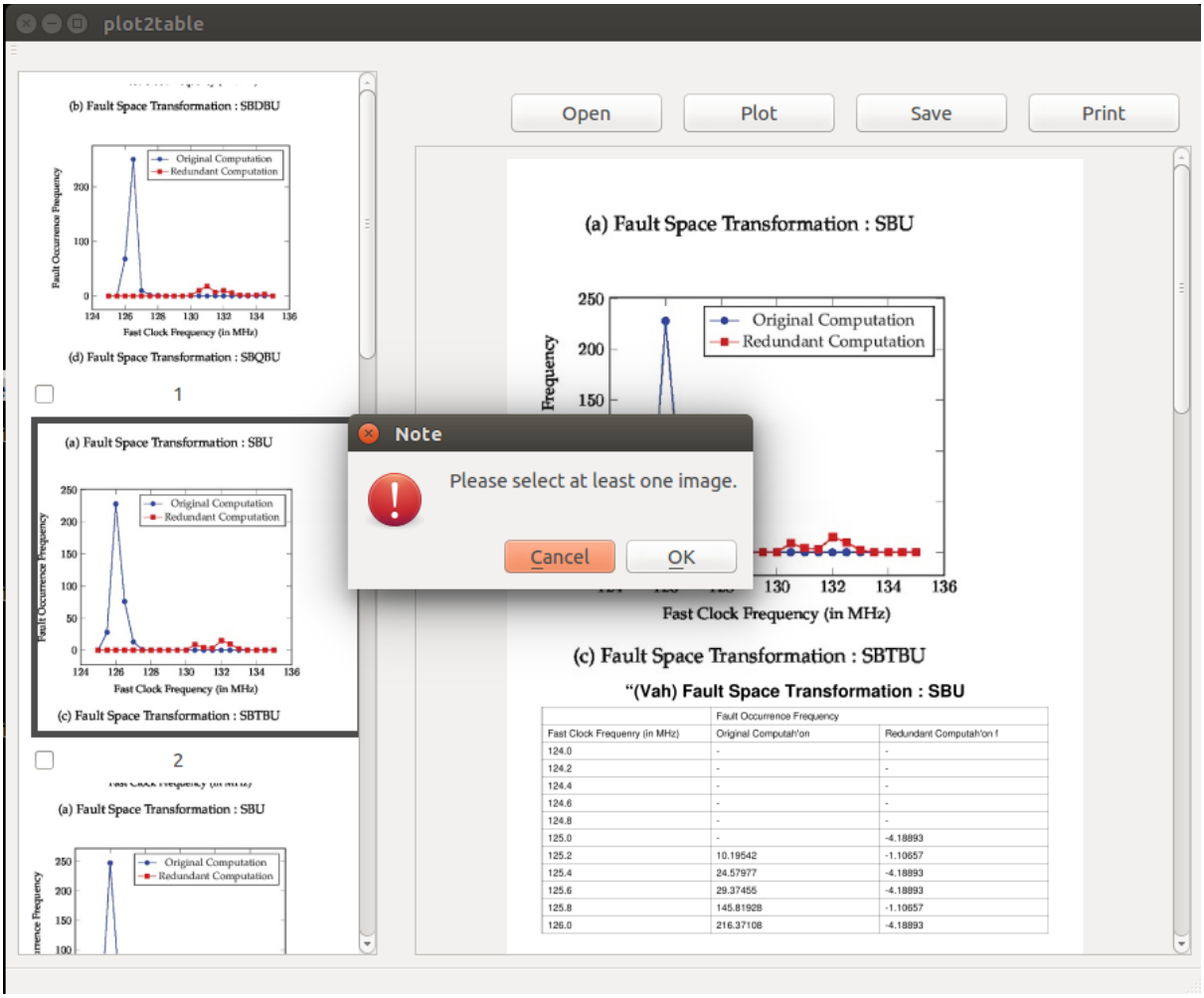
9.4 Print Output File

Steps to print the output file are given below:

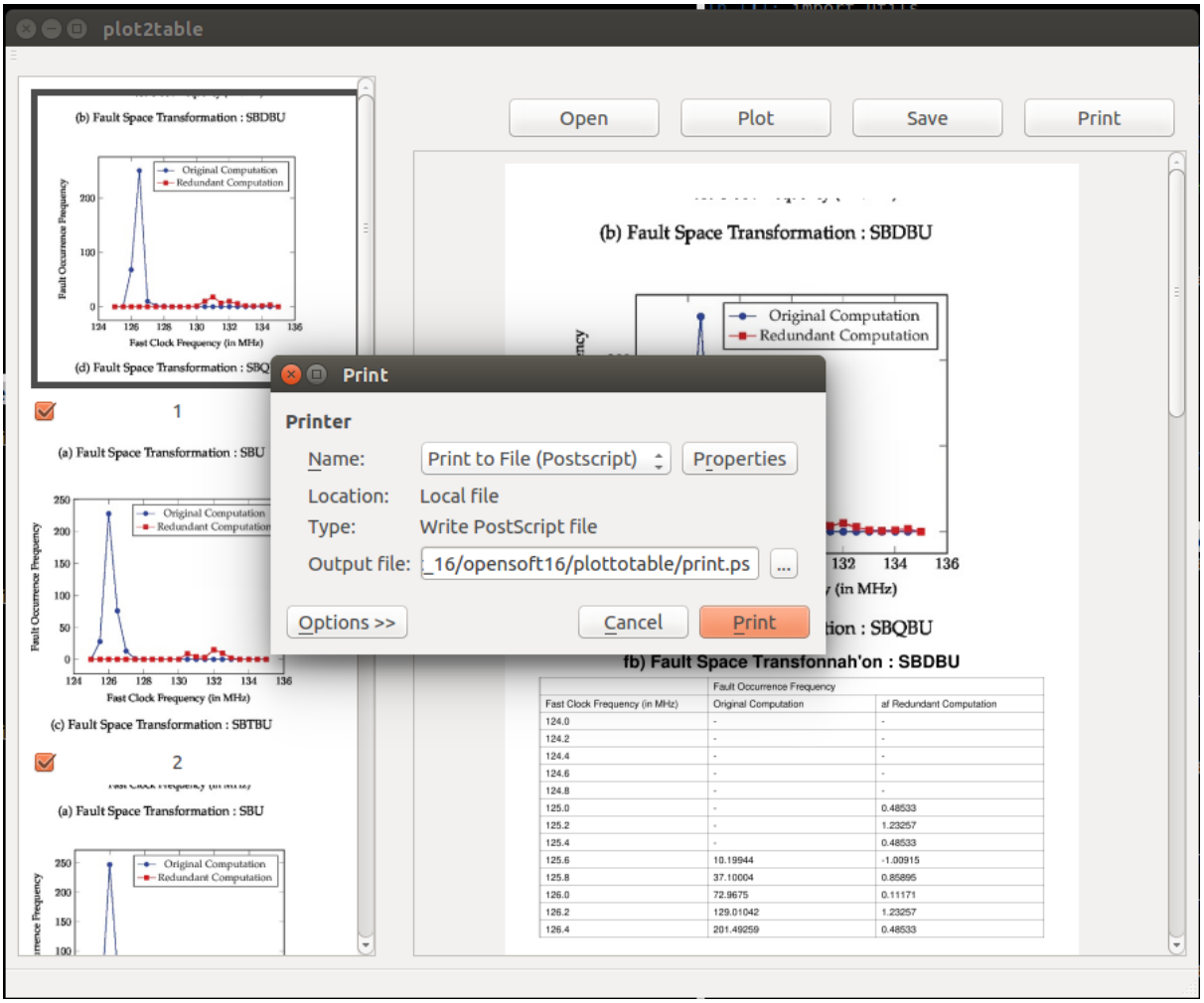
1. If you click on any button except "Open" without any input PDF File then you get this warning.



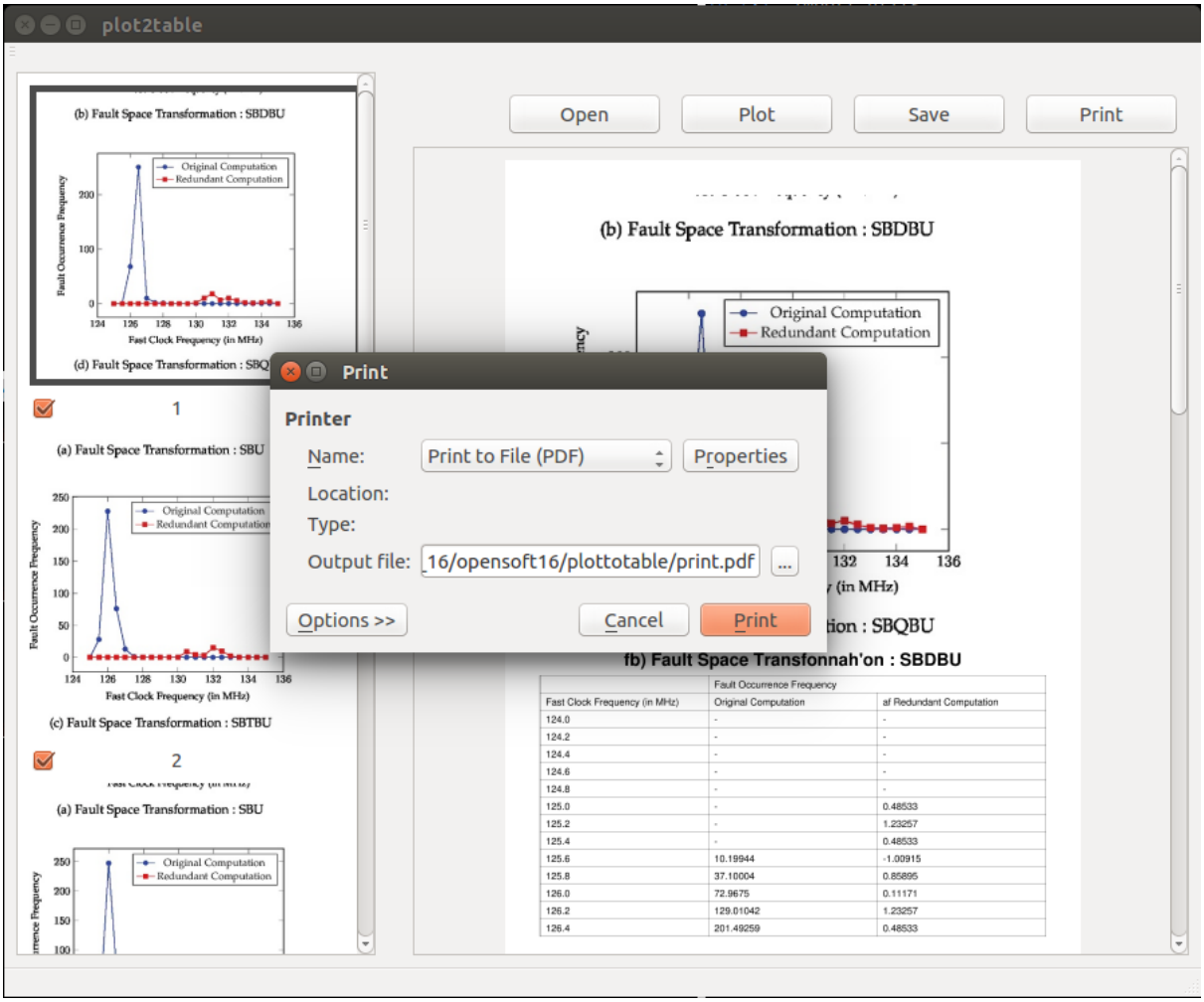
2. Select the plot(s) from the image panel for which you want to save the data.
   If you have not selected any plot



3. Click on the "Print" button.



4. Print to File


plot2table
Open
Plot
Save
Print
(b) Fault Space Transformation : SBDBU
Original Computation
Redundant Computation
Fault Occurrence Frequency
Fast Clock Frequency (in MHz)
(d) Fault Space Transformation : SBQ
1
(a) Fault Space Transformation : SBU
(c) Fault Space Transformation : SBTBU
2
(a) Fault Space Transformation : SBU
Print
Printer
Name:
Print to File (PDF)
Properties
Location:
Type:
Output file:
16/opensoft16/plottotable/print.pdf
Options >>
Cancel
Print
(b) Fault Space Transformation : SBDBU
(in MHz)
ion : SBQBU
fb) Fault Space Transfonnah'on : SBDBU
Fault Occurrence Frequency
Fast Clock Frequency (in MHz)
Original Computation
af Redundant Computation
124.0 - -
124.2 - -
124.4 - -
124.6 - -
124.8 - -
125.0 - 0.48533
125.2 - 1.23257
125.4 - 0.48533
125.6 10.19944 -1.00915
125.8 37.10004 0.85895
126.0 72.9675 0.11171
126.2 129.01042 1.23257
126.4 201.49259 0.48533

Part IV

Algorithms

Chapter 10

Pre Processor

10.1 Introduction

In this part we extract images from PDF file and improve their features for better results.

10.2 Generating Images

We generate images of different plots from PDF file. This method does the following things. It converts input PDF into ppm. It uses *multicrop* to extract images from ppm files one by one. It enlarges images which have height and width both less than a threshold.

10.3 Resizing Image

In this part we increase the size of images. It increases the height and width of image but keeping them less than or equal to 1000. It helps in better character recognition.

10.4 Morphological

We enhance the given image by using erosion followed by dilation.

10.5 Clustering similar Colors

We try to decrease the number of colors in the image by using K-Means clustering . This eases the process of detecting different lines in given plot.

10.6 Sharpening Image

We increase the contrast of the given image.We use bilateral filter to make the given image smooth and then overlap the original image and filtered image to get the sharpen image.

10.7 Otsu's Binarization

To binarize(only black and white) the gray scale image we use otsu's Binarization. It makes the pixel white if pixel value is greater than otsu's threshold otherwise makes it black.

10.8 Histogram

In this part we find which colors are dominant in an image. We calculate which pixels of image is similar to the center of different clusters. We decide similarity of two pixels using hue values.

10.9 Enlarging Image

We enlarge images to increase the size of vector images if there are any so that we can easily read text from it. So we convert PDF file into two different ppm files. In first type we zoom the PDF file and then convert PDF file to ppm while in other we convert the original PDF file to ppm. Next, it checks if size of big image is less than threshold if it is more than that then it rejects the big image and keep the small image. If small image is garbage Image then it rejects both the images.

10.10 Hue Measurement

We use this method to find hue of a color pixel. This method is used in histogram.

Chapter 11

Garbage Image

11.1 Introduction

In this part we find whether given image is a garbage image. It checks if given image has horizontal axis and vertical axis.

11.2 Checking Axis

In this part we check if the given image has horizontal and vertical axis. To detect lines first we draw edges of different contours and then we detect lines. If there is a line having theta between minimum vertical threshold and maximum vertical threshold then vertical axis is present. Similar criteria for horizontal axis.

Chapter 12

Legend Detector

12.1 Introduction

Function which maps colours of plots to legends.Input parameter is a plot with the axes cropped out. This ensures that all text if any in the image must belong to the legend.

12.2 Pre-process

The image is processed before using teseract-ocr. The image is resized and then sharpened. The image is binarised by using Otsu's binarization for better results from the OCR.

12.3 OCR

12.3.1 Generate hOCR

We run tesseract on the image to generate the output in hOCR format.The output is a HTML file with the detected text and also its location in the image.

12.3.2 Get OCR Lines

The method getOCRLines takes a hOCR file as input. It parses the hOCR file to return a list of ocr lines each containing one or more ocr words.

12.3.3 Orientation

There are two orientations possible for the legends. Text followed by the colour or vice-versa. The text part is identified by a regular expression having having

atleast one alpha-numeric character(via function isWord).For all the ocr lines we can determine the orientation - whether it is right orientated or left. If the number of right oriented ocr lines are in majority then all the lines are considered as right oriented. Similarly for left oriented. For equal number of left and right oriented lines the orientation is assigned as NONE.

12.3.4 Get Legends Box

Returns the coordinates of the top left and bottom right points of the legend box.We then also find out the y coordinates of each individual legend.

12.4 Fetch Legend Colors

The words and colour for the legend are extracted according to the orientation.If the orientation is not NONE then the colour is extracted from the right part of the ocr lines for Right orientation and from the left for Left orientation. If the orientation is NONE then we use the local orientation for all ocr lines i.e. considering individual orientation for each ocr line. Finally it returns a dict of list(text to colour) and orientation.

Chapter 13

Title Detector

13.1 Introduction

The title of the corresponding graph is extracted from the image by the following process

13.2 Get Title

13.2.1 Formatting Image

The original image is cropped left and right to get rid of the stray text by just considering the portion of the image confined within the two Y -axes(with a deviation of 50 pixels).To facilitate title detection the image is converted to *grayscale* and the image is negated.

13.2.2 Scanning Image

There can be two possible positions where the title can be found, it could be either above the graph or below the graph.Each row of pixels is scanned from the top X-axis of the graph and the percentage of white pixels is calculated for each row.A similar process is followed for the below part of the image excluding the labels and the scale numbers which would be present starting from the X -axis to the bottom of the image.

13.2.3 Histogram Cut

At first there will be the white pixel percentage would be much lesser than global white pixel percentage which corresponds to the blank space between the label and the X axis. And then there will be a sudden spike in the white

pixel percentage which could correspond to the title.And then any white pixel percentage above this would be stray text.The above process corresponds to histogram cut.

13.2.4 Extracting Title

It there exists no maxima in white pixel percentage then there exists no title.The predicted text area is then sent to tesseract to extract the title.

Chapter 14

Axis Detector

14.1 Introduction

The axis of the image are the immediate lines which are parallel to the image's height and width.Parallel lines in between are assumed to be those belonging to the graph.

14.2 Getting Lines From Plot

With the use of the *getPointsOnAxis* all the lines present in the image are obtained.Before processing the image is converted to *grayscale* from BGR

14.2.1 OpenCV Functions

The image is eroded with a 5*5 kernel to remove undesirable noise from the image.Followed by dilation to fill few gaps that could connect the broken lines in the images.Canny edge detection is performed on the resulting image to get a smooth image which filters edges according to gradient.Applying Hough Line transform on the image would return the equations of all the lines present in the image.

14.3 Axes Detection

The lines which qualify to be axis are those which are atleast 15 pixels away from the borders of the image. And are approximately parallel to either the height or the width of the image.All horizontal lines and vertical lines are put in array in increasing distance from the y-axis and x-axis of the image.The

nearest and the farthest of the lines in the two arrays constitute the 4 axis of the image.If no such exists then null is returned.

Chapter 15

Calibrator

15.1 Introduction

The calibrator helps to detect the labels and the axis of the plot from the image with its corresponding scale,gradients and min,max values.

15.2 Getting Vertical/Horizontal Axis Data

This process would be to extract data from the X-axis and the Y-axis of the obtained sub-image from the *getVerticalSplit*.The element in the hOCR block is being split with the delimiter ',' giving the values on the axis as well as the x and y coordinate values of the value points.

15.2.1 Gradient

A 2-D array is then constructed with the values and their respective y co-ordinates. The gradient is then found as the quotient between the y values and their y coordinates and with respect the scale.

15.2.2 Reference Point

The point with respect to which the gradient was calculated.The reference point and the gradient is finally returned.

A similar procedure is followed to get the X - axis Data.

15.3 Getting Label Data

15.3.1 Processing Y Axis

The Y axis and its label are processed and its corresponding data is extracted with the above functions. They are also saved as separate images.

15.3.2 Processing X Axis

The Y axis and its label are processed and its corresponding data is extracted with the above functions. They are also saved as separate images.

Chapter 16

Color K-Means

16.1 Introduction

In this part we find the maximum variance of clusters of different colors build by using K-Means Clustering.

16.2 Color Cluster

We use this method to make cluster of colors of given image using K-Means Clustering and return the maximum value of variance of clusters. To do this, we first convert image from BGR to RGB. After that we cluster the pixels by using K-Means Clustering and then find the maximum value of variance and return it.

Chapter 17

Generate Table

17.1 Introduction

In this part we extract data from plots and store it in tabular form in PDF file.

17.2 Color Distance

17.3 Generating Data from Plots

We generate data of a plot in this part. To do so, we find all legends' features in a plot. We traverse x-axis from left to right and for a point on x-axis we traverse y-axis from bottom to top. If we find a legend's color at some point then we store the coordinate of that point for that legend. If we do not get data for some legend at some point on x-axis, we estimate missing data depending upon its surroundings. In this manner we find data for a plot.

17.4 Estimating Missing Data

In this part we estimate data which is not found by normal procedure. For point for which data is not present, we check its right side and then its left . If we find data at any of them then we estimate the data for current point and store it. Same procedure is followed for all the point of different legends.

17.5 Printing Tables

We write data of plot(s) for all legends in tabular form in a PDF file.

Chapter 18

Utility Functions

18.1 Introduction

In this part we have defined methods to understand the features of an image.

18.2 Centroid Histogram

We use this method to generate histogram for clusters. Our histogram plots the number of points assigned to each cluster. After that we normalize the histogram such that area of the histogram become equal to 1.

18.3 Plot Colors

In this method we generate the bar chart for cluster of colors using above mentioned histogram.

18.4 Merge Output

We use this method to merge data for different plots if user selects more than one plot to save/print.

18.5 Clean

In this part we clean all the temporary files build during the processing of input PDF file.

18.6 Plot Data

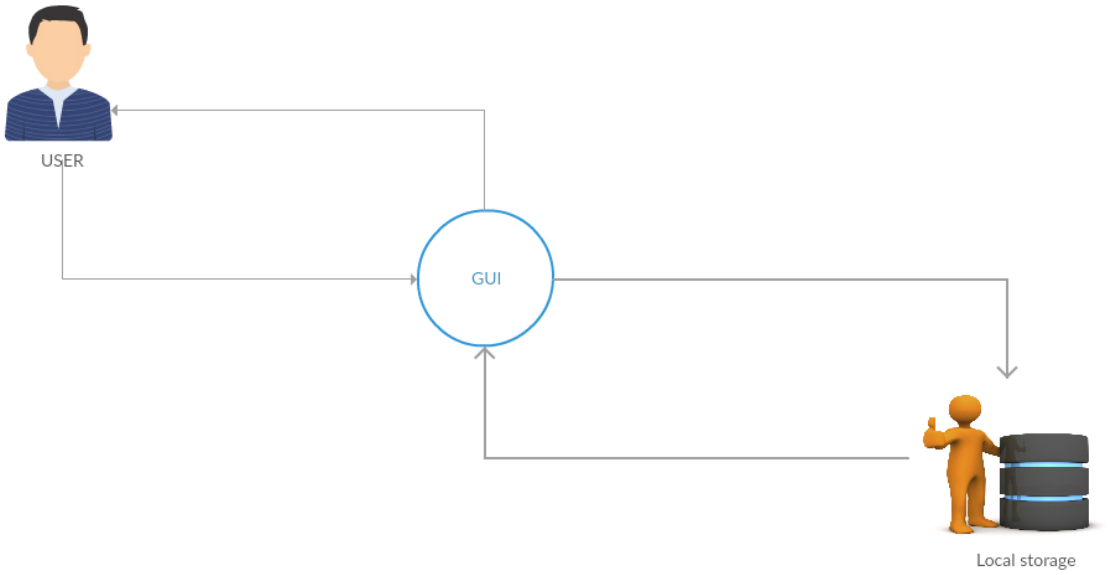
This method plot the data which we have obtained from an input plot.

Part V

Architecture

Chapter 19

Context Diagram

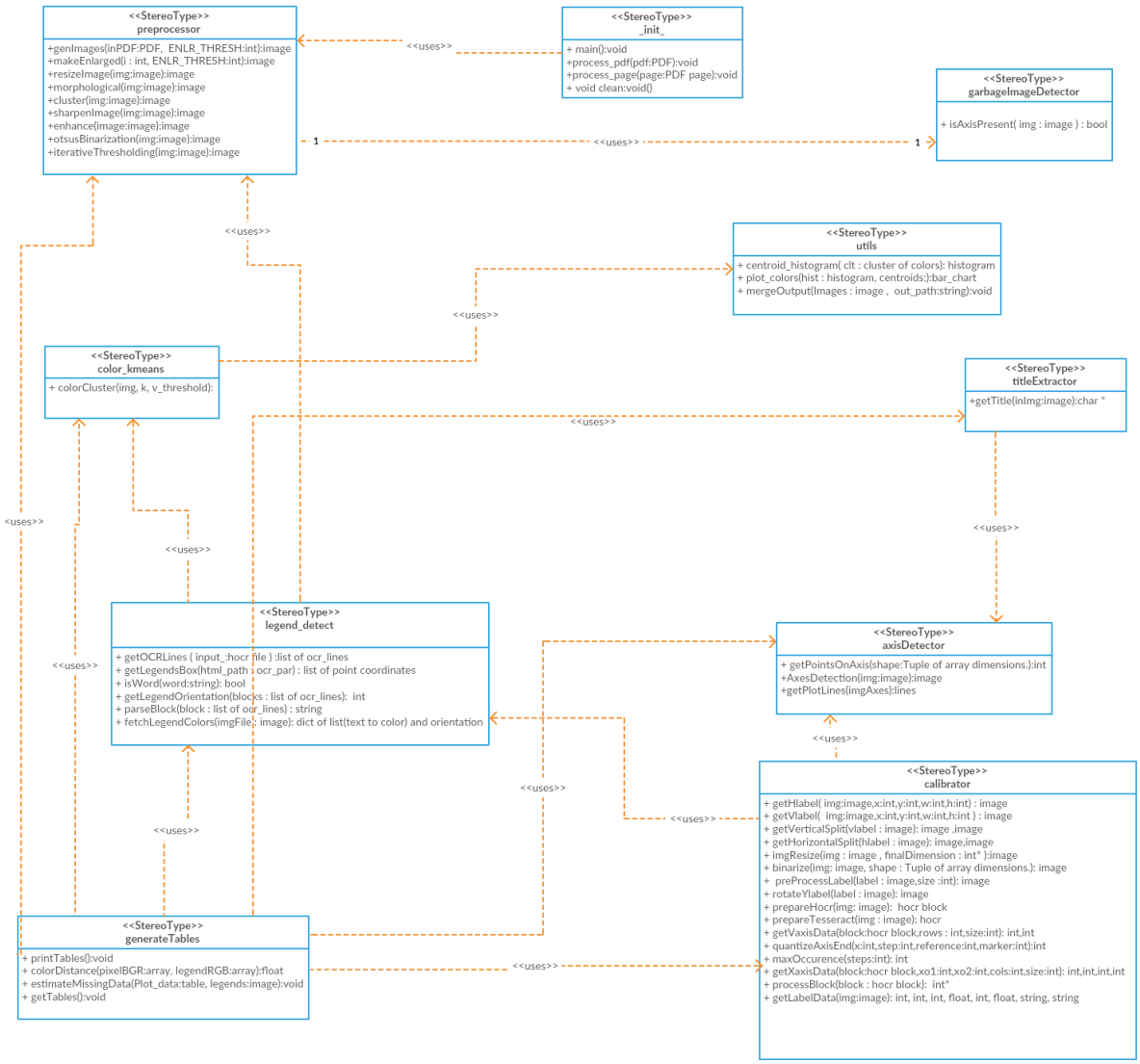


The figure above provides an overview of the functionality of the whole software in a very crude term. It shows the distinct parts of the software and it interacts to effectively put the whole system working. The various parts of the system are explained below:

- GUI: GUI is the interactive and the visible part of the whole software which the user of the software interacts with. It just displays the actions performed by the plot2table.
- User: User is the object which represents the control held by the user.
- Local Storage: This refers to the Local storage where every image and output table is stored in the process.

Chapter 20

Class Diagram

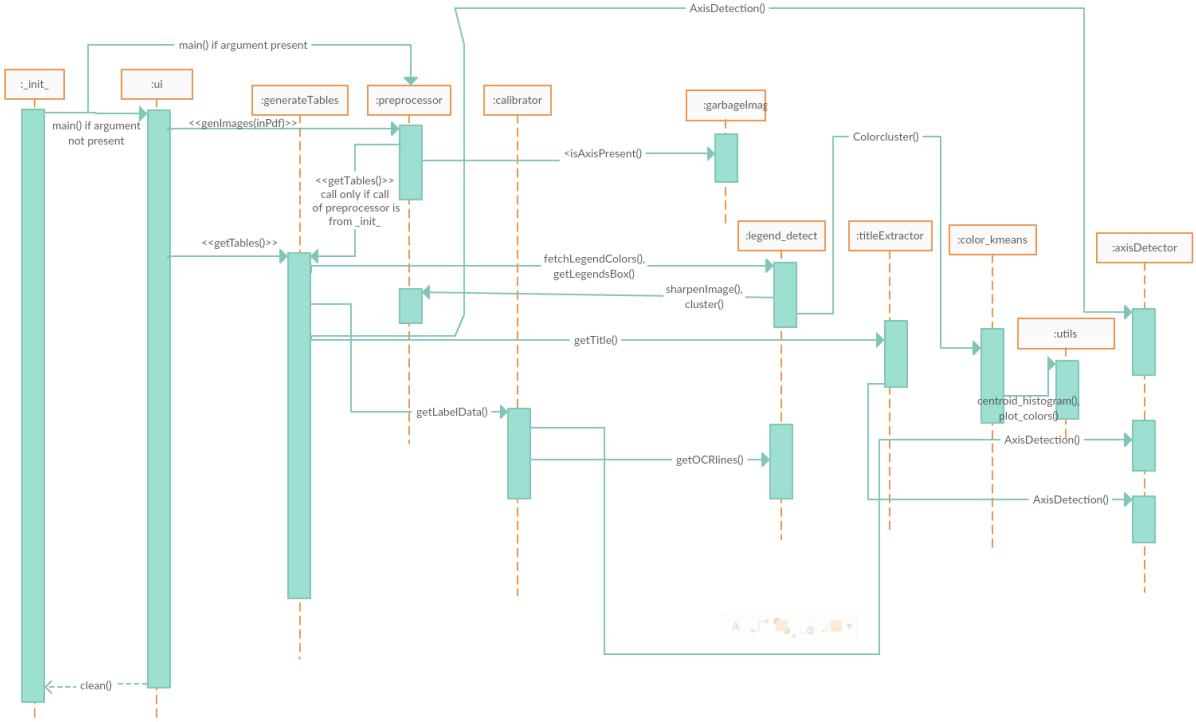


High resolution picture in images folder.

1. There are 9 main classes used in the software.
2. The relationships between the classes are as shown in the diagram.

Chapter 21

Sequence Diagram



High resolution picture in images folder.

Analysis

1.The sequence diagram above shows the time flow of the software.

2.The Program starts with instance of init class.

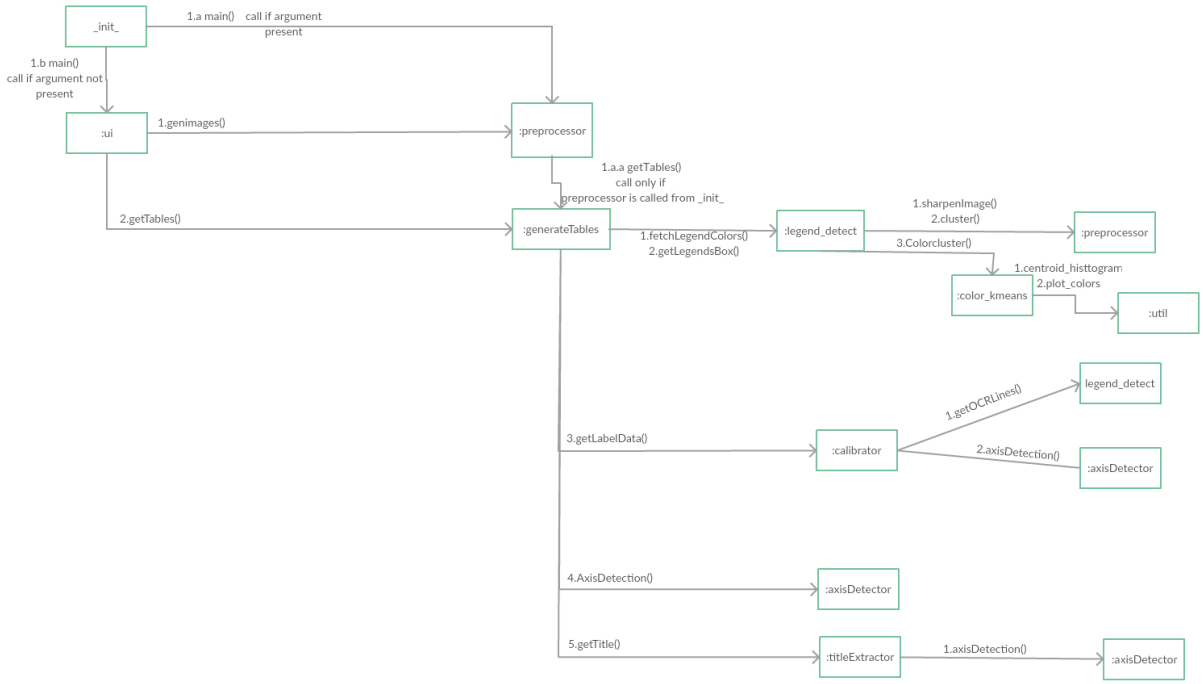
3.The init then may call graphical user interface or may not depending on whether the user calls with or without arguments.

4.If the program doesn't run through GUI, the init object calls an instance of preprocessor class directly .

5.The program then proceeds by message communication between objects as shown in the diagram.

Chapter 22

Communication Diagram



High resolution picture in images folder.

Analysis

1.The figure above shows the communication between instances of different classes of the software.

2.The program starts with init class object.

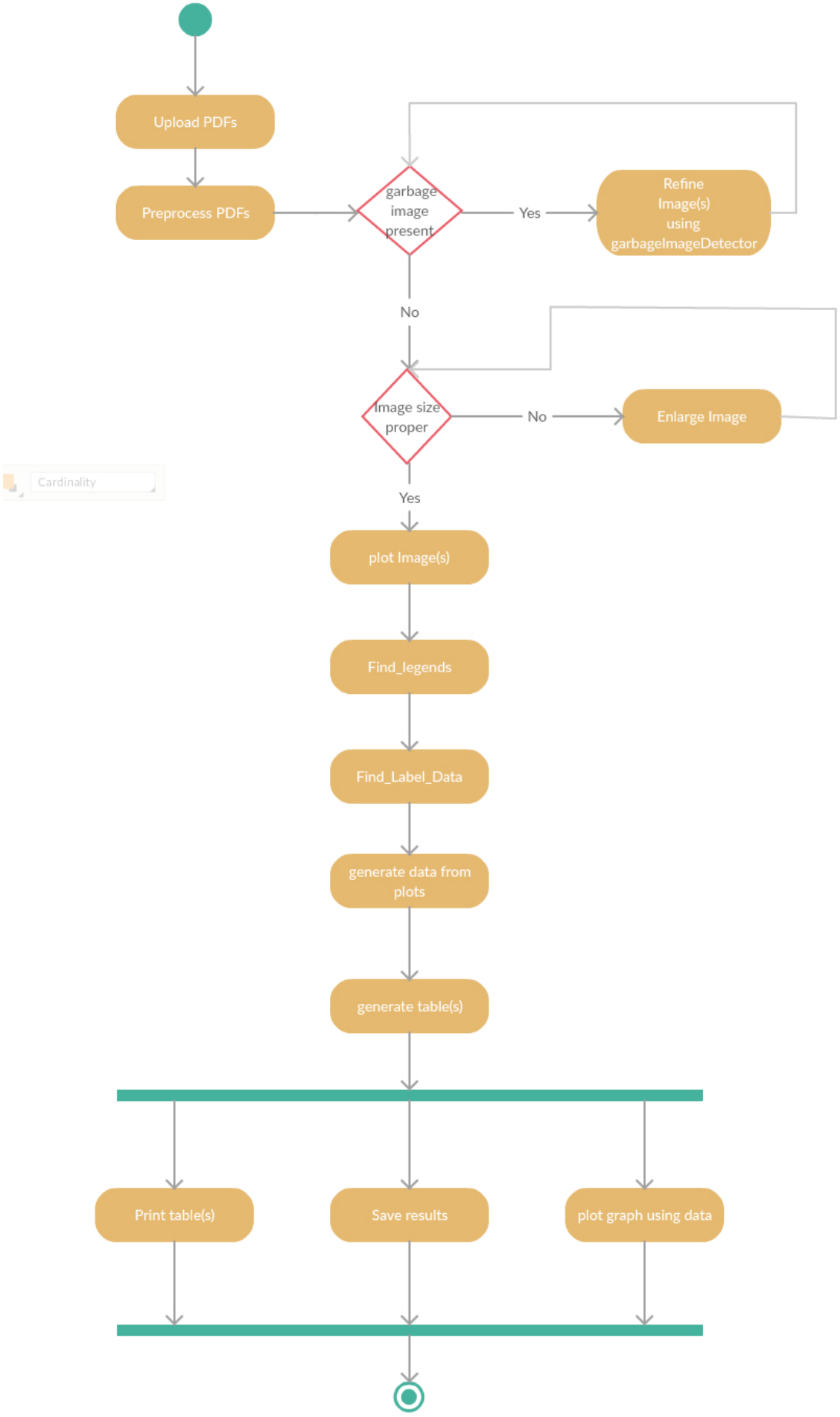
3.The init then may call graphical user interface or may not depending on whether the user calls with or without arguments.

4.If the program doesn't run through GUI, the init object calls an instance of preprocessor class directly .

5.The flow then flows by the communication between objects as shown in the diagram.

Chapter 23

Activity Diagram


Upload PDFs
Preprocess PDFs
garbage image present
Yes
Refine Image(s) using garbageImageDetector
No
Image size proper
No
Enlarge Image
Cardinality
Yes
plot Image(s)
Find_legends
Find_Label_Data
generate data from plots
generate table(s)
Print table(s)
Save results
plot graph using data

High resolution picture in images folder.

Analysis :

1.The diagram above shows the various activities involved in the execution of the software.

2.The software starts when the user uploads input PDFs.

3.Then the PDFs are preprocessed so as to remove any garbage images(if generated any) and enhance the size of images generated(if not proper). Then respective images are plotted .

4.Then the software tries to identify legends and color of various entities present.

5.Then it tries to find x label data, y label data,x axis gradient,y axis gradient,x reference point , y reference point from the image .Also it searches for the title of the graph.

6.Then it tries to find respective y axis values for each x axis values. Hence it generates tables in this step.

7.The next step involves saving the results thus obtained, cross checking the input graph by plotting a graph from the output values and print the generated output ,if directed by the user.

8.After these activities have finished, the Software proceeds towards it's finish.

Part VI

Test Plan

Chapter 24

Introduction

This document is a high-level overview defining testing strategy for the plot2table software. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application. Testing criteria under the white box, black box, and system-testing paradigm will be utilized. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design. Throughout the testing process the test documentation specifications described in the IEEE Standard 829-1983 for Software Test Documentation will be applied.

Chapter 25

Test Objective

The objective of this test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, a broad range of tests will be exercised to achieve the goal. There will be following functions that can be performed by this application: Read PDFs, Extract plots, select multiple plots to plot tables for, generate PDF file containing tables. The user interface to utilize these functions is designed to be user-friendly and provide easy access to all the functions.

To check the integrity of the software and verify if it's working properly, go to the root of the project, and in Terminal, execute

```
$ nosetests -v
.....................................
----------------------------------------------------------------------
Ran 31 tests in 13.992s

OK
```

`nose` is a python library used for extensive unit testing and picking out code smells.

Chapter 26

Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

Chapter 27

Testing Strategy

The following outlines the types of testing that were done for unit, integration, and system testing. It includes what will be tested, the specific use cases that determine how the testing is done.

27.1 Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

27.1.1 White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test cases are generated that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once.

Testing the preprocessor module

A module test_preprocessor was implemented as a test bench for the preprocessor class. The module contained the following methods:

- test_resizeImage() - Verifies the function resizeImage() of the preprocessor module.

- test_sharpenImage() - Verifies the function sharpenImage() of the preprocessor module by comparing the input image and the image returned by the function.
- test_enhance() - Verifies the function enhance() of the preprocessor module.
- test_morphological() - Verifies the function morphological() of the preprocessor module.

Testing the axisDetector module

A class TestAxisDetector was created to test the axisDetector module which contains the following methods:

- test_getPlotLines() - This method asserts the attributes of the generated axes with the required values from selected images.
- test_AxesDetection() - This method tests the function AxesDetection()

Testing the legend_detect module

A class TestLegendDetect was implemented to test the legend_detect module. The class contains the following methods:

- test_fetchLegendColors () - This function compares the colors of the legends of an image with the output generated by the function fetchLegendColors.

Testing the garbageImageDetector module

A class TestGarbageImageDetector was implemented to test the garbageImageDetector module. This class contains the following methods:

- test_isAxisPresent() - to check the functionality of the isAxisPresent() function
- test_isTable() - to check the functionality of the isTable() function.
- test_isGarbageImage() - to test the functionality of isGarbageImage() function.

Testing the Calibrator module

A class TestCalibrator was implemented to test the calibrator module. This class contains the following methods:

- test_imgResize() - to check the functionality of the imgResize() function
- test_binarize() - to check the correctness of the binarize() function.
- test_getLabelData() - to assert the correctness of getLabelData() function by comparing the results with a known image.

Testing the color_kmeans module

A class TestColorKMeans was implemented to test the color_kmeans module. This class contains the following methods:

- test_colorCluster() - to test the results of the colorCluster() function on known images.

All the tests were rigorously scrutinized and were successful.

27.1.2 Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Error guessing and Boundary Value Analysis testing on our application.

27.2 Integration Testing

27.2.1 Incremental Testing

There are two primary modules that were needed to be integrated: the Graphic User Interface module and the plot2table (back-end). The two components, once integrated form the complete plot2table Application. The following describes these modules as well as the steps that will need to be taken to achieve complete integration. We will be employing an incremental testing strategy to complete the integration.

Module 1 - Graphic User Interface (GUI) Module. This module provides a simple GUI where the user can perform the different actions (functions). This module was tested separate from the back end to check if each interface (e.g. search button) is functioning properly, and in general, to test if the mouse-event actions were working properly. The testing was performed by writing a stub for each element in the interface.

Module 2 - plot2table convertor Module The convertors provide functions to extract plots from PDFs and tabulate their values to return to the GUI. This module was tested separate from the GUI by taking out the PDFs separately. In testing this module we followed the incremental testing method i.e. testing one function first and then keep adding additional function and test it again until all the required functions are tested.

When the GUI is combined with the back end module, we will have a complete plot2table application. To achieve complete integration of these two modules, we test each element in the GUI by replacing the stubs with the appropriate function from the back end. The results were displayed within the GUI instead of through the Console. In testing the combined modules, we followed the incremental testing method. Each stub will be replaced one at a time and tested. This was done until all stubs have been replaced by the appropriate functions from the back end.

27.3 System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case we focused only on function validation and performance. And in both cases we used the black-box method of testing.

27.3.1 Function Validation Testing

The integrated p̈lot2tableẅas tested based on the requirements to ensure that we built the right application. In doing this test, we tried to find the errors in the inputs and outputs, that is, we tested each function to ensure that it properly implements the extraction and processing procedures, and that the results are expected. The behavior of each function are contained in the Software Requirement Specification.

Function	Expected Behavior
Read PDF	see Software Requirement Specification
Extract Plots	see Software Requirement Specification
Select Plots to tabulate	see Software Requirement Specification
View tables	see Software Requirement Specification
Print output PDF	see Software Requirement Specification

In addition, we tested:

- The interfaces to ensure they are functioning as desired (i.e. check if each interface is behaving as expected, specifically verifying the appropriate action is associated with each mouse_click event).
- The interaction between the GUI and the back end convertor modules. In this case the images will be inserted and check if they are processed in the back end and give the expected output.

27.3.2 Performance Testing

This test was conducted to evaluate the fulfillment of a system with specified performance requirements. It was done using black-box testing method. The following were tested:

- Taking input PDF with large number of plots and checking how much time the application take.
- Taking input PDF which is densely populated and has small plots.

Chapter 28

Entry and Exit Criteria

This section describes the general criteria by which testing commences, temporarily stopped, resumed and completed within each testing phase. Different features/components may have slight variation of their criteria, in which case, those should be mentioned in the feature test plan. The testing phase also maps to the impact level definition when a defect is entered in the bug-tracking phase.

28.1 Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

28.1.1 Black Box Phase

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We will use Equivalence Partitioning and Boundary Value Analysis complexity metrics in order to quantifiably determine how many test cases needed to achieve maximum code coverage.

Black Box Entry Criteria

The Black Box Entry Criteria will rely on the component specification, and user interface requirements. Things that must be done on entry to the Black Box stage:

- All functions like Read PDFs, Extract plots, view corresponding tables, select multiple plots to plot tables for, generate PDF file containing tables must either be coded or stubs written.
- The type of Black Box testing Methods will be determined upon entry. We will use Error Guessing, and Boundary Value Analysis.
- Error Guessing included entering garbage string in search field, trying to add the same entity multiple times to favorite, closing the internet connection and searching etc.
- Boundary Value Analysis included setting high value(like 5000) for maximum results to be displayed, opening large number of tabs and searching simultaneously.

Black Box Exit Criteria

The Black Box Exit Criteria listed below explains what needs to be completed in-order to exit Black Box phase. To exit the Black Box phase 100% success rate must be achieved. Things that must be done upon exiting the Black Box stage:

- The application showed the message if no results are found in case of garbage string.
- The applications searched on a maximum of 12 - 15 tabs simultaneously beyond which a crash might occur which was handled.
- All code bugs that are exposed are corrected.

28.1.2 White Box Testing

The White Box criteria apply for purposes of focusing on internal program structure, and discover all internal program errors. Defects will be categorized and the quality of the product will be assessed.

White Box Entry Criteria

The White Box Entry Criteria will rely verifying that the major features work alone but not necessarily in combination; exception handling will not be implemented. The design and human interface are stable. Things that must be done on entry to the White Box stage:

- All functions Read PDF, Extract plots, select multiple plots to plot tables for, generate PDF file containing tables- must be coded.
- The type of White Box testing Methods was determined upon entry. We will use unit testing and test for memory leaks.
- Black Box Testing should be in its late stages.

White Box Exit Criteria

The plot2table software in the White Box stage should have a generally stable feel to it. White Box testing continued until the Black Box or next milestone criteria were met. To exit the White Box phase 100% success rate must be achieved. The following describes the state of the product upon exit from the White Box Stage:

- All functions- Read PDF, Extract plots, select multiple plots to plot tables for, generate PDF file containing tables- are implemented, operational and tested.
- All Branch Testing test cases were generated. The test cases will be generated from the Control Flow diagrams of all functions.
- The graphical interface has been reviewed and found to satisfactory and is stable, that is, no further changes to dialog boxes or other interface elements are planned. Minor changes are acceptable, but must be arranged with the Development and Test Engineers.
- All code bugs that are exposed are corrected.

28.2 Integration Testing

There are two modules that will be integrated for Integration Testing. The two modules are The Graphic User Interface module and the table generators (back-end). The two components consists of a mixture of stubs, driver, and full function code. The following describes the entry and exit criteria for Integration testing.

28.2.1 Integration Test Entry Criteria

The Integration Test Entry Criteria will rely on both modules to be operational. The plot2table convertor and human interface must be stable. Things that must be done on entry to the Integration Test stage:

- All modules functions - Read PDF, Extract plots, select multiple plots to plot tables for, generate PDF file containing tables- must either be coded and/or stubs created.
- The Graphical User Interface must either be coded and/or a driver and stubs must be created. The driver is implemented to facilitate test case input and output values.
- Interfaces and interactions between the plot2table convertor and the Graphical User Interface must be operational.
- A bottom-up Integration Test Strategy will be conducted. The low level details of the convertor and graphical interface were integrated. A driver will be written to facilitate test case input and output values. The driver temporarily satisfied high-level details of the input and output values.

28.2.2 Integration Test Exit Criteria

The Integration Test Exit Criteria relied on both modules to be operational. The plot2table convertor and human interface must be stable. To exit the Integration Testing phase 100% success rate must be achieved. Things that must be done on exit from the Integration Test stage:

- All code bugs that were exposed were corrected.
- The convertor and Graphical User Interface Module interacted together with complete accuracy, according to the System Specification Design. All discrepancies were corrected.
- Both Modules are ready for System Testing. Stubs and drivers are replaced with fully functional code.
- Black Box Testing was completed.

28.3 System Testing

The System Test criteria apply for purposes of categorizing defects and the assessing the quality level of the product. All elements of the plot2table Module and Graphical User Interface are meshed together and tested as a whole. System test focuses on functions and performance, reliability, installation, behavior during special conditions, and stress testing.

28.3.1 System Test Entry Criteria

- Unit testing of all the modules has been completed.
- System test cases have been documented.
- Specifications for the product have been completed and approved.
- UI and functionalities to be tested should be frozen.
- Priorities bugs have been fixed.

28.3.2 System Test Exit Criteria

- Application meets all the document requirements and functionalities.
- Defects found during System testing should be fixed and closed.
- All the test cases for the system have been executed and passed.
- No critical defects have been opened which is found during system testing.

28.4 Shipping/Live Release

The Convertor testing is scaled down and combines all phases of testing into two phases - Function Complete and Regression testing - and follows the release criteria.

28.4.1 Shipping/Live Release Entry Criteria

The installer of the software plot2table was rigorously tested by many users and it was found to be installed and executed without any found bugs. The users also tested the basic functionality implemented in the software and no found bugs were detected.

- All open product defects, regardless of fixed defects, documented, deferred, or otherwise addressed were identified.
- Regression testing on all product defects and the entire product has been completed was verified.

28.4.2 Shipping/Live Release Exit Criteria

- No critical defects found.
- Business processes are working satisfactorily.