

WEB-BASED INTERACTIVE EDITING AND ANALYTICS FOR  
SUPERVISED SEGMENTATION OF BIOMEDICAL IMAGES

---

A Thesis presented to  
the Faculty of the Graduate School  
at the University of Missouri-Columbia

---

In Partial Fulfillment  
of the Requirements of the Degree  
Master of Science

---

by  
RAHUL KUMAR SINGH  
Dr. Kannappan Palaniappan, Thesis Supervisor

DECEMBER 2014

© Copyright by Rahul Kumar Singh 2014

All Rights Reserved

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

**WEB-BASED INTERACTIVE EDITING AND ANALYTICS FOR  
SUPERVISED SEGMENTATION OF BIOMEDICAL IMAGES**

Presented by Rahul Kumar Singh,

a candidate for the degree of Master of Science,

and hereby certify that, in their opinion, it is worthy of acceptance.

---

Dr. Kannappan Palaniappan

---

Dr. Jianlin Cheng

---

Dr. Filiz Bunyak

## ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Kannappan Palaniappan for his support and guidance during this project. I completely utilized every opportunity I had during my study at University of Missouri, Columbia with his help and encouragement. I would also like to thank Dr. Stefan Jaeger and Dr. Sema Candemir at the National Institute of Health, Dr. Surya Prasath and Rengarajan Pelapur at the Computation Imaging and Visual Analysis Lab and all my other colleagues for their continuous assistance and advice on this project.

I would like to thank Dr. Jianlin Cheng and Dr. Filiz Bunyak for giving me high-quality knowledge and taking out the time to serve on my project committee. I would also like to thank my parents, brother and sister in law for constantly motivating me during this project and guiding me in the right path in all aspects of life and my new born nephew for bringing immense joy in my life. Finally, I thank all my friends, especially Mansha without whose support at the right moment, this wouldn't have been possible.

## TABLE OF CONTENTS

<b>ACKNOWLEDGMENTS .....</b>	<b>ii</b>
<b>LIST OF TABLES .....</b>	<b>vi</b>
<b>LIST OF FIGURES .....</b>	<b>vi</b>
<b>ABSTRACT.....</b>	<b>xii</b>
<b>Chapter 1 Introduction.....</b>	<b>1</b>
1.1 Brief Introduction.....	1
1.2 FireFly Version 3.0 .....	2
1.3 FireFly Datasets.....	5
1.4 Other Similar Tools.....	10
<b>Chapter 2 FireFly Architecture and Technologies .....</b>	<b>21</b>
2.1 New System Architecture.....	21
2.2 Technologies Used.....	22
2.3 Local Cache and Screen Buffer.....	27
<b>Chapter 3 Multiuser Annotations for Large Image Collections.....</b>	<b>29</b>
3.1 Restructuring of Database .....	29
3.2 Video Sequence vs Image Collection.....	33
<b>Chapter 4 Segments and Contours Editing.....</b>	<b>37</b>
4.1 Need for Segments and Contours Editing.....	37

4.2 Adding Vertex in Contours and Segments.....	40
4.3 Deleting Vertex in Contours and Segments.....	42
4.4 Splitting of Contours and Segments.....	44
4.5 Merging of Contours and Segments.....	48
4.6 Cases Involving Multiple Edit Operations.....	56
4.7 Protection and Editing Mode.....	60
<b>Chapter 5 Web-based Supervised Image Segmentation .....</b>	<b>61</b>
5.1 MATLAB Interfacing .....	61
5.2 Segmentation Results.....	66
5.3 Alpha Blending of Images.....	67
5.4 Loading of Data through PHP Controllers.....	69
<b>Chapter 6 Data Analytics and Visualization .....</b>	<b>71</b>
6.1 Study of Toolkits and Online kits .....	72
6.2 Data Visualization in Flex 3.....	75
6.3 Data Visualization in FireFly.....	77
<b>Chapter 7 Optimizing Web-Based Performance .....</b>	<b>82</b>
7.1 Dataset Locks .....	82
7.2 Internet Explorer vs. Other Browsers.....	83
7.3 User Interface Improvements for Biomedical Image Annotation.....	84
7.4 Incremental Save vs. Bulk Save.....	87

<b>Chapter 8 New Intuitive GUI Tools .....</b>	<b>90</b>
8.1 Creeping of Polygons and Polylines .....	91
8.2 Additional Annotation Objects .....	94
8.3 Redesign of Save/Display Panel .....	99
8.4 Redesign of Frame Advance Panel .....	101
8.5 Redesigning of Tools Panel.....	101
<b>Chapter 9 Import/Export for Image Analysis.....</b>	<b>103</b>
9.1 KW 18 Format.....	103
9.2 Lung Boundaries Format.....	105
9.3 Microvasculature File Format .....	106
9.4 Data Analytics File Format for Microvasculature.....	107
9.5 Region of Interest (ROI) Format.....	109
9.6 Downloading Results .....	111
<b>Chapter 10 Conclusion &amp; Future Work .....</b>	<b>112</b>
10.1 Summary .....	112
10.2 Future Work.....	113
<b>Appendix A Development and Debugging in FireFly.....</b>	<b>118</b>
<b>Appendix B FireFly Manual .....</b>	<b>121</b>
<b>Bibliography .....</b>	<b>133</b>

## LIST OF TABLES

Table	Page
1.1 List of datasets supported by FireFly .....	9
2.1 Pros and cons for (a) Silverlight (b) HTML5 (c) Flex [7] .....	23
5.1 Comparison of different technologies [28] .....	63
6.1 Flex data visualization table [18] .....	76
7.1 FireFly vs. Internet Explorer .....	83
9.1 Details of KW18 format .....	105

## LIST OF FIGURES

Figure	Page
1.1 FireFly login page .....	3
1.2 VIRAT .....	6
1.3 HeLA cells .....	6
1.4 Malaria cells .....	7
1.5 Vessels .....	7
1.6 X-rays .....	8
1.7 Wound healing .....	9
1.8 Viking interface [6] .....	11
1.9 Cell Profiler interface [4] .....	12
1.10 Ilastik interface [2] .....	14
1.11 ICY interface [23] .....	15
1.12 Virtual annotation tool [32] .....	16
1.13 Advanced virtual microscope .....	18



1.14 Kolam interface [24] .....	20
2.1 New FireFly architecture .....	21
2.2 Flex framework [8] .....	24
2.3 Cairngorm framework [29] .....	25
2.4 Local cache and screen buffer.....	27
3.1 Expansion of dataset level into image set and annotation .....	29
3.2 Multiple annotations on single image set.....	30
3.3 Old representation of user labs menu.....	31
3.4 Hierarchy and intuitive name of datasets.....	32
3.5 Soft links with the actual image names.....	34
3.6 Database table to store image information.....	35
4.1 Lung boundaries with edit points.....	37
4.2 Vessels with edit points .....	38
4.3 Tool panel .....	39
4.4 Segment represented by polyline showing the vertex to be inserted .....	40
4.5 Contour represented by polygon showing the vertex to be inserted.....	40
4.6 Segment after add operation .....	41
4.7 Contour after add operation .....	41
4.8 Contour with vertex to be deleted.....	43
4.9 Segment with vertex to be deleted.....	43
4.10 Segment after deletion of vertex .....	44
4.11 Contour after deletion of vertex .....	44
4.12 Splitting of a contour .....	45

4.13 Splitting of a segment .....	46
4.14 Contour after split operation .....	47
4.15 Segments after split operations .....	47
4.16 Merging of closed contour .....	48
4.17 Merged contours .....	50
4.18 Merging of two polygons with same orientation .....	51
4.19 Merging of polygons drawn in opposite directions .....	51
4.20 Merging in complex polygons .....	52
4.21 Merging of segments.....	53
4.22 Results after merge operation with join1 .....	54
4.23 Results after merge operation with join 2 .....	54
4.24 Results after merge operation with join 3 .....	54
4.25 Results after merge operation with join 4.....	55
4.26 Closed contours- case 1 .....	56
4.27 Closed contours- case 2 .....	57
4.28 Segments- case 1 .....	58
4.29 Segments- case 2.....	59
5.1 Vessels image.....	61
5.2 Architecture of CGI .....	63
5.3 Running MATLAB executable in FireFly.....	66
5.4 Segmentation results .....	67
5.5 Alpha blending.....	68
5.6 Image with processing result .....	69

5.7	Graph analysis.....	70
6.1	Google chart API [13].....	72
6.2	FLOT [14].....	73
6.3	Raphael [15].....	73
6.4	D3JS [16] .....	74
6.5	Flare [17].....	75
6.6	Vessels with curvature graph .....	77
6.7	Malaria image with malaria count column chart .....	78
6.8	Data grid with scatter plot.....	79
6.9	Bar graph and pie chart .....	80
6.10	Zoomable line chart .....	80
6.11	Zoomed portion of line chart .....	81
7.1	Annotation with R/W access.....	82
7.2	Alert box if another instance of same user is logged in.....	82
7.3	Website data settings.....	84
7.4	Lung X-ray with the mask .....	85
7.5	Malaria cell count .....	87
7.6	Bulk save.....	88
7.7	Incremental save .....	89
8.1	Coordinate system in Flex .....	90
8.2	Creeping of points in wound dataset.....	92
8.3	Old code for calculating and drawing.....	93
8.4	New code for calculating distance .....	93

8.5	New code for drawing.....	94
8.6	Circle drawing with edit points.....	95
8.7	Polygons used for marking lateral side of lungs.....	96
8.8	Polylines in vessels image .....	97
8.9	Curves .....	98
8.10	Free form.....	99
8.11	Save/Display panel- version updates .....	100
8.12	Frame advance panel- version updates .....	101
8.13	Tools panel version updates.....	102
9.1	GUI for writing KW 18 file .....	104
9.2	KW-18 file .....	105
9.3	Lung coordinates.....	106
9.4	Microvasculature file format.....	107
9.5	Data analytics format .....	108
9.6	ROI format file.....	109
9.7	GUI for downloading results .....	111
10.1	Support for mobile devices .....	117
B.1	FireFly login page .....	121
B.2	User lab menu.....	122
B.3	Workspace.....	123
B.4	Class chooser panel .....	124
B.5	Frame advance panel.....	125
B.6	Drawing tool panel.....	126

B.7 Different drawing shapes in FireFly.....	127
B.8 Save/Display panel.....	128
B.9 Data analytics panel .....	129
B.10 Help panel .....	130
B.11 Attribute window.....	131
B.12 Debug window .....	132
B.12 Info and message window .....	132

# WEB-BASED INTERACTIVE EDITING AND ANALYTICS FOR SUPERVISED SEGMENTATION OF BIOMEDICAL IMAGES

Rahul Kumar Singh

Dr. Kannappan Palaniappan, Thesis Supervisor

## ABSTRACT

Biomedical imaging and image analysis is a vital source of information for quantitative studies in life sciences and improving healthcare medical diagnostics. Various imaging algorithms have been developed to extract essential features and information that assists in this process. One critical aspect is assessing the quality of automatic image and video analysis algorithms. The accuracy of automatic algorithms is usually evaluated against ground truth, which is determined by manual annotations provided by multiple experts at different locations, to develop robust object detection, segmentation and classification algorithms. Another aspect of manual annotations is supervised segmentation i.e. finding boundaries of regions associated with objects of interest in images and videos. Therefore, a tool that can support a host of complex annotation creation and editing operations in a collaborative manner with an easy to use web interface for cloud-based editing, analysis and storage is a key requirement for which there are few scalable solutions available.

FireFly is a tool that was developed for manual and assisted expert annotation of images and videos for algorithm development and discovery. FireFly provides multiple domain experts at geographically different locations a collaborative tool for shared visualization and annotation that allows them to create, visualize and validate consensus ground truth and perform various image analysis tasks. FireFly is a web-based Rich Internet Application that is built on Adobe Flex, PHP and MySQL. In the context of big data, FireFly is used for managing large image collections, video sequences, collaborative ground truth generation, tracking and labeling for high-throughput studies and algorithm development. The primary objective of this project was to enhance FireFly's capabilities to allow interactive creation and editing of annotation objects like segments, contours

and automatic back-end analysis of Biomedical images by interfacing with MATLAB executables. Efficient algorithms were developed for local cache management of large collections of image frames each with possibly hundreds of annotated objects that have complex geometry like polylines and polygons, along with their associated labels and related attributes. The FireFly user interface was significantly enhanced to provide greater functionality and ease of use for the non-specialist. FireFly has been used for several Biomedical applications in collaboration with NIH including malaria cell counting, wound healing cell assay analysis, detecting the boundaries of lung regions in patient chest x-ray images and tuberculosis disease classification for computer aided diagnosis. FireFly has also been used for the morphological analysis of vessel structure in capillaries and microvasculature of dura mater using epi-fluorescence microscopy images and for bacteria cell tracking.

# Chapter 1

## Introduction

### 1.1 Brief Introduction

In the world of medical sciences, images have become a vital source of information utilized by biologists and doctors for diagnostics and research. Various imaging algorithms have been developed to extract essential features and information that assists in the decision process. The accuracy of these automatic algorithms has to be evaluated against ground truth since automatic analysis may not always be accurate. The images contain numerous objects and correcting each of them is time consuming. Several commercial tools which are used in medical image analysis are not quite as generic and only target a specific problem. FireFly has been designed with the sole intention to provide a unified platform for visualizing, editing, ground truth and data analysis.

In this thesis we will discuss several of our biomedical projects. One of the main project we were concerned involves detecting lung boundaries in Chest X-rays [3] for analysis. Detection of lung regions in chest x-ray images is an important component in computer-aided diagnosis (CAD) [3][5]. In certain diagnostic conditions, the relevant image-based information can be extracted directly from the lung boundaries without further analysis. For example, shape irregularity, size measurements and total lung volume, provide clues for serious diseases such as cardiomegaly, pneumothorax, pneumoconiosis or emphysema. CAD-based identification of lung disease based on accurate lung boundary segmentation plays an important role in the subsequent stages of automated diagnosis. With the contour editing and labelling support, FireFly proved to be a useful tool in this research. As FireFly fetches images via HTTP, a user can work remotely anywhere around the world and annotate the images by using different drawing tools on local machine. The annotations are stored in the database, which can be retrieved anywhere, anytime in future. This feature made it possible for us to work collaboratively on X-rays with researchers and radiologists based at University of Missouri, Columbia and National Institute of Health. Some additional



support and enhancements were made in order to support large lung boundaries which are described in more detail in Chapter 7.

Another project described in this thesis is analysis for Dura-mater vasculature network from Epi-fluorescence microscopy vessel images. Initially, doctors had to manually mark the Vessels in these images. Manual marking was a labor intensive task. With the help of CGI scripting and MATLAB executables we were able to run a robust smoothing based thresholding [19] segmentation scheme on these images remotely on the server. The resultant segmentation mask was processed to extract the network graphs which was later corrected by doctors and further analyzed in FireFly to study tortuosity, curvature and angles. MATLAB interfacing and segmentation editing is described in detail in Chapter 4 and Chapter 5.

Besides X-rays and Microvasculature, FireFly is being used for several other biomedical datasets like Malaria Cell Count, HeLa cells and Bacteria segmentation. FireFly is also being used for video surveillance tracking in UPS, VIRAT, and FPSS.

Next section gives a brief overview of FireFly Version 3.0.

## 1.2 FireFly Version 3.0

With the current advancement in web, we are capable of handling rich interactive applications that can be as powerful as desktop applications. Web applications have advantage of being extremely portable, reduces memory and time required to install heavy software on the client's machine. FireFly is a rich multimedia web-based tool based on Adobe Flex. The server side scripting is done using PHP and MySQL database is used to store annotations. In FireFly version 3, the system architecture was extended using PERL CGI. CGI scripting was used for running MATLAB executable through Common Gateway Interfacing.

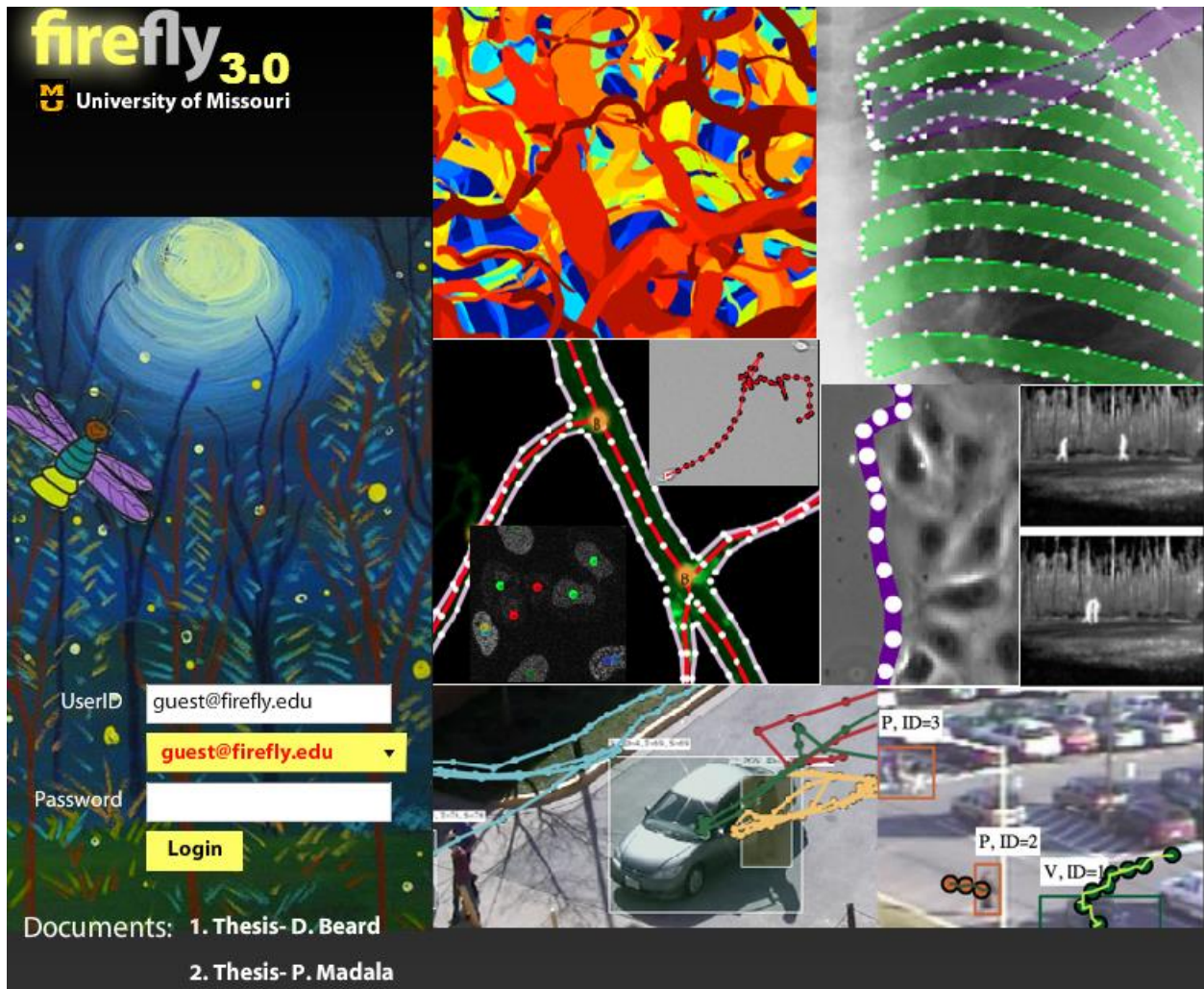


Figure 1.1 FireFly login page

Figure 1.1 shows the login page and FireFly interface. FireFly was developed at Computation Imaging and Visual Analysis lab at University of Missouri, Columbia as a general tool for ground truth and image analysis. The system architecture and technologies used are described in detail in Chapter 2. Major changes from Version 2.0 to 3.0 are listed below-

- Support for image collections
- Automating backend analysis of images by interfacing with MATLAB executables
- Contour and segment editing support

- Data visualization and analysis
- More intuitive GUI with new drawing tools
- Better local cache management and database update
- Better lock mechanism and multiuser support
- Support for multiple import/export data format

FireFly is used for 6 main purposes: Visualization, Classification, Tracking, Labelling, Data Analysis and Segmentation, It provides an interface to the researches and doctors where the object produced by algorithms can be mapped directly on to the image. It becomes easier for the user to correct and mark due to several drawing/editing options available. The data analysis feature provides users a quick way to analyze the changes in data. The primary features of FireFly are explained below-

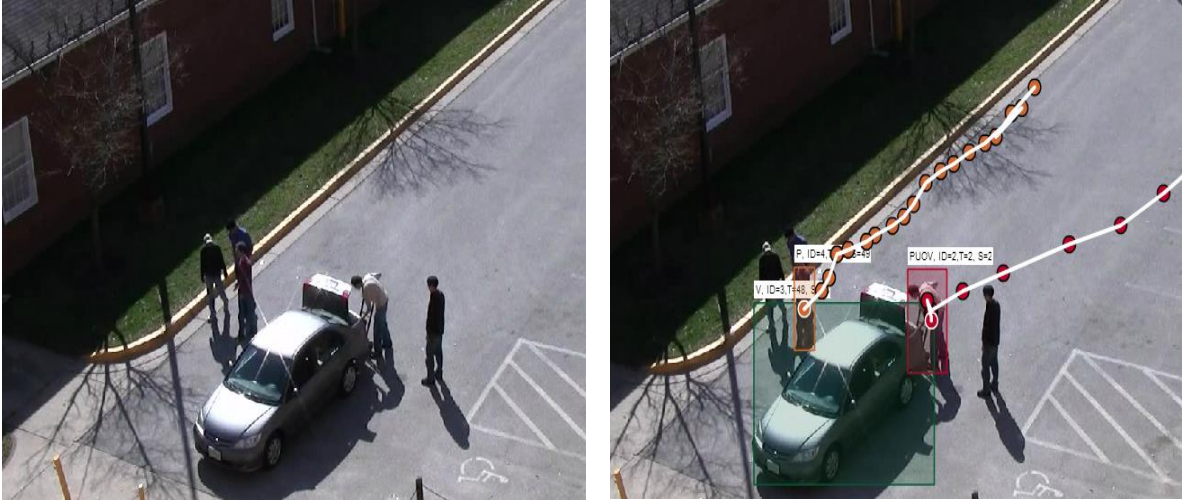
1. Visualization- FireFly maintains an image base from where it fetches the images over HTTP. It visualizes the annotations in the form of objects on top of the image, which gives users a better interface for editing. FireFly uses various graphical objects for annotations like lines, boxes, points, polylines, polygons, curve and freeform.
2. Classification- It is the process of separating objects into different categories. FireFly supports classification by maintaining a set of diverse classes. Users can easily mark objects with one class, which can be changed to any other class later. Each class is denoted with a specific color, which helps in distinguishing several objects with different classes on the screen.
3. Tracking- Tracking is used to study lineage, cellular events and migration. In general, various automated trackers are used to generate the tracks. FireFly provides an interactive interface with different join and split modes for track editing, which enables the user to correct the tracks. These tracks can be edited interactively by using several track editing operations. Besides single object manual interactive tracking, FireFly also supports track split and merge operations. Track editing is described in more detail in [21].

4. Labelling-Image labelling help us to mark and store important information corresponding to not only images but also segmented objects on the screen. Examples of labels are diseases in a chest X-ray or properties of a cell in a microscopic image. This feature of marking and saving information allows user to store some extra information besides just the coordinates or ground truth which can be utilized for further analysis.
5. Data Analysis- Every image contains large amount of data associated, which can be utilized for further analysis. This data is very useful for the users as it can be used to compute various other parameters and features. FireFly provides a data analytics panel, which can be used for the computation and analysis. The parameters and features produced after analysis can be visualized by using the Data Visualization panel.
6. Segmentation- It is the process of extracting or partitioning an image into similar regions. These segments or contours are the pixels which share similar characteristics. FireFly provides a rich interface to visualize these segments. Segments are represented by polygons and polylines. These segments can be edited with different editing options such as adding or deleting of vertices, splitting and merging of polylines or polygons.

### 1.3 FireFly Datasets

FireFly is a generic tool that supports a vast variety of datasets. Each dataset uses some of the functionalities mentioned above. The datasets are divided into two labs- Biomedical and surveillance. Examples of datasets with the primary functionalities used are described below-

1. VIRAT- Fig 1.2 shows one of the primary functionalities of FireFly- tracking. Figure 1.2(a) is an image with no ground truth and Figure 1.2(b) shows the 2 tracks- orange track representing 'walking person' class and the red track represents 'person unloading from car' class. The red and orange circles specify the position of these objects in different frames. This connection is maintained in the database and is retrieved as per the requirement.

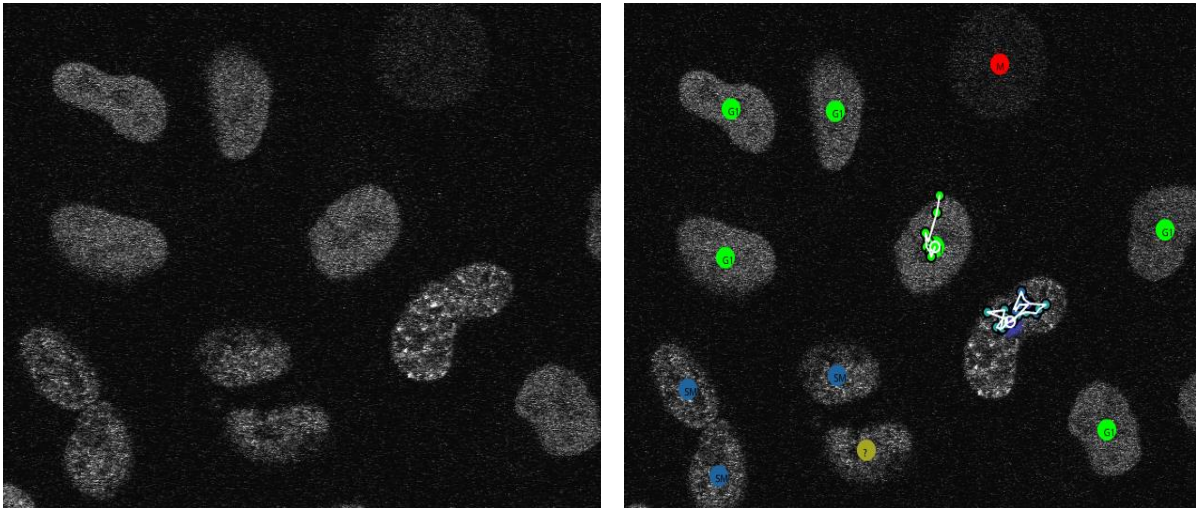


(a) Without ground truth and track

(b) With ground truth and track

Figure 1.2 VIRAT

2. HeLa Cells- Figure 1.3 shows HeLa cells. These cells have been classified into Mitosis, G1, S-Early, S-medium, S-late, G2, Apoptosis, undecidable and other class. These cells are classified based on the time of formation and each cell contains the lineage information that is used to track the cell and its origination.

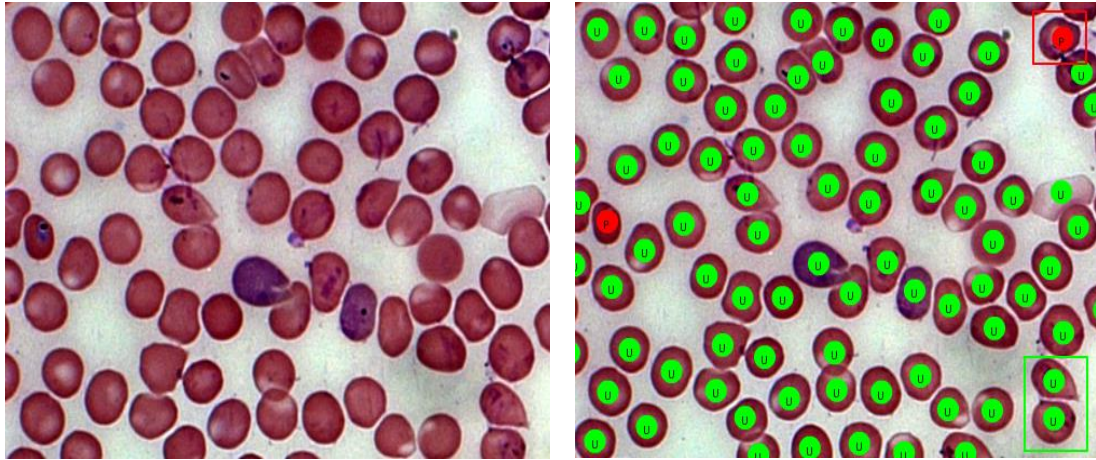


(a) Without tracks

(b) With tracks

Figure 1.3 HeLa cells

3. Malaria Cells- Fig 1.4 shows Malaria cells. The labels have been classified into- parasitemic, uninfected and others. Classification is used to count the number of infected cells. Multiple colors are used to distinguish between different classes on the screen.

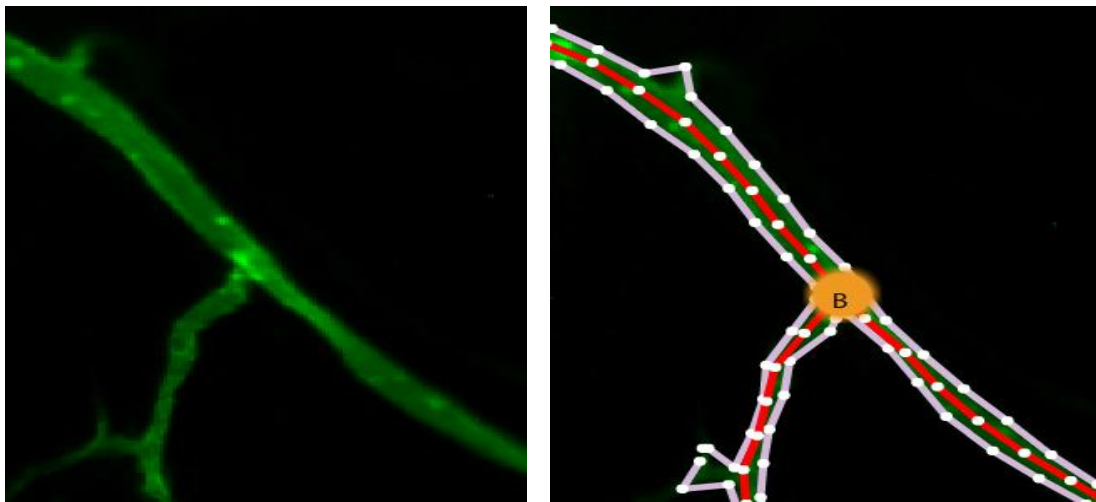


(a) Without ground truth

(b) With ground truth

Figure 1. 4 Malaria cells

4. Vessels- Figure 1.5 shows vessel images taken from mice. Figure 1.5(a) contains no annotations and Figure 1.5(b) contains the network extracted after segmentation. The mice were treated over the weeks and then the changes were analysed.



(a) Without segmentation

(b) With segmentation

Figure 1.5 Vessels

In Figure 1.5(b) Medial Axis, Boundaries and Branch points are shown which are computed by automatic segmentation. For analysis the curvature, angle and tortuosity is computed from this graph and visualised in the form of column charts.

5. Lung X-rays- Figure 1.6(a) shows lung X-rays without any ground truth and Figure 1.6(b) shows a lung X-ray with masks. Many diseases, including TB, can be discovered by segmenting out the correct portion of the lungs. FireFly was enhanced to support the large X-ray boundaries. With the help of segmentation editing, the lung boundaries can be efficiently marked in lesser time.

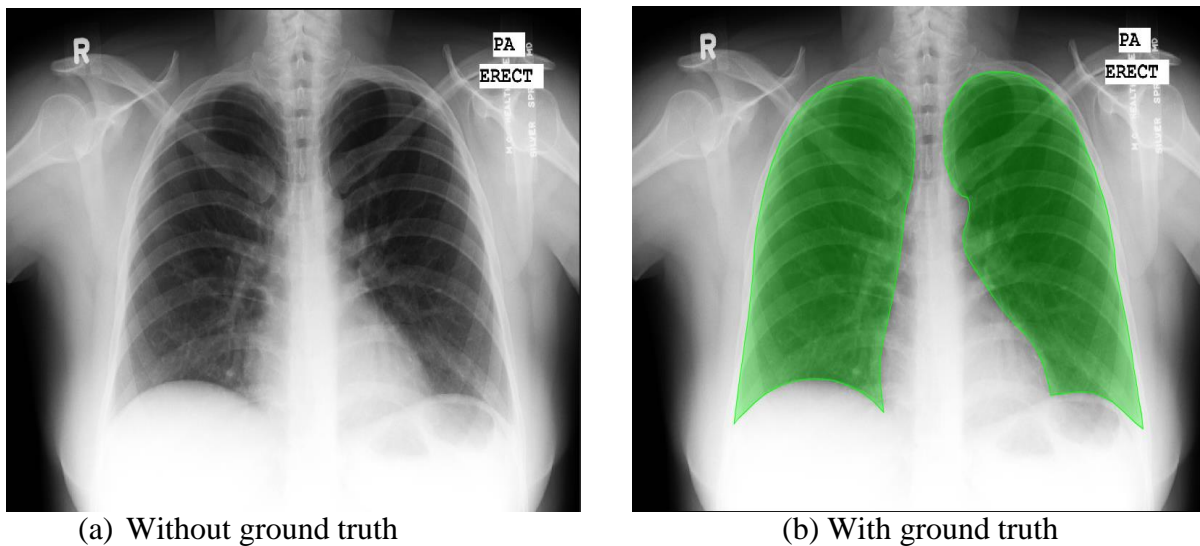
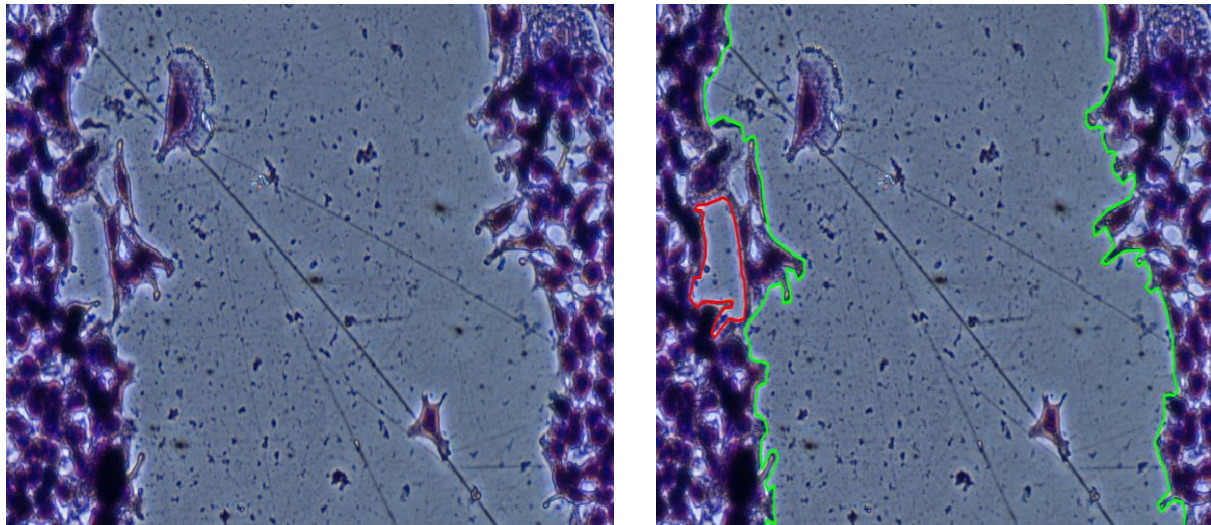


Figure 1.6 X-rays

6. Wound Healing – Figure 1.7(a) shows wound healing datasets without any annotations and Figure 1.7(b) shows wound healing with annotations. With these images we mark the wound boundaries and monolayer voids which are used for tracking and studying dynamics of cells to understand the wound healing process.



(a) Without annotations

(b) With annotations

Figure 1.7 Wound healing

DATASETS	OBJECTS
Surveillance Video Labeling (UPS, VIRAT, FPSS)	Points, Boxes, Lines, Polylines and Polygons, Circles
Malaria Cell Count	Points, Box
Vessel Segmentation	Polylines, Points, Boxes
Lung X-ray Segmentation	Polygons
Lung X-ray Labeling	Boxes, Curves, Circles, Lines, Points, Polygons, Polylines
Bacteria Labeling	Polylines, Polygons
Bacteria Segmentation	Polylines, Polygons
Cell Labeling and Tracking	Points

Table 1.1 List of datasets supported by FireFly



## 1.4 Other Similar Tools

In order to approach the solution for the above mentioned problem a study of existing annotation tools is necessary. There are few annotation tools already in existence. In this section we will be analyzing the supported features and the drawbacks associated with these tools. Some of the tools have been described earlier in D. Beard's thesis [20] and P. Madala's thesis [21] - LabelMe, LabelMe video, Kolam, ViPER, Allen Brain Atlas, Omero.web, Bisque, NeuronJ and DCellIQ.

Besides the above mentioned tools here we will be discussing more tools like Viking, Cell Profiler, Ilastik, ICY, VATIC and AVM in detail. Also some enhancements with respect to ViPer and Kolam will be discussed later in this section.

### 1.4.1 Viking

Viking [6] was developed by James Anderson at the Marc lab, University of Utah. It is a web-based tool that supports multiple users and is a collaborative management system for images and data. Viking supports multi-terabyte datasets. It was developed for use with the first retinal connectome, which was assembled using serial section Transmission Electron Microscopy and Computation Molecular Phenotyping. Viking performs real-time transformation of the original tiles using the fast texture mapping abilities of graphical processing units. It is based on Microsoft C#, SQL server Express 2008, XNA, MATLAB 2009b, Python 2.6, NCR toolset. 2D/3D graphics plot is done by using Viking plot, which is a separate web page on the website. It supports manual track editing and was mainly developed for tracking R1 cells.

Viking is a web based tool that works through HTTP. With its current interface several new modules can be added, without making any changes to the interface. It uses a multi-tiered architecture consisting of viewer, image tiling server, WSDL server and the database. A slice of an image from the volume can be viewed at any time using the viewer. It fetches the image tiles over HTTP and gives the user real time feedback by applying transformations in the backend. Viking provides a good interface to map these tiles at appropriate positions on the screen. It uses Microsoft SQL Express 2008 to store annotations. The service exposes various methods for updating the objects in the database. Even though Viking is a web-based application, it still

requires a dedicated graphics card, .NET framework, XNA 4.0 on the system. Since, it is not browser based, it requires Viking Viewer to be installed on the system before its use. Viking lacks any kind of segmentation support.

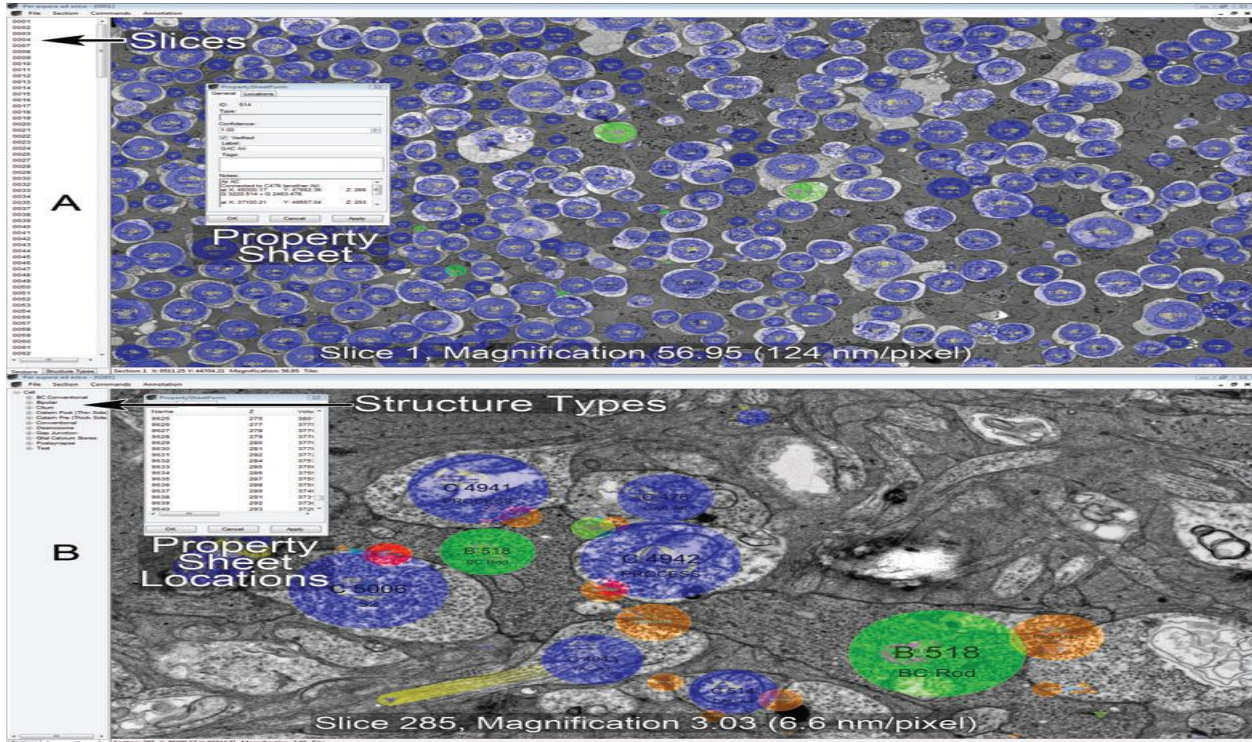


Figure 1.8 Viking interface [6]

## 1.4.2 Cell Profiler

Cell Profiler [4] was developed by Anne E. Carpenter and Thouis Jones at laboratories of David M. Sabatini and Polina Golland at the Whitehead Institute for Biomedical Research and MIT's CSAIL. The source code was originally written in MATLAB. Computationally intensive tasks used MATLAB's native compiled functions. Cell Profiler was later rewritten in Python. It was designed to help biologists measure phenotypes from a number of images without any knowledge of image processing. Cell Profiler also contains advanced segmentation algorithms. Since it is free source software, different modules can be added easily. Algorithms are kept together to form a pipeline, which is used for detecting objects and features. Cell Profiler consists of two different parts.

The first part consists of image processing and production of numerical data. This numerical data produced can be exported to Excel sheets or can be stored in the database. The second part consists of analysis, which can be utilized by researchers and biologist to examine the data produced. Initially, this part was called Cell Visualizer software project, later its name was changed to Cell Profiler Analyst. Cell Profiler gives the freedom to develop and add to its free source codebase. The user doesn't have to worry about GUI, compilation and cross platform execution and can concentrate on developing its own algorithm.

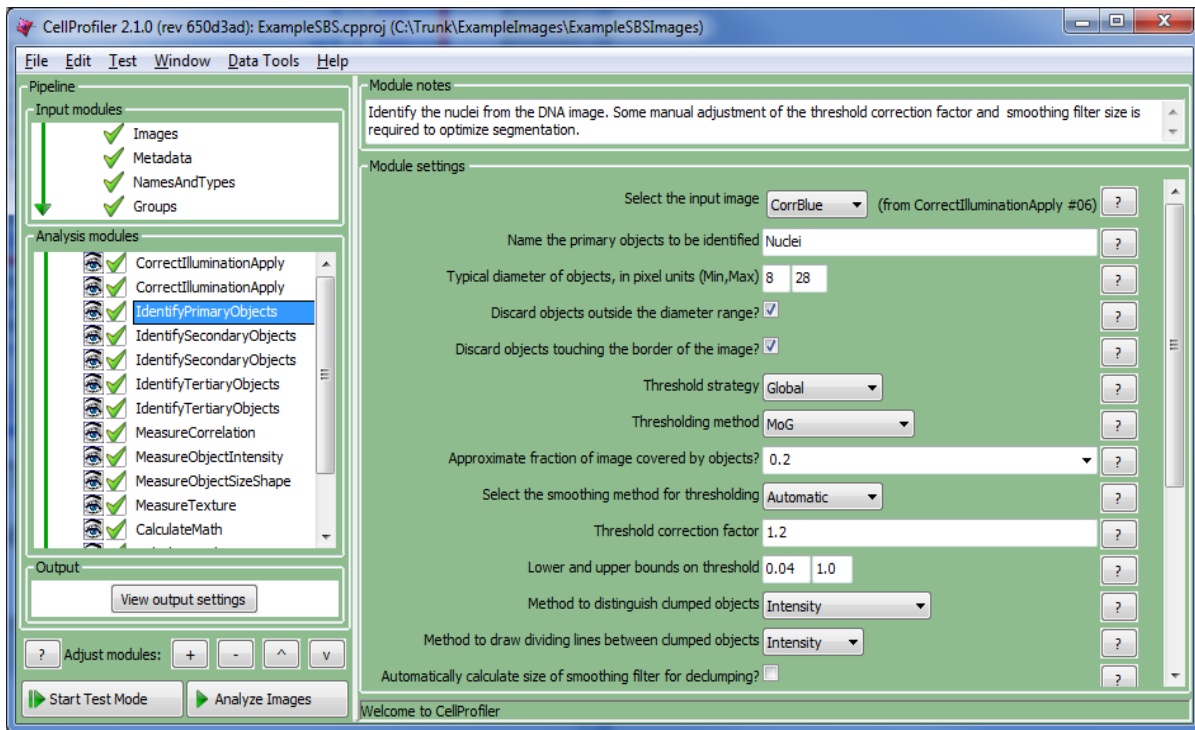


Figure 1.9 Cell Profiler interface [4]

The algorithms can be implemented using Scipy and Numpy. Implementation of algorithms can also be done by integrating C and Java code with Python or directly using Cython, The plots can be generated by the Data-Tools menu. Some advanced algorithms such as advanced image alignment are not present in Cell Profiler but the algorithm plugin can be called from ImageJ. This feature provides Cell Profiler an advantage over other tools as it can interface with tools like ImageJ easily. Cell Profiler is a desktop based application requiring a prior installation and setup. It supports classification, tracking and segmentation. But it doesn't provide any kind of editing of tracks or segments.

### 1.4.3 Ilastik

The scientists at the Heidelberg Laboratory for Image Processing (HCI), University of Heidelberg first released Interactive Learning and Segmentation Toolkit in 2011. Ilastik [2] is a simple tool for image classification and segmentation. It uses mouse interaction for labelling the classes. The labels are then used to run a Random Forest classifier. Ilastik allows a real time feedback and it trains algorithm accordingly, which helps in faster labelling. Wrong clicks can direct the user to wrong classification. After the classifier has been trained, it can be exported and be used on a larger amount of data.

Ilastik is easy to use and provides automated workflows for various features. It supports semi-computerized and manual tracking and segmentation, object counting and classification. The process evaluation simultaneously occurs offline without slowing down the operation. The pixels are classified using annotations marked by the user, which are further used as segments. This classification works fine for distinct objects. However, if color and brightness are too similar the pixels can't be distinguished. Ilastik provides two different workflows- manual and semi-computerized. The manual workflow can be used for ground truth analysis. The main file format used by Ilastik is HDF5. The file format bmp, gif, jpg, tif, ras, png, ppm, pnm, hdc, xv, npy also can be imported directly into a project. Using Ilastik requires no experience in image processing. Ilastik is a desktop application that is built on QT and C++. Its exe file is available for WIN, Mac and Linux OS. It supports segmentation with Watershed algorithm and classification with random forest classifier. Even though Ilastik supports pixel and object classification, tracking, carving, density counting, headless operations, it lacks in providing the segmentation and track editing feature. Being a desktop application means it requires installation. It also requires larger storage space as the images have to be stored on the local machine. Ilastik works well for small set of images but for larger set it requires high storage unit and better processing power.

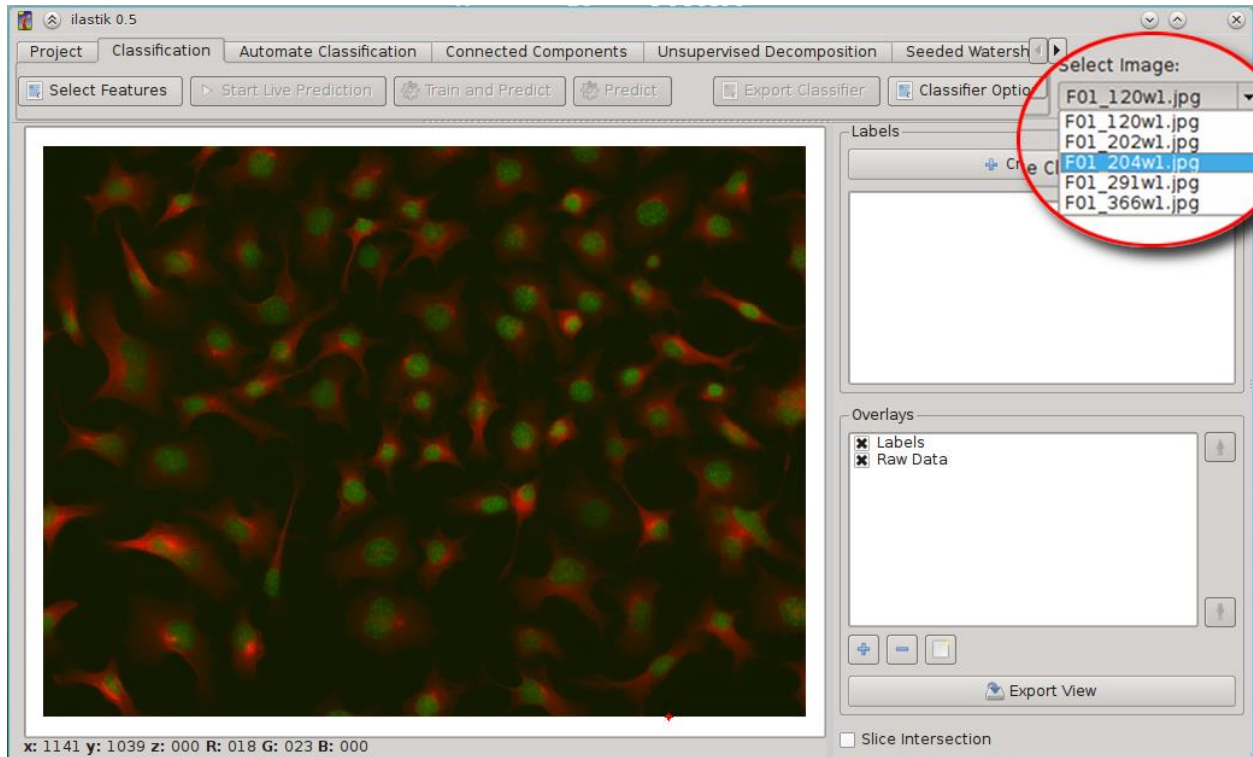


Figure 1.10 Ilastik interface [2]

#### 1.4.4 ICY

ICY is a GPL open source software developed at Quantitative Image Analysis Unit at Institut Pasteur, France. It is a collaborative framework for mathematicians and biologists. Mathematicians can develop algorithms as plugins, which can be utilized by biologists for image analysis. Icy is an open source Java based desktop application.

ICY [22] has plugins for supporting its different functionalities. It uses BioFormates for loading and saving of XML files. It uses Flamingo for interface look and VTK for 3D rendering. The results, manually marked ROIs and other parameters are stored in the form of XML. When any plugin starts, these files are bundled together with images and loaded into memory. This data is shared across different plugins in the XML format. <http://icy.bioimageanalysis.org> works as a central repository for plugins. Users can write their own plugins, test on their local machines and it can be later uploaded in the repository. The biggest advantage for Icy is this scalable architecture which provides people with APIs to interface easily. Due to this feature, a number of plugins are

available. Currently ICY supports almost all formats of images available. Even though it supports all kinds of data but certain conversion has to be done before display, as all LUT (look up table) formats are not compatible with computer.

ICY has great capability of visualizing all kinds of data; it has diverse set of tools and the support for plugins makes it a good solution for the users. The only drawback is that it is a desktop application which means, one must have Java Runtime Environment (JRE) setup. For plugin development eclipse setup with jdk 1.6 is required. Another drawback is that the image files must be present on the system, which results in heavy memory usage.

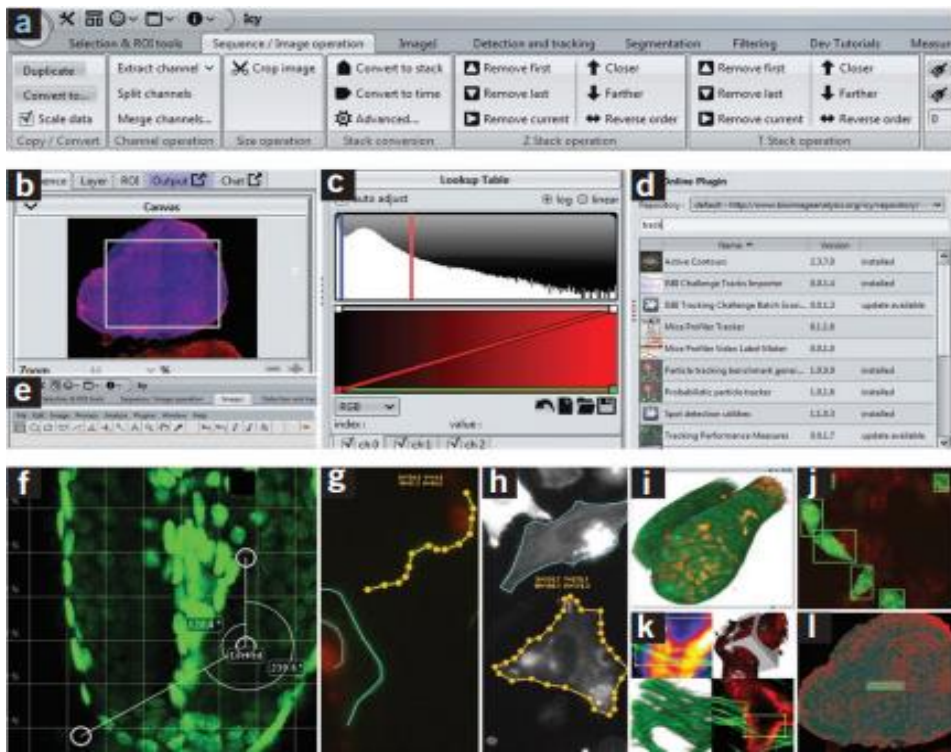


Figure 1.11 ICY interface [23]

#### 1.4.5 Video Annotation Tool

Video Annotation Tool from Irvine, California (VATIC) [32] is an online interactive tool for annotating different kinds of datasets. This tool was developed at University of California, Irvine for computer vision research. The annotations work crowd sourced to Amazon's Mechanical Turk. This tool has been written in python and can be deployed over cloud.

The users can annotate while the video is being played. The bounding boxes can be used to mark objects. The speed of the video can be adjusted by the user anytime. Each object can be marked by using a different attribute. These attributes describes some action taking place. The action can be anything ranging from Person walking to Person opening a door. The objects can also be marked occluded, if it is off the screen. Some keyboard shortcuts available assists in marking of annotations. Manually marking each and every frame is tedious and time consuming, so an interpolation is applied to place the boxes in each frame. This box can be changed at any time by the user. The user has to mark only T number of frames and rest of the frame can be interpolated offline. The videos like car may require less time while videos like basketball requires more time, so the frame rate can be decided by the user depending upon the dataset. This system has been tested only on Ubuntu with Apache 2.2 HTTP and MySQL server.

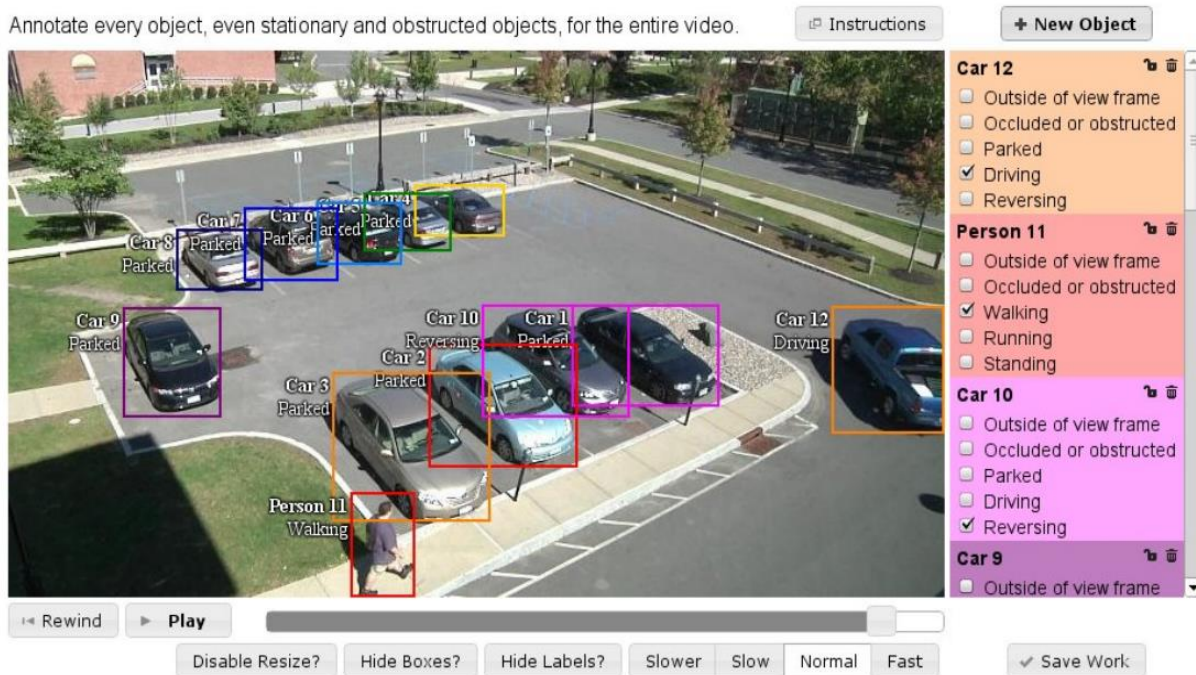


Figure 1.12 Virtual annotation tool [32]

The tool setup can be downloaded from their website. Even though the tool is online but it needs to be installed and database needs to setup in order to be used. For annotation it only uses bounded box. There is no support for segmentation or track editing. The analysis can be done but it occurs outside the tool on the results that are generated from the tracking.

#### 1.4.6 Advanced Virtual Microscope

Advanced Virtual Microscope (AVM) [31] is an image annotation tool developed at Applied Visions Laboratory, Texas Tech University. AVM was designed to work with multi-gigapixel virtual histology [30] slides. The images can be uploaded as slides and are stored remotely on the servers. It consists of 3 parts: Java Web service, a client application and a NoSQL database. Currently the application is hosted on Texas Tech server. The web services runs on Glass Fish server and it provides CRUD interface for the client and database. With the help of Feature generators, test classifiers can be trained and the values can be visualized to show different image sections. The client application helps to interact rapidly with very large images. The images are stored on the server in the form of small individual tiles of different resolutions. These tiles are fetched and displayed depending on the sub region. JPEG compression quality has to be selected while uploading the images before converting into tiles or it would be converted to a compatible format depending on user's machine. If they are already in compatible tiff format, the images are directly uploaded without any conversions

AVM contains a study manager for viewing and editing studies on the right side of the viewer. Studies are combination of slides and annotations. By using the study manager various operations on these studies can be performed. Solutions are stored on server which is displayed at the center of the main window. These solution consists of annotations with class labels, features, generators and classifiers. Currently it only supports ImageJ for generating features and LIBSVM for classification.

Any slide can be selected and displayed in the viewer by double clicking on the name. There are various options available on the viewer for annotating the ROI like freehand, oval, rectangular and polygon. It does provide some good editing features like point push and select points. With the help of point push tool the ROI points can be pushed to an appropriate place but it only works when ROIs are selected in ROI manager whereas with the select point's option individual points can be moved anytime. This tool doesn't support any kind of polygon editing. It also doesn't support any polyline or curve drawing tools. AVM is not browser based, the client application needs to be downloaded and installed on the user's machine, which requires some extra effort during initial use.



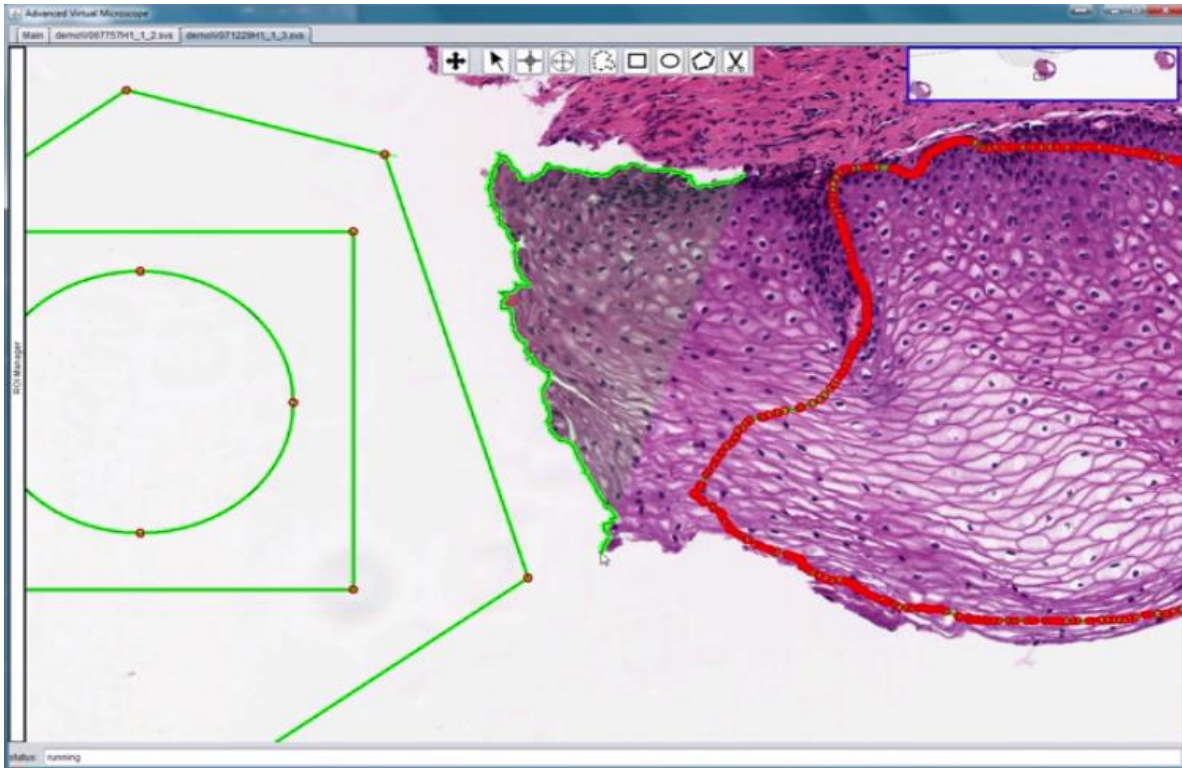


Figure 1.13 Advanced virtual microscope

#### 1.4.7 ViPER (Video Performance Evaluation Resource)

ViPER was developed at the Language and Media Processing Lab- University of Maryland. It is a video-analyzing tool [12] designed to allow frame-by-frame markup of video metadata stored in the ViPER format. As discussed [21] ViPER Ground Truth provides an interface for tracking people and detecting text. It uses shapes like rectangle, ellipse, point, etc. to annotate the ground truth. ViPER has two new additional tools: ViPER-Performance Evaluation for comparing the results of analysis from ground truth and ViPER- Viz\_ which is a set of UNIX scripts that can be used to compactly analyze ground truth and results of video clips.

ViPER was developed in Java and is a desktop based application. The data is stored in XML format, which is read by ViPER for data analysis. It also provides an API through which interfacing can be done. As ViPER is a desktop application, it requires installation on the client's machine and a lot of memory to store the images.

#### 1.4.8 Kolam

Kolam [1] was developed at the Computational Imaging and Visual Analysis Lab, University of Missouri, Columbia. It is an open source software built on QT API and UI framework. Kolam is used for visualizing and tracking in wide-area imagery and it supports manual and automatic tracking by interfacing with MATLAB executables. The user can also interface by writing these plugins in languages like C, C++ and Python.

Kolam provides an interactive visualization system and supports very large gig pixel per frame video. It provides a faster approach to monitoring, analyzing and evaluation of algorithms. It supports a dual cache for visualization of big data. Kolam supports datasets from gigabytes to petabytes in size. It uses its own tiling mechanism to fetch and display tiles. The images are converted to tiles by using quad tree regular tiling process. Kolam also supports interactive color map and histogram enhancements. Its biomedical application mainly comprises high-resolution, high throughput image visualization for microscopy imagery. With the capability of manipulating images and ability to interface with user-written plugins many image processing analysis algorithms can be performed. Kolam also provides users with a multi monitor display which helps in visualizing big data and performs visual analysis. It has been successfully for cell tracking, lineage and analysis. Kolam is a desktop application that not only requires local software installation but the images also have to be present locally on the system in order to be visualized.

Although, it supports Big data visualization and provides a rich interface for editing and tracking different objects across multiple frames, but the biggest drawback is that the large gig pixel images have to be present in the local memory. Storing images and performing analysis tasks consumes CPU memory and also demands extra storage space.

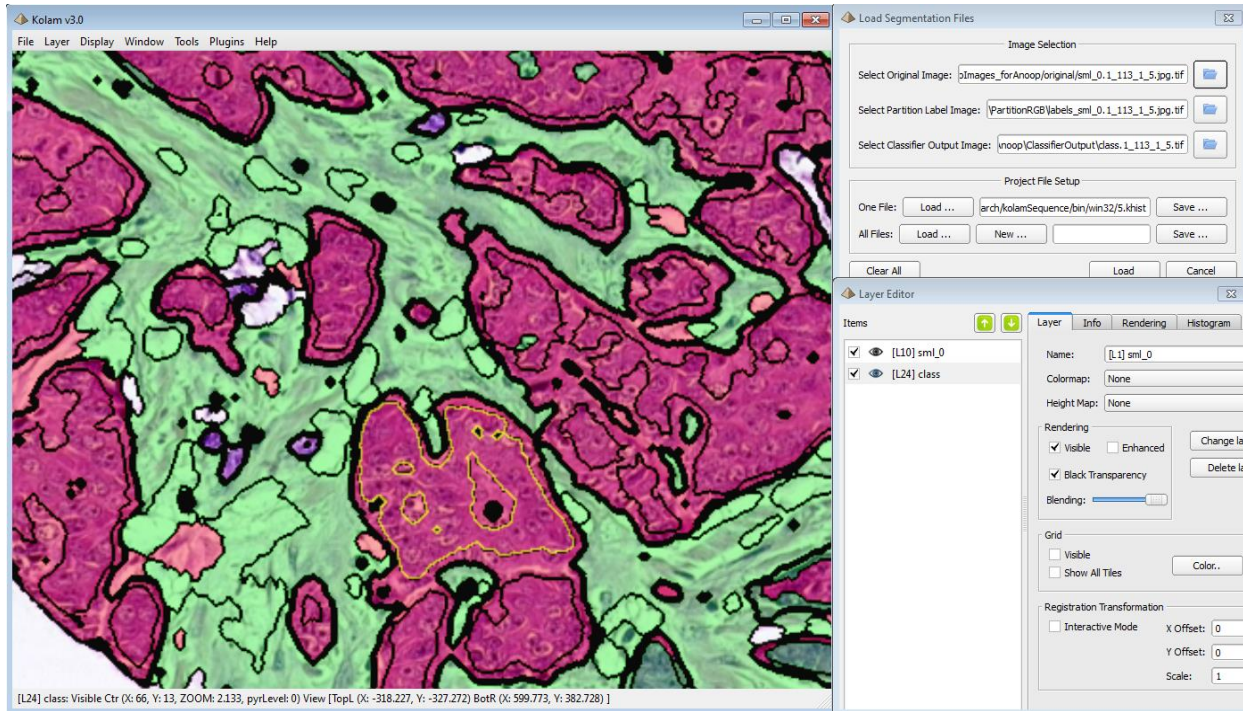


Figure 1.14 Kolam interface [24]

As we can see the major drawback with most of the existing tools is that they are desktop based applications and too specific to certain types of data. Most of the tools require installation and high memory as the images have to be present on the system. FireFly on the other hand is web based tool. The images are stored on the server and fetched over HTTP on the client for visualization. Due to this feature, FireFly is accessible anywhere remotely. As the algorithm run on high power CPUs i.e. on the server, it gives freedom to the clients to use lightweight CPUs as no processing is required on the client machine. This feature makes the processing faster with no setup time on the client system. FireFly's scalable architecture makes it easier for development and addition of new modules without disturbing the existing modules.

# Chapter 2

## FireFly Architecture and Technologies

### 2.1 New System Architecture

As shown in Figure 2.1, FireFly is based on client/server technology. Firefly is a Rich Internet Application with its client side written in Adobe Flex. The Flex code is compiled to produce SWF (Shock Wave Flash) file which is a file containing byte code. This file is transmitted over the web to the client side. On the client side Action Script virtual machine runs this file in the browser using Adobe Flash Player.

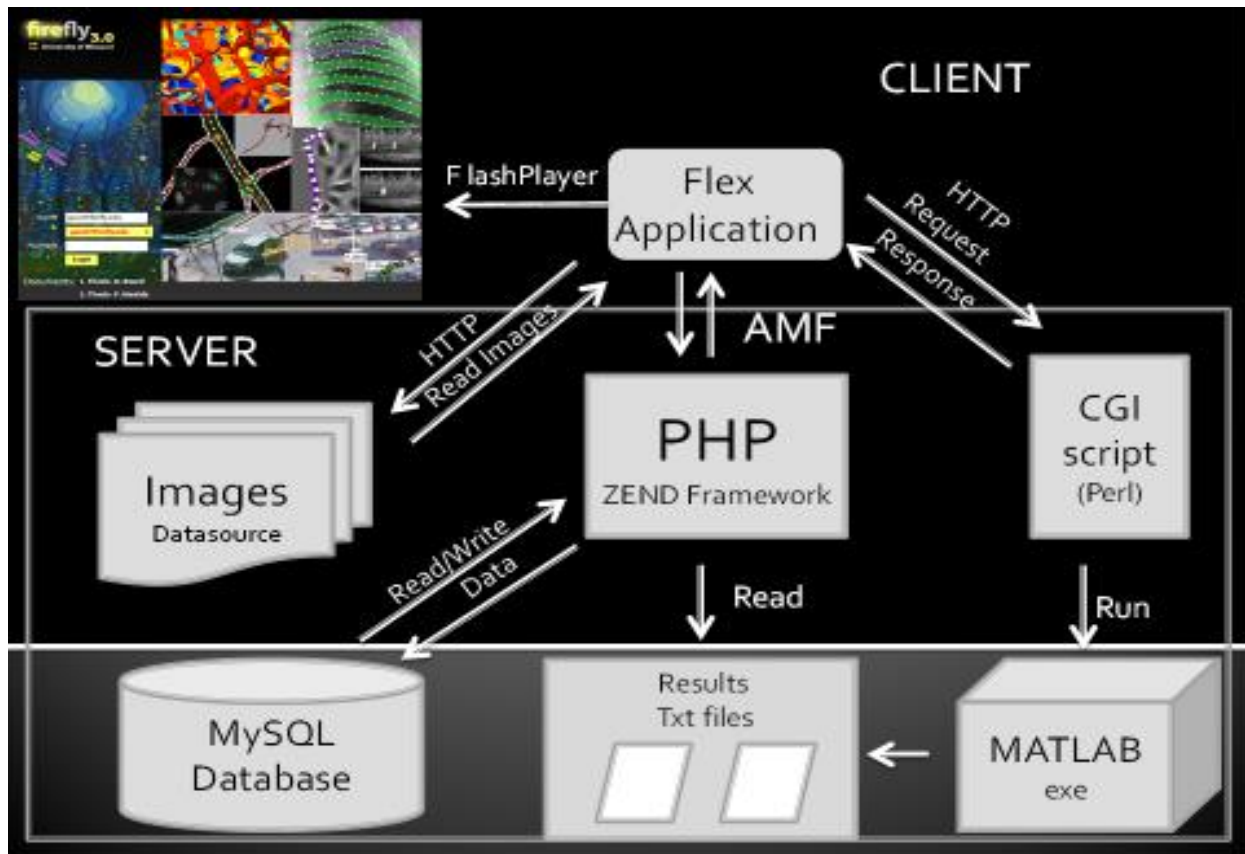


Figure 2.1 New FireFly architecture

FireFly is a heavy client based application with most of the interactions and data changes happening on the client side. When there is change, a request for update is sent to the server using AMF (Asynchronous Message Format) which further updates the database. The services are maintained on the Apache server in PHP. Zend framework has been used to provide MVC (Model View Controller) architecture on the server side. The database used for storage is MySQL database. The images are fetched through HTTP and displayed in FireFly, from the image base maintained on the server. For providing a more real time feedback, the client side also maintains a local cache. This cache is continuously updated and stores current, previous two and next two images in the sequence.

In FireFly v3, the architecture was extended and an additional module was added: MATLAB executable- PERL CGI interfacing. As shown in the Figure 2.1, a HTTP request is sent to invoke the PERL CGI script. The CGI script is placed in the CGI-bin folder. When an HTTP request is made to the CGI script with the required parameters the shell script is invoked. The shell script is used to execute the MATLAB executables. Once the backend processing is complete, a response is sent back to the client. The client sends another request to invoke PHP controller to load the results. This result is visualized in FireFly as different objects on the screen.

## 2.2 Technologies Used

FireFly is a rich and heavy client based application. It utilizes Flex for all the client side programming. There are multiple other client side technologies available that could have been used. Table 2.1 provides a comparison of 3 main technologies- Flex, Silverlight and HTML5. This section later describes why Flex technology was more beneficial for us as compared to other technologies:

1. Silverlight-

PROS	CONS
.NET framework / Visual Studio	Current adoption
Developer availability	Interaction with HTML requires JavaScript
Multithreaded	Maturity / longevity
Powerful styling	Framework complexity

(a)

2. HTML5-

PROS	CONS
No plugin = lightweight	Features not present in old browsers
Future proof	JavaScript language
Maximum reach (browser / OS / platform)	Lack of maturity
Multithreaded	Skills availability
CSS / HTML are designer friendly	Developer tools not as advanced as Flex and Silverlight

(b)

3. Flex-

PROS	CONS
Maturity / ubiquity	Single threaded
Predictable runtime	Heavyweight – Flex libraries required
An Object Oriented language and familiar tools for Java developers	Interaction with HTML requires JavaScript
Data services for Java	Skills availability

(c)

Table 2.1 Pros and cons for (a) Silverlight (b) HTML5 (c) Flex [7]

### 2.2.1 Apache Flex (Formerly Adobe Flex)

Flex was developed by Macromedia, which was later acquired by Adobe. After the acquisition, Adobe terminated Flex support and Flex was donated to the Apache Foundation. Flex is a cross-platform development language. It is based on Adobe Flash and requires Adobe flash player to run on the browser. Flex has a separate data access layer and presentation layer, this separate visualization totally from services. Flex by core is still flash. Due to the requirement of a web based language that can be used to create animations similar to the ones created in Flash, Flex was developed. The animations in flash are maintained by the user by maintaining a specific frame rate. Each object is drawn in a frame and then the animation is played. By default, Flex supports 24 frames per second, which is used for animating objects on the screen.

Flex provides various classes and features that can be used to develop a Rich Internet Application. One of the features is an XML based user interface language called MXML. MXML makes it easier to layout the components on the interface.

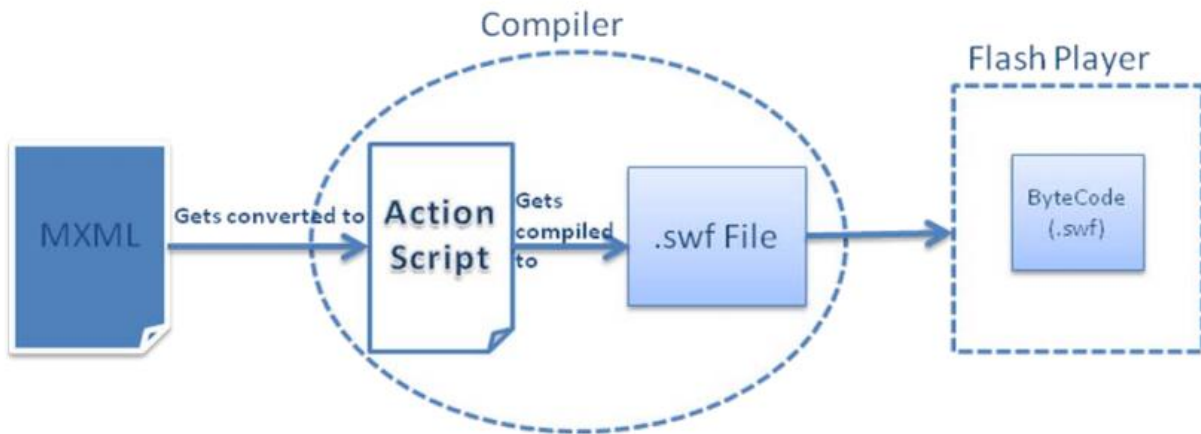


Figure 2.2 Flex framework [8]

MXML gets converted to Action Script and Action Script is compiled to SWF file, which is a file with byte codes. The SWF file is read by AVM (Action script Virtual Machine) in Flash player the same way as Java byte codes are read by JVM.

### 2.2.2 Cairngorm and Swiz Framework

To maintain the definite structure of an application, application frameworks are used. One basic principle is to separate the application logic, data access and user view. Other usage of developing an application with a framework is its reusability. The developers can easily utilize the already implemented classes and libraries.

FireFly uses Cairngorm framework, which is based on the Model View Controller model.

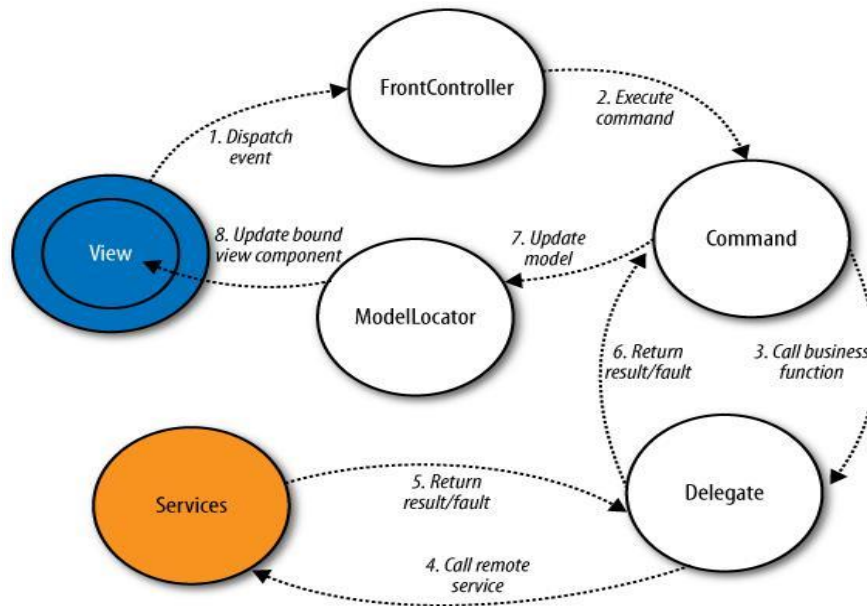


Figure 2.3 Cairngorm framework [29]

One of the most important things to understand about Cairngorm is that everything should be mapped to an event [10]. Figure 2.3 illustrates a typical event flow-

1. View- this is where the user interacts with components.
2. Dispatch Event- An event is dispatched after any interactive operations from the user
3. Front Controller – It receives the underlying logic and executes the command.
4. Command- some business logic is executed by the Command.
5. Model – The data that is altered by the logic in application.
6. View – The view changes as the logic alters the data since the data is tied in the view.



### 2.2.3 Swiz Framework

Swiz framework is used to simplify the operations and event handling in the application [11]. Swiz provides the framework to decouple the application code. It provides a simple life cycle for asynchronous remote method invocation. Swiz provides two main tags- Autowire and Mediate. While autowire tags are used to represent the dependency of an object, mediate tags are responsible for handling the events, anywhere in the application. Swiz is an inversion of a control framework that uses the idea of dependency injection. Dependency injection is a software practice, which allows us to remove hard-coded dependencies and allow it to change either at compile time or run time.

### 2.2.4 Server Interaction

Server interaction can be divided into three main phases [10]:

1. Execution Phase -When a command is dispatched, it calls the delegates. The Delegate uses the Service Locator to get the required service and calls the specified method on the server.
2. Application Tier Processing Phase- It runs on the backend server. It executes the heavy business logic on the server and sends back the data. To enable communication between the service and Flex we use AMF streaming.
3. Response Phase- The delegate receives the data back, which then passes it to the responder. The responder changes the model and since the model is tied to the view, the view is updated simultaneously.

FireFly uses Zend framework in the backend for writing services. Zend is also an MVC framework. FireFly has a dual MVC on server and the client side. FireFly uses AMF (Action Script Message) streaming to interact with services. This proves to be faster than HTTP GET and POST calls. The objects are serialized while sending to the server or client. When an object is sent from the server, Flash player can automatically serialize the object into the corresponding Action Script class. This is a great feature and helps in exchange of high volume of data.

### 2.2.5 CGI Scripting with PERL

Common Gateway Interface (CGI) is a method to generate web content by executable files. It is mostly written in a scripting language. Practical Extraction and Report Language (PERL), which was developed by Larry Wall, was used in FireFly to run the MATLAB executable. The PERL script invokes a shell script, which runs the executables. The data is interchanged via simple text files. This mechanism is described in more detail in Chapter 5.

### 2.3 Local Cache and Screen Buffer

FireFly maintains a local cache along with the screen buffer. Local Cache maintains frames which have already been fetched from the server. Along with the current frame it buffers 4 extra frames (2 previous and 2 next). This buffering of image helps the user to have a real time feeling while browsing through the images. The local cache maintains a set of markedObjects fetched directly from the database.

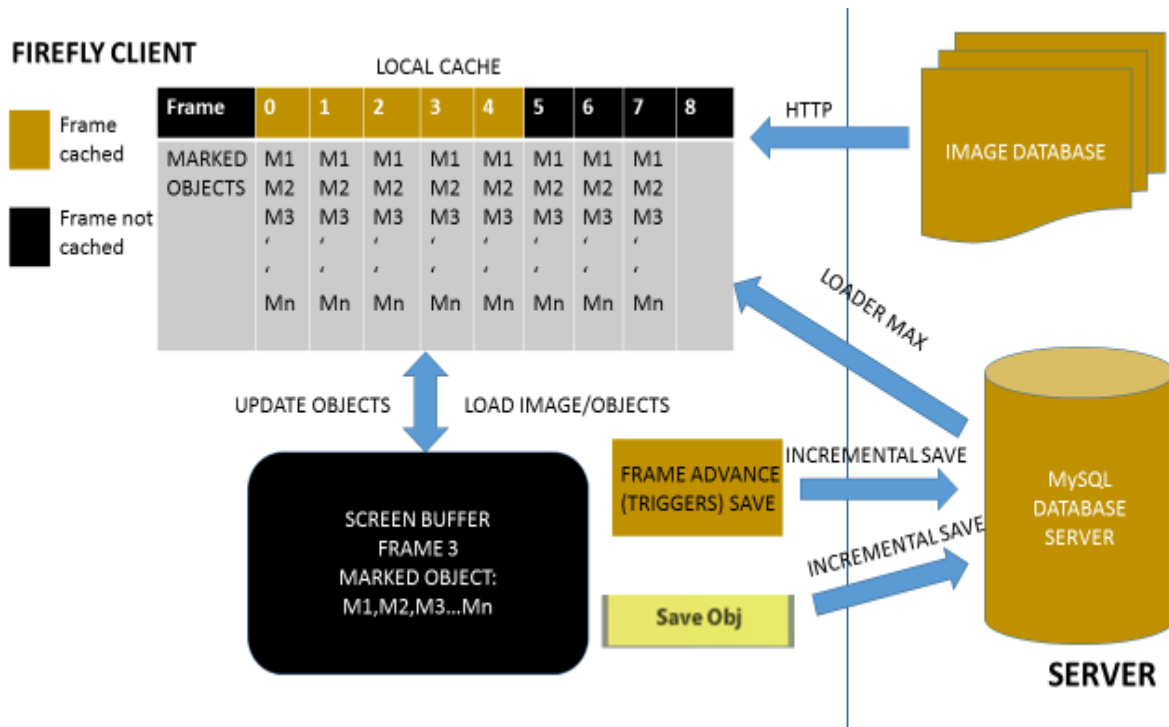


Figure 2.4 Local cache and screen buffer

Whenever there is a change in objects on the screen, that objects is marked and stored in the local cache as dirty. This object is updated and saved in the database when the user changes frame or an explicit save operation is performed. As shown in Figure 2.4 Frame No. 2 is currently loaded in the screen buffer, so Frame 0, Frame 1, Frame 3 and Frame 4 have been fetched and stored in the cache. We use loadermax library to fetch the objects from the database and the images are fetched through HTTP. If any change occurs in the objects M1, M2,.etc. it is marked as dirty. By default these updates are saved in the database when the user goes to next or previous frame. If auto save option is off then there won't be any update when the user switches the image. These updates can be always saved explicitly by using the save button in Save Panel. This new value is returned as a response from the database which refreshes the local cache and the screen buffer.

# Chapter 3

## Multiuser Annotations for Large Image Collections

### 3.1 Restructuring of Database

With the growth of our system, multiuser support became essential. Initially, FireFly supported image set specific instead of user specific annotations. Every sequence of image had annotations (circle, box, polygon, polylines, points etc.) specific to that set of images. For the new datasets like X-rays, Vessels and Malaria we had multiple users distributed all around the country. Different users required their own annotations for each set of images. In order to maintain consistency and avoid confusion, an additional level of FireFly’s hierarchy was introduced and the level names were changed on both the superficial level and the individual annotations to make annotations meaningful. So, in order to preserve consistency and increase the level of data abstraction some changes with respect to hierarchy were made.

FireFly consisted of the one dataset level, which was divided into two different levels:

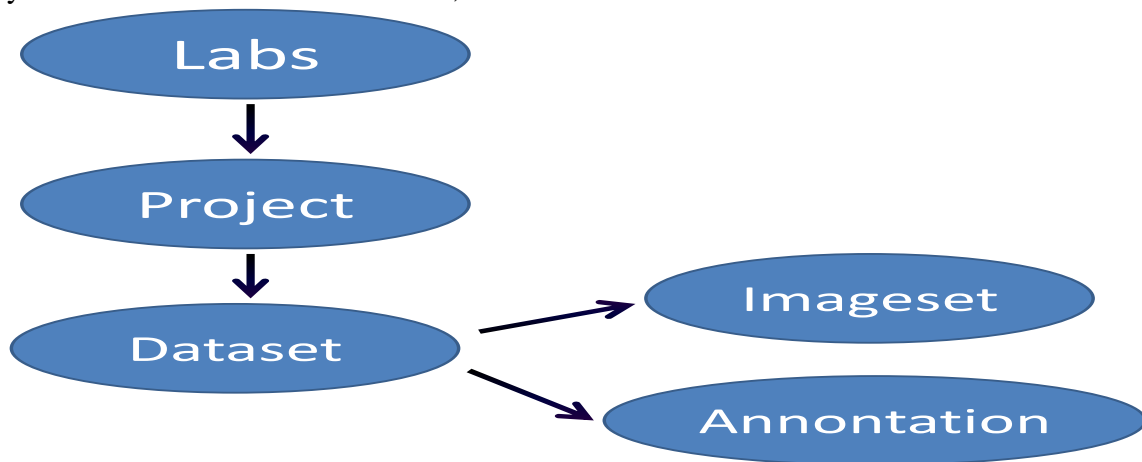


Figure 3.1 Expansion of dataset level into image set and annotation

A. Image set- It consists of sequence or collection of images.

B. Annotations- It consists of annotations. Annotations are markings on objects that can be done by using one of the editing tools (points, lines, circles, boxes, polylines, polygons, freeform etc.).

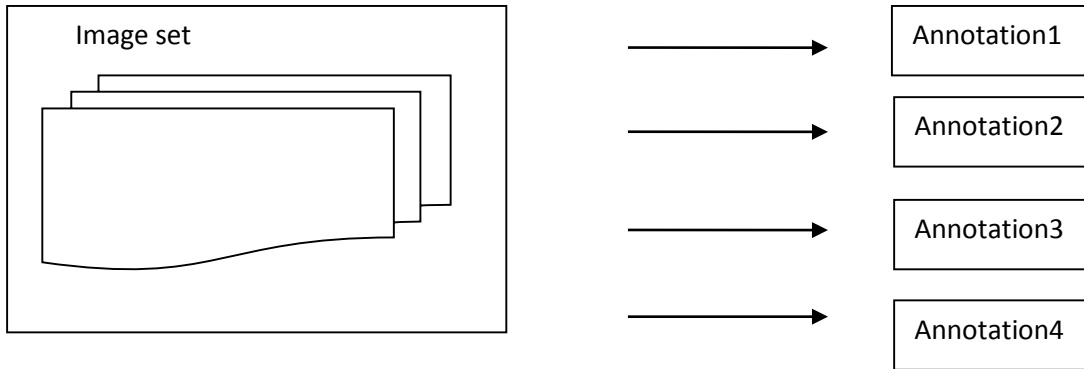


Figure 3.2 Multiple annotations on single image set

### 3.1.1 Impact of Restructuring

Due to changes in the hierarchy, modifications had to be made in the database and code base (consists of PHP services and Flex files). A dual-MVC is maintained on both server and client sides. Thus, the major models, controllers and services had to be modified according to the new changes in the database. Some of the functions had to be rewritten on both server and client side. All these modifications required a careful and timely approach, which also caused new implementations to be put on hold. Restructuring gave FireFly a multiuser approach to annotate same imageset simultaneously by different users. This assisted us in involving multiple radiologists and researchers both at the University of Missouri and National Institute of Health to mark and verify the annotations simultaneously.

### 3.1.2 Old vs New Representation

The old representations had 3 levels as shown in Fig. 3.3

Lab: It consisted of LabID, Title, and Description

Project: It consisted of ProjectID, LabID, AuthID, Creator, and Title

Dataset: It consisted of ProjectID, DatasetID, Type, Edit Date, Title, Permission, and Lock

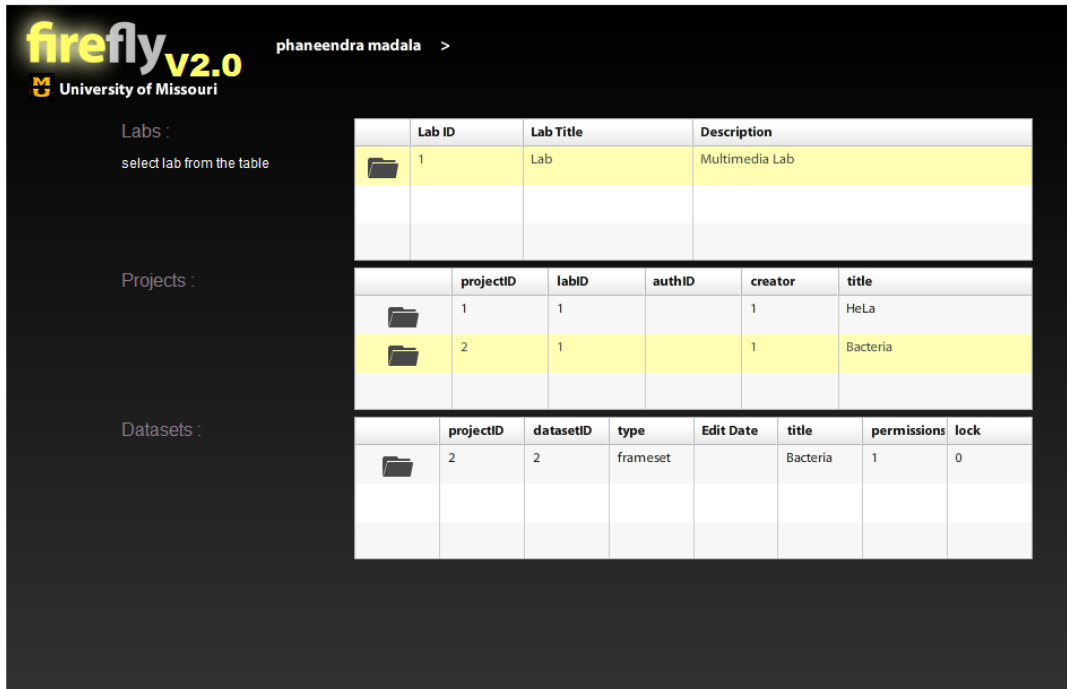


Figure 3.3 Old representation of user labs menu

The new user labs menu has 4 levels of hierarchy:

**Lab:** It consists of LabID, Labname, and Description. At the University of Missouri, our projects can be easily divided into two types- Biomedical and Surveillance. Biomedical projects contain projects such as HeLa, bacteria, satellite cells, wound healing and vessels. Surveillance project contains projects related surveillance and tracking, for example, UPS, FPSS, VIRAT, VIVID, WAMI, and Four Hills etc. The lab for the National Institute of Health is hidden from general public guest account and is only visible for NIH accounts.

**Projects:** It consists of ProjectID, Creator, and Title. It fetches the data from project and project user table. Each project has different image sets.

**Imageset:** This consists of the image level. The images can be a video sequence or collection. This level contains image set name, resolution width, resolution height and number of frames.

Annotation: This level contains Annotation ID, Annotation Title, Permission, and Edit. Annotations are user specific and have user name as suffix. Multiple users can get access to the same annotation and data coherency is maintained by the lock mechanism.

**firefly 3.0**  
University of Missouri

**Labs :**

Lab ID	Lab Name	Description
1	U.Missouri_cs	Biomedical
2	U.Missouri_cs	Surveillance

**Projects :**

ProjectID	Creator	ProjectTitle
4	1	TUD Cardosa HELA
5	1	Princeton Shaevitz Bacteria
10	1	Cornelison Satellite Cells
12	1	UMASS wadsworth Wound healing
16	1	NKITraining_original

**ImageSet :**

Imageset	resolutionWidth	resolutionHeight	Number of Frames
HeLa/TS2/	1024	1024	173
HeLa/TS3/	1024	1024	174

**Annotation :**

AnnotationID	AnnotationTitle	Perm	Edit
1	TS2 Set	R/W	Open
5	HeLa Test	R/W	Open
10	HeLa Vadim	R/W	Open

Figure 3.4 Hierarchy and intuitive name of datasets

FireFly version 3 handles multiple users and several datasets. The naming convention for the datasets or annotations has to be clear and concise to avoid any confusion and maintain data integrity. Initially, there was only one kind of dataset for each set of images, without any intuitive names. With multi-user support, there was a need to have unique and more intuitive names. Annotations were named majorly in two categories:

User Specific-These annotation were user specific and only a specific user was granted R/W access so these were named after the userID.

Test Data- These data were used for training and testing purposes. Several users had access to this data. So these were named with a test suffix.

The project names and image names were changed to make it more intuitive and understandable depending upon source and labs corresponding to the images.

### 3.1.3 Multiple Annotations and Data Sharing

FireFly supports multiple annotations, which can be shared amongst multiple users. A single image set is shared among different users. Multiple users can mark on the same data provided they are provisioned with the access to do it. In order to avoid inconsistency, a lock mechanism has been built in FireFly. If one user is logged in an annotation, the annotation is locked for other users. Other users can access the annotation only in 'Read' mode. Read mode allows the user to read the data but doesn't give permission to edit the data. As the first user logs out, the lock is released in the database. Once the lock is released other users can log in and access in Read/Write mode depending on their role in database.

Our future goal would be to copy annotations from one user to another user through GUI. Currently this feature is provided at the backend but is not present in the GUI. This would create an easy way to exchange data between users.

## 3.2 Video Sequence vs Image Collection

FireFly was designed to handle continuous video sequences. The image sequences were composed of images in increasing order of frame numbers.

The continuous video sequence had two benefits:

1. Transferring image over HTTP was easier as the images were in a definite order. This allowed us to buffer next 2 and previous 2 images. For e.g. For Frame no. 10 we can fetch +2 and -2 images easily. It provides a faster approach to fetch and display images.
2. It also took less storage space in the database and allowed us to maintain a unique link between frame Number and marked Objects.

The Bio-Medical X-rays and Vasculature images are mostly collection of images instead of a sequence with no specific pattern or number. These image names mostly depended on the date and patient's records. The random collection of images with random names made selection of the next



image difficult as there was no fixed sequence to identify the next image to be transferred over HTTP.

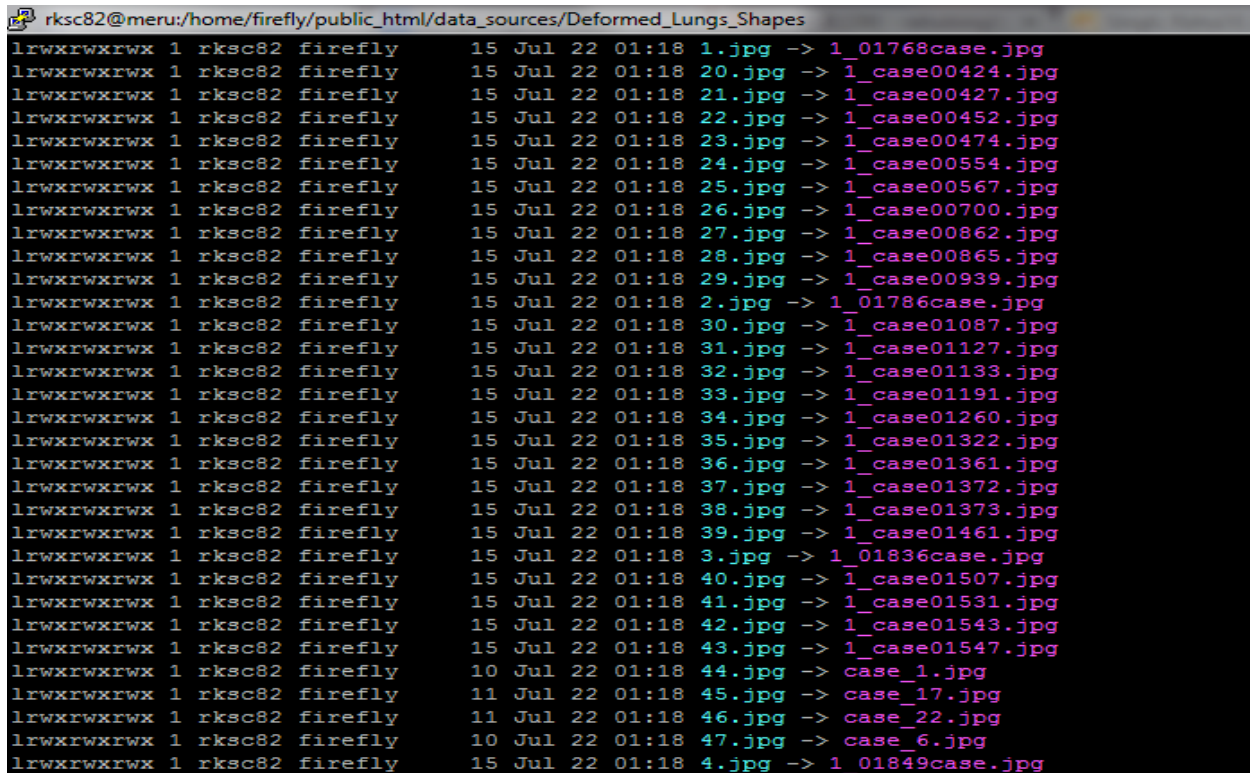
### 3.2.1 Symbolic Links

The image names correspond to something specific and important, so preserving name was crucial. This increased our problem since FireFly only supporting a video sequence and not image collections. A change in code and database was not feasible, as it would require a big change and could also cause duplication of code and database tables. A short and feasible solution was to generate symbolic links.

There are two types of links that can be made in Linux:

- Symbolic links: Refer to a symbolic path indicating the abstract location of another file
- Hard links: Refer to the specific location of physical data.

With the creation of symbolic links, the images could be stored outside the FireFly folder.



```
rksc82@meru:/home/firefly/public_html/data_sources/Deformed_Lungs_Shapes
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 1.jpg -> 1_01768case.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 20.jpg -> 1_case00424.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 21.jpg -> 1_case00427.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 22.jpg -> 1_case00452.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 23.jpg -> 1_case00474.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 24.jpg -> 1_case00554.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 25.jpg -> 1_case00567.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 26.jpg -> 1_case00700.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 27.jpg -> 1_case00862.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 28.jpg -> 1_case00865.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 29.jpg -> 1_case00939.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 2.jpg -> 1_01786case.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 30.jpg -> 1_case01087.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 31.jpg -> 1_case01127.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 32.jpg -> 1_case01133.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 33.jpg -> 1_case01191.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 34.jpg -> 1_case01260.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 35.jpg -> 1_case01322.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 36.jpg -> 1_case01361.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 37.jpg -> 1_case01372.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 38.jpg -> 1_case01373.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 39.jpg -> 1_case01461.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 3.jpg -> 1_01836case.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 40.jpg -> 1_case01507.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 41.jpg -> 1_case01531.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 42.jpg -> 1_case01543.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 43.jpg -> 1_case01547.jpg
lrwxrwxrwx 1 rksc82 firefly 10 Jul 22 01:18 44.jpg -> case_1.jpg
lrwxrwxrwx 1 rksc82 firefly 11 Jul 22 01:18 45.jpg -> case_17.jpg
lrwxrwxrwx 1 rksc82 firefly 11 Jul 22 01:18 46.jpg -> case_22.jpg
lrwxrwxrwx 1 rksc82 firefly 10 Jul 22 01:18 47.jpg -> case_6.jpg
lrwxrwxrwx 1 rksc82 firefly 15 Jul 22 01:18 4.jpg -> 1_01849case.jpg
```

Figure 3.5 Soft links with the actual image names

A symbolic link (also known as a soft link) consists of a special type of file that serves as a reference to another file or directory. UNIX/Linux like operating systems often use symbolic links.

To create a symbolic link in UNIX or Linux, at the shell prompt, following command can be used:

`ln -s {target-filename} {symbolic-filename}`

`ls -l`: It lists all the links and their actual file names.

By creating multiple links, entire set of images can be linked to a specific frame numbers. These frame numbers can be utilized for fetching and displaying on the client side. The same number is also used in the database to store annotations. Additional problems like creating copy of images in our `data_sources` folder were also resolved by creating symbolic links.

IM_No	annotationID	frame_No	resolution_Height	resolution_Width	frame_name	info
14	25	0	1430	1722	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.122846.101554	India/
15	25	1	1430	1722	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.123124.101554	India/
16	25	2	2010	2446	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.124716.101554	India/
17	25	3	1572	2010	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.132411.101554	India/
18	25	4	2010	2446	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.154331.101554	India/
19	25	5	2010	2446	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.162429.101554	India/
20	25	6	2010	1572	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.170002.101554	India/
21	25	7	2010	2446	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.91939.101554	India/
22	25	8	2010	2446	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.93534.101554	India/
23	25	9	2010	2446	IM-00-1-1.2.392.200036.9107.500.305.1554.20110901.94818.101554	India/
24	25	10	2010	2446	IM-00-1-1.2.392.200036.9107.500.305.1554.20110902.110411.101554	India/
25	25	11	2010	2446	IM-00-1-1.2.392.200036.9107.500.305.1554.20110902.110807.101554	India/

Figure 3.6 Database table to store image information

We could now access the files by using these links remotely which reduced duplication of images. Since each image was of a different size so we had to explicitly create a table in database to store the extra information. Figure 3.6 shows the database table to store the extra information.

A CGI script written in PERL was used to perform following functions:

1. Generation of soft links and linking them with actual images.

2. Creating a text file, containing the image names with corresponding frame numbers. The text file is stored in the results folder that can be later downloaded by the user using 'Download Results' in 'Data Analytics' panel.
3. The script invokes the PHP script to load the details like name, resolution height, resolution width, and info in the database.

With the help of the symbolic link we were able to efficiently handle all kinds image collections such as lung X-rays, vessel images.

# Chapter 4

## Segments and Contours Editing

### 4.1 Need for Segments and Contours Editing

Manual Editing of automated segmentation results has always been a challenging task. Segmentation editing involves changing of parameters such as width, height and length of the graphical shapes, which may involve different levels of complexities depending on the contour. Box and line objects in FireFly supported simple editing. However, Figures, which were being used to represent segmentation results on the image, were primarily polygons and polylines. Thus, simple editing had to be extended and modified for polygons and polylines.

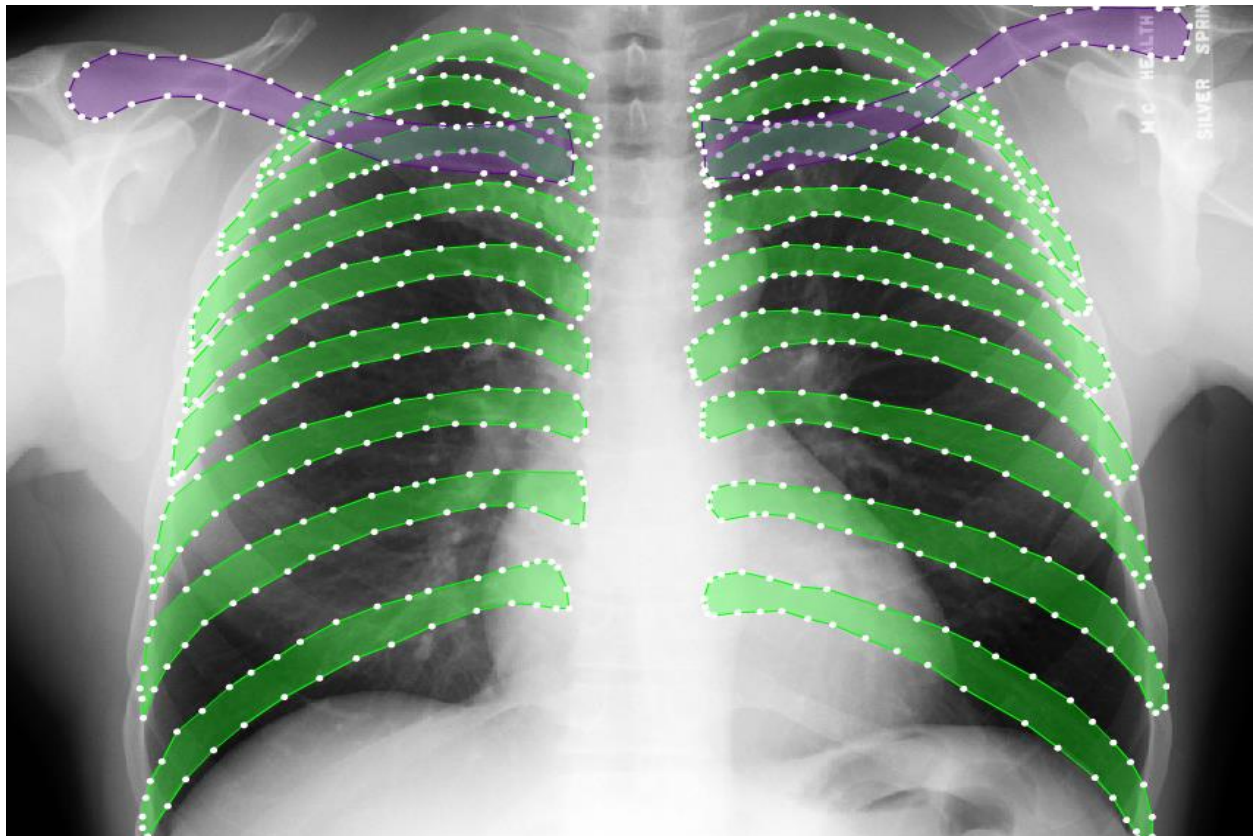


Figure 4.1 Lung boundaries with edit points

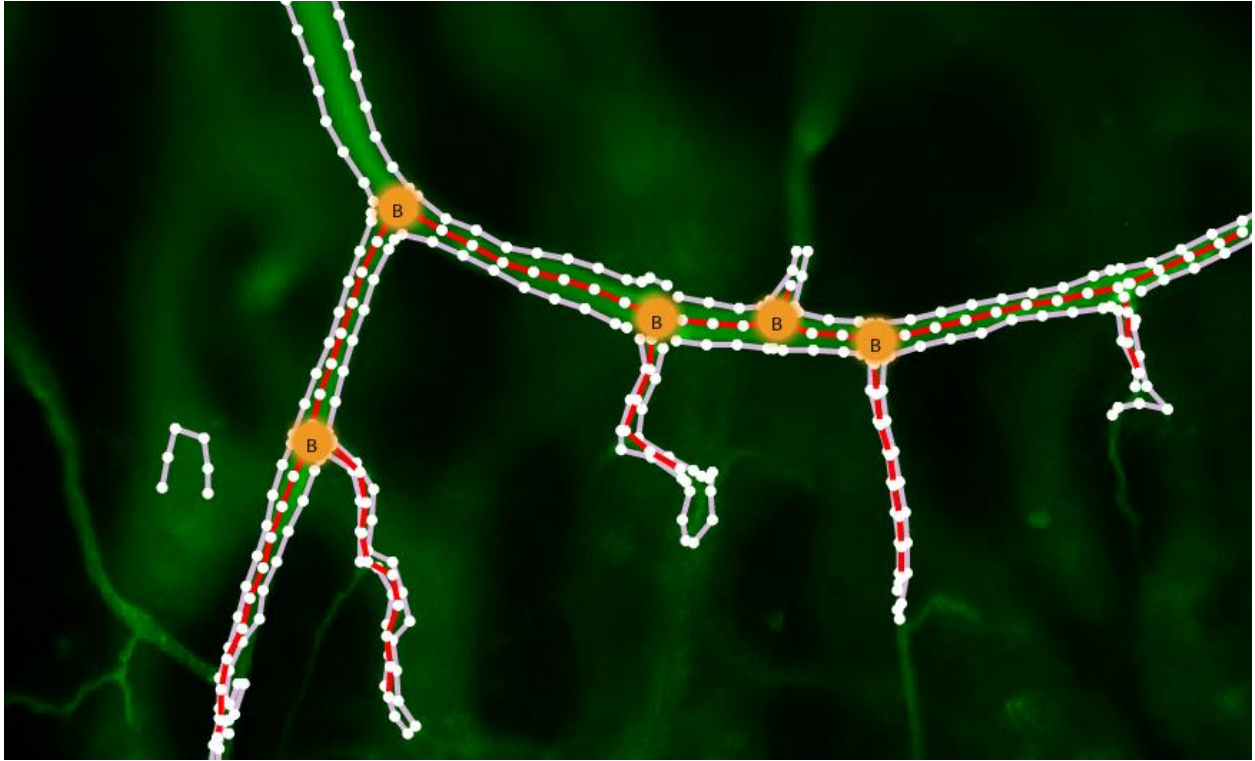


Figure 4.2 Vessels with edit points

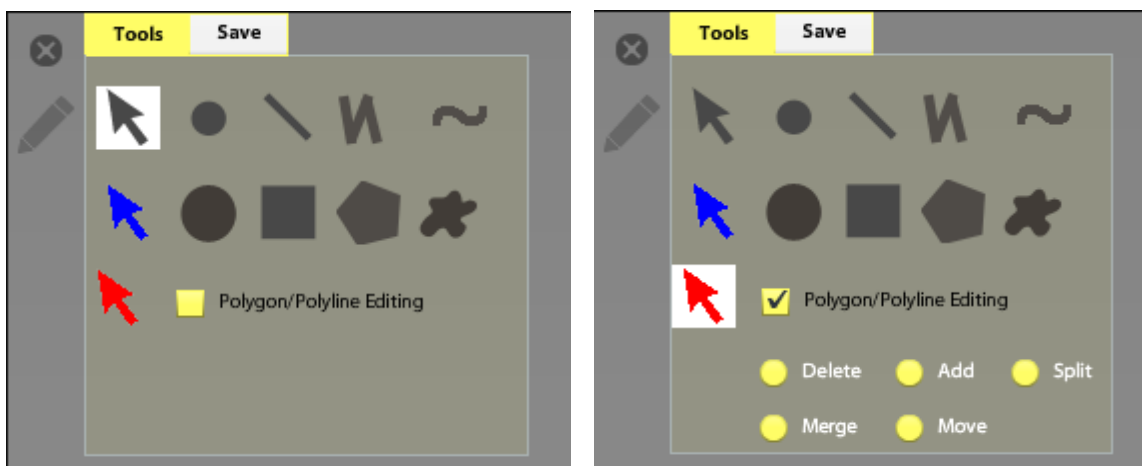
A variety of editing operations were added for smooth editing of contours and segments. Our recent work consists of large datasets with more than thousand points along the contours, for e.g. Lung X-rays and Vessels as shown in Figures 4.1 and 4.2. The points visible on the screen are sub sampled pixels from the actual result generated from MATLAB. Besides simple movement of the contour points, it also required additional manual editing operations to correct the contour data. Any merge or split operation involves creation and deletion of objects from the database. To maintain consistent information in the database, auto-populating of ParentID, ChildrenID and label properties was also needed along with updated coordinates. Practically, this manual edit operation and manual update is time consuming and infeasible for large population of objects. Therefore, an automatic segment editing and updating algorithm is required when an edit operation is made on an object. This involves studying all possible scenarios of manual edit operations. The possible instances, when manual correction is required are:

- Adding vertex in contour and segments

- Deleting vertex in contour and segments
- Splitting of contour and segments
- Merging of contour and segments
- Cases involving multiple edit operations

The edit operations in FireFly are contained within the Tool Panel. The editing option is hidden by default as shown in Figure 4.3 (a). Polygon/Polyline Editing checkbox must be selected to show all the possible options on the screen. The checkbox can be selected by three different ways:

1. By pressing Ctrl+P, a user can enter into Polygon/Polyline mode. If the user is already in P/P mode the user can press Ctrl+P to exit the mode.
2. By checking the checkbox also a user can enter into P/P mode. Once the user has selected the checkbox he enters into the editing mode. To exit, the checkbox can be unselected.
3. Red Arrow indicates the user is in P/P mode. A user can directly click on this arrow to enter into P/P mode. Pressing escape also exits the P/P mode.



(a) Without edit mode

(b) With edit mode

Figure 4.3 Tool panel

## 4.2 Adding Vertex in Contours and Segments

A contour can be a closed figure, which is represented by a polygon or an open segment represented by a polyline.

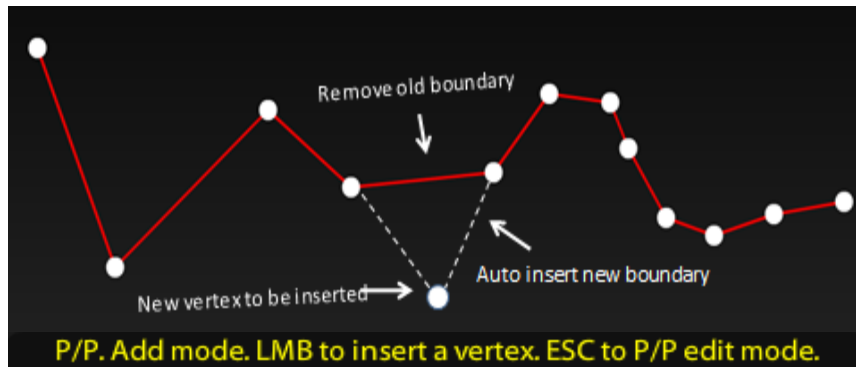


Figure 4.4 Segment represented by polyline showing the vertex to be inserted

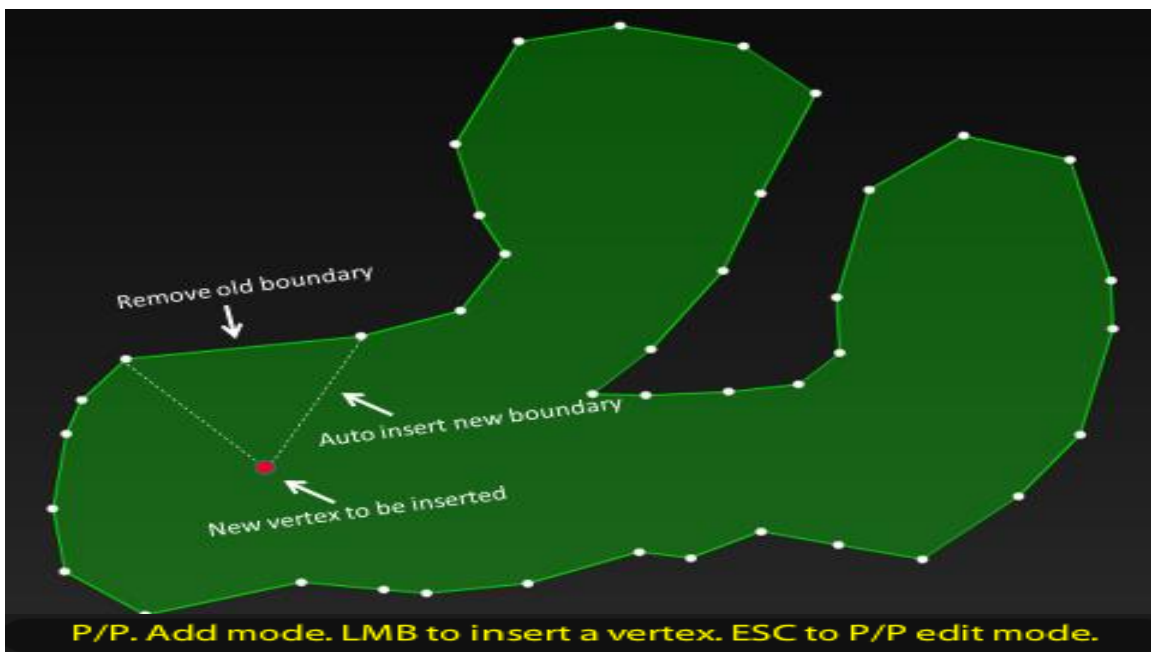


Figure 4.5 Contour represented by polygon showing the vertex to be inserted

### User interface operations

Figures 4.4 & 4.5 show the mechanism of vertex insertion. For inserting a vertex, a user must be in editing mode. To make insertion active, the add radio button must be selected. A user can

also enter insertion mode by pressing 'A' button on keyboard. Inserting a vertex involves only a LMB click. A vertex is inserted at the point of click. The old boundaries are deleted and new boundaries are drawn and the new vertex is inserted. Multiple vertices can be added with continuous clicks. As the vertices are added, the shape is auto updated with new boundaries. Figures 4.6 & 4.7 show the objects after insertion of vertices.

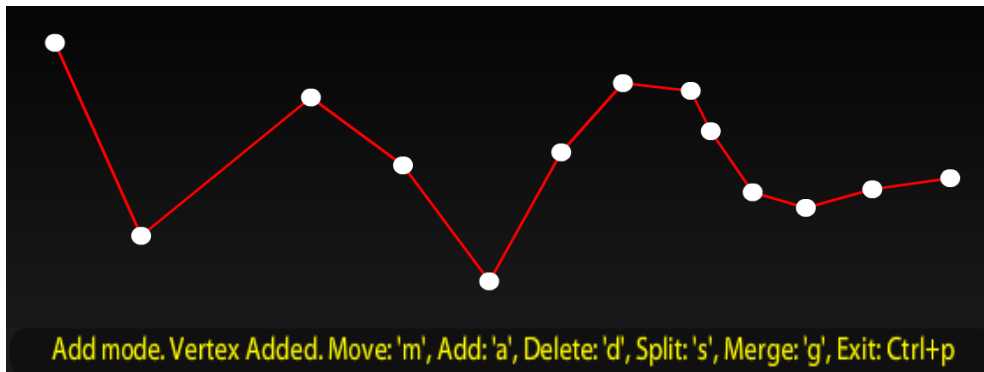


Figure 4.6 Segment after add operation

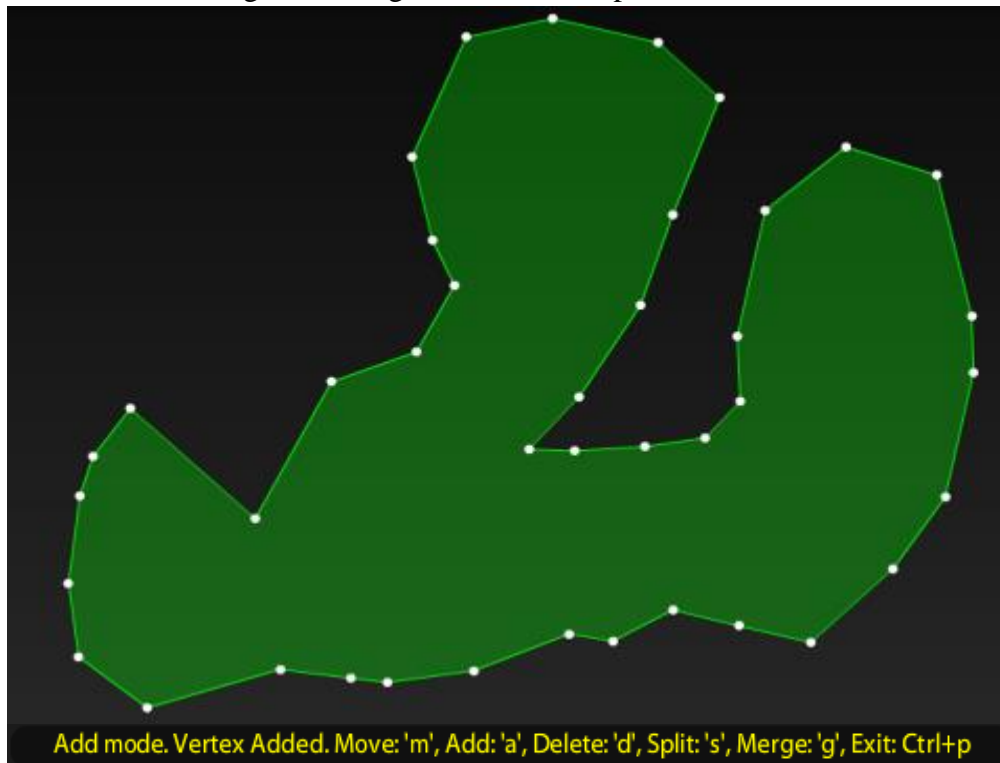


Figure 4.7 Contour after add operation



## Backend operations

The new vertex inserted can be at any position. As all the points are stored sequentially in an array, an automatic array update is needed at the client side and the database. The insertion of new vertex takes place by using the following algorithm:

Step 1: The mid-point is calculated for each pair and stored in a data structure

Step 2: Euclidian distance is found between each mid-point and the new vertex

Step 3: The midpoint with the least distance is chosen

Step 4: The new vertex coordinates are inserted in between the pair with minimum distance from the new vertex.

Step 5: The updated value is shown on the screen with deletion of new boundaries and drawing of new boundaries. Simultaneously, the data is transmitted through AMF to the client side and the database is updated simultaneously.

## 4.3 Deleting Vertex in Contours and Segments

In the process of contour and segment editing, the deletion of a vertex is similar to addition.

### User interface operations

For deleting any vertex, the user has to select the delete radio button. A user can also enter into delete mode by pressing 'D' key on the keyboard. LMB click provides the automatic deletion feature. Multiple vertices can be deleted by clicking on the vertices in continuation. Auto-update is being performed at the backend and new updated Figure is continuously displayed on the screen. Figures 4.8 and 4.9 show the vertex that needs to be deleted. Hovering of the mouse over the vertex changes the pointer to a hand cursor. LMB click will mark the vertex for deletion. The vertex is marked with yellow color and is removed after the update. To exit the deletion mode, the user has to uncheck the box or select another editing option or exit P/P mode.

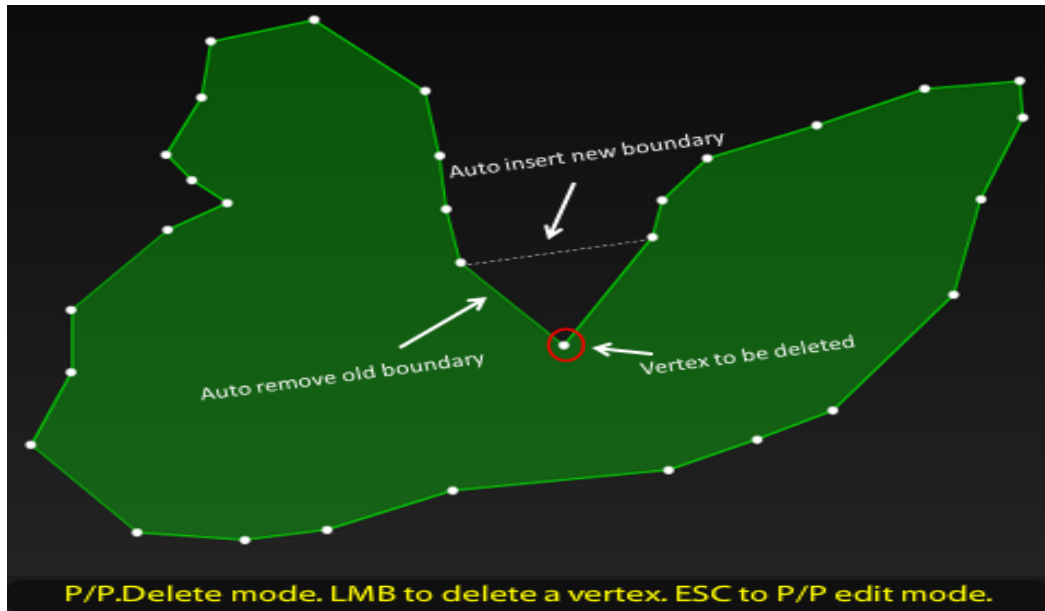


Figure 4.8 Contour with vertex to be deleted

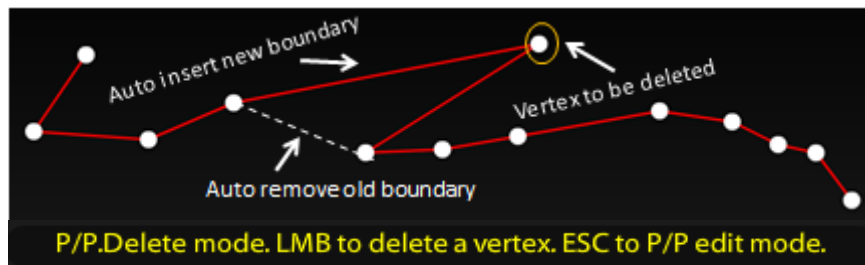


Figure 4.9 Segment with vertex to be deleted

### Backend operations

The vertex deleted can be at any position. An automatic update is needed to store the new changed points. The deletion of new vertex takes place by using following steps:

Step1: An invisible circumference is created around the vertex. As the mouse enters this area, the cursor changes to a hand, which indicates that the vertex can be selected.

Step 2: LMB click marks the vertex for deletion. After selection, the vertex is removed from the data structure.

Step 3: The data structure is updated with the remaining vertices.

Step 4: The updated value is shown on the screen with deletion of old boundaries and insertion of new boundaries. Simultaneously, the data is updated in the database.

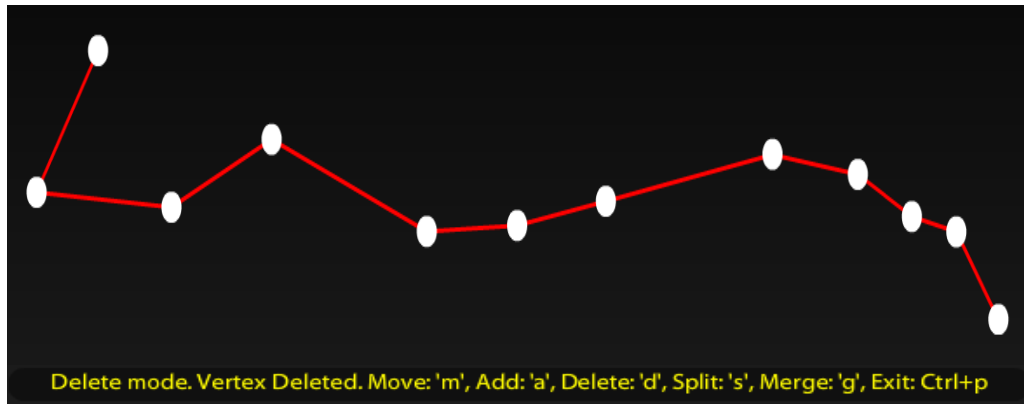


Figure 4.10 Segment after deletion of vertex



Figure 4.11 Contour after deletion of vertex

#### 4.4 Splitting of Contours and Segments

The result of algorithmic analysis can segment out a portion as a whole instead of different individual objects. Due to noise present in image or wrong parameter values can cause variance in the segmentation results. This may lead to incorrect perception of objects and create some false branches along with original objects. As shown in the Figures 4.12 and 4.13 splitting is necessary between objects. To enter into split mode a user has to either select Split option in the tools panel or press 'S' from the keyboard.

## User interface operations

Splitting operation is done differently for closed contours and open segments. For a closed contour, the user must select two pairs of vertices in the object whereas, for open segment, the user has to select one pair of vertices in the object.

Steps for closed contour:

Step 1: LMB click 1 and LMB click 2 to select the first pair. The color of the vertices turns yellow to signify the vertices has been selected. LMB click 3 and LMB click 4 for selecting second pair and the vertices turn yellow. The split operation is performed in the backend and updated Figure is displayed on the screen.

Note: In split operation, the first vertex selection in both the pairs represents a joining edge of the first object. The second vertex selection of the pair represents a joining edge for the second object.

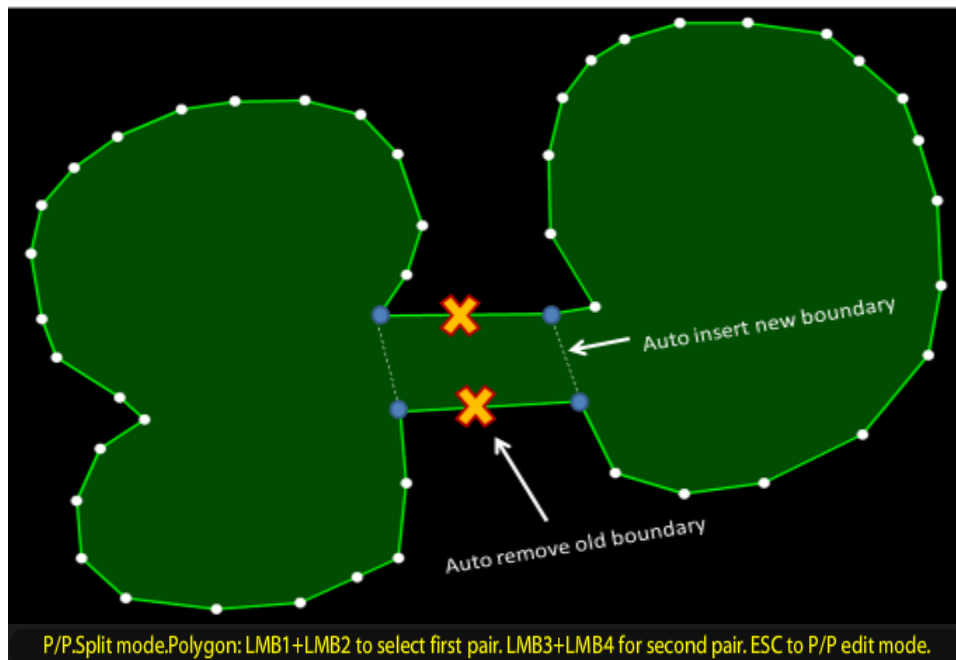


Figure 4.12 Splitting of a contour

Steps for open segment:

Step1: LMB click 1 marks the end of the first object

Step2: LMB click 2 marks the starting of second object

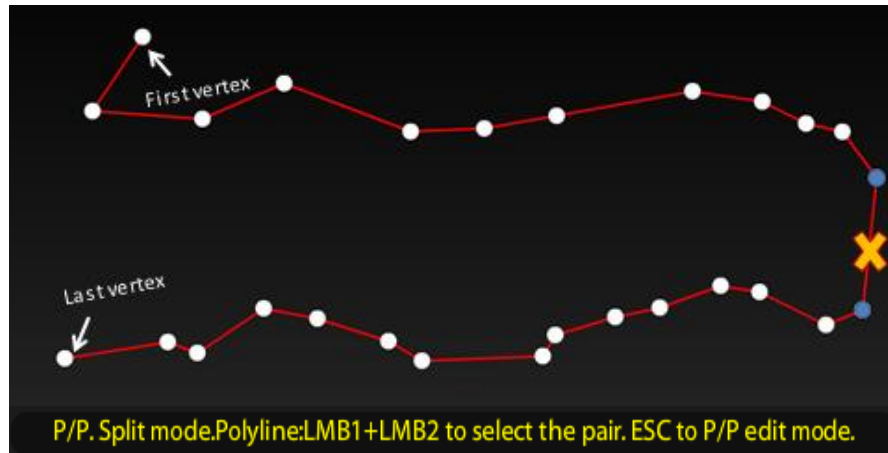


Figure 4.13 Splitting of a segment

#### Backend Operations for Closed Contour

The complexity for this operation was to update two different data structures with new points, such that not only is the old data preserved but also the newly created data doesn't overlap with the existing data. Each object contains an array of points in increasing order of their drawing, which are connected on the screen. To update the data structure efficiently, following steps were performed-

Step1: The vertices that are derived in the data structure for object 1 are as follows:

- a. First vertex of the selected data structure to vertex 1 of first selected pair
- b. First vertex of second selected pair to the last vertex.

Step2: A new object is created with new set of vertices derived from the original contour. The vertices that are derived in the object 2 data structure are as follows:

- a. Second vertex of the selected pair 1 to second vertex of selected pair 1.

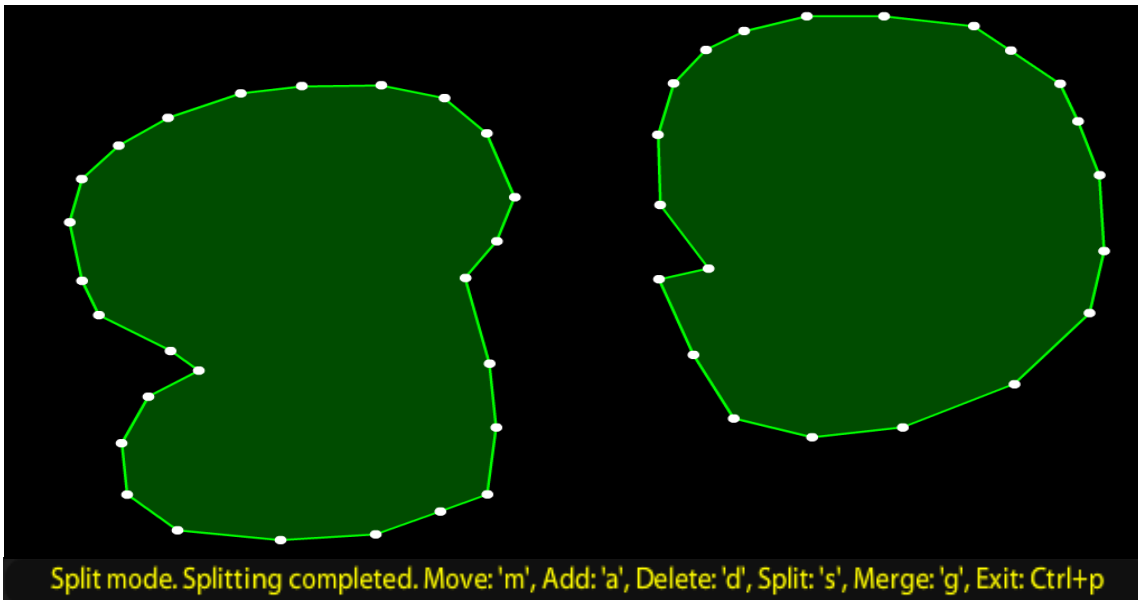


Figure 4.14 Contour after split operation

Backend operation for open segment

An open segment can be broken along two adjacent vertices, which the user must select.

Step1: The vertices from first vertex to first selected vertex are derived on the first data structure for object

Step2: The vertices from the second vertex selected to the last vertex are derived into the data structure for the new object.

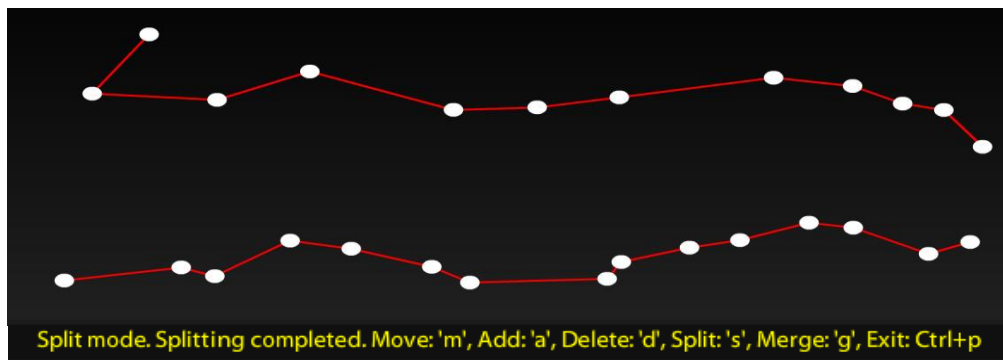


Figure 4.15 Segments after split operations

## 4.5 Merging of Contours and Segments

Disturbances or noises in algorithmic analysis may yield separate patches of segments or contours instead of a single contour or segment. In order to correct the errors, the merge operation can be used. The segments and contours have different steps for merging. To enter into merge mode a user has to select merge from the panel or press 'G' from the keyboard.

### Steps for Closed Contour

Step 1: 'LMB click 1' and 'LMB click 2' to select the first pair in the first object. The color of the vertices is turned to yellow to signify it has been selected.

Step2: 'LMB click 3' and 'LMB click 4' for selecting second pair in the second object. The color of the vertices turns to yellow to signify it has been selected. The merge operation is performed and updated figure is displayed on the screen.

Note: In merge operation the first vertex selection from both the pairs represents the first joining edge between two objects. The second vertex selection from both pairs represents the second joining edge between two objects.

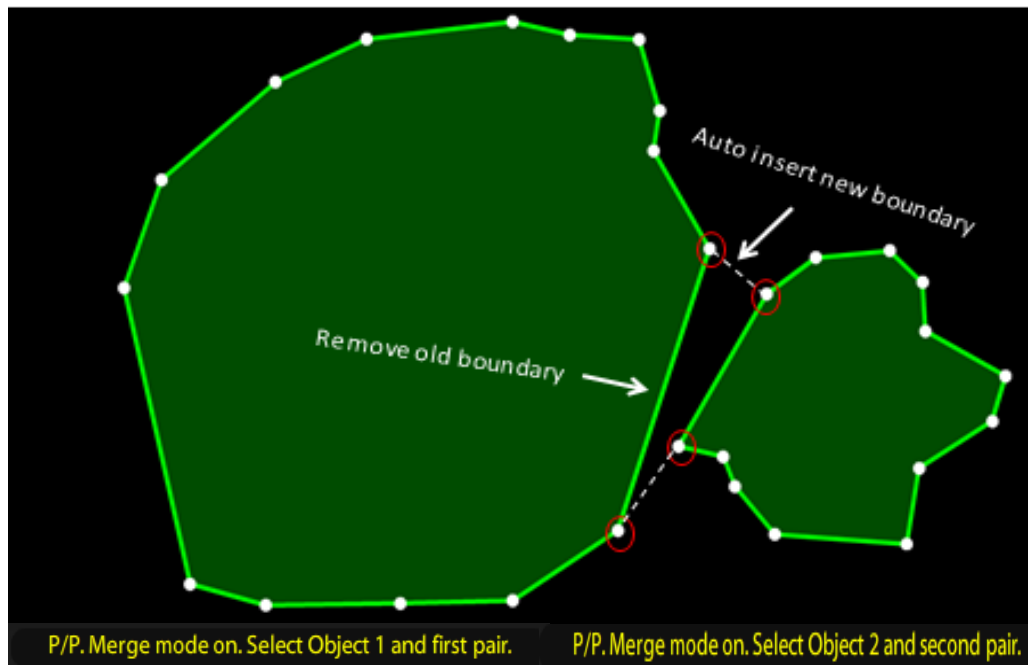


Figure 4.16 Merging of closed contour

## Backend Operations for Closed Contour

The complexity for this operation was to update one data structure with combination of two data structures while maintaining the correct order of the vertices. Each object contains an array of points in increasing order of their drawing, which are connected on the screen. To update the data structure efficiently, following steps were performed.

The vertices are derived based upon the first vertex of the data structure and the selected pair of vertices. The vertices are derived in the counter clockwise direction.

First, the vertices from Object 1 are derived and second, the vertices from Object 2 are derived in the new data structure. However, the order in which the vertices are derived depends upon the following scenarios-

### Object 1

Scenario 1: Selected vertex 1 is the first vertex

Step1: The vertices are derived from first vertex to the second vertex selected

Scenario 2: Selected vertex 2 is the last vertex

Step 1: The vertices is derived from the first selected vertex to the last vertex

Scenario 3: Selected vertex 1 and vertex 2 is NOT the first vertex or the last vertex

Step 1: The vertices are derived from selected vertex 1 to the last vertex

Step 2: The vertices from first vertex to selected vertex 2 are derived

### Object 2

Scenario 1: Selected vertex 1 is the first vertex

Step1: The vertices are derived from selected vertex 1 to selected vertex 2

Scenario 2: Selected vertex 2 is the last vertex

Step 1: The vertex is derived from selected vertex 1 to the last vertex

Scenario 3: Selected vertex 1 and vertex 2 are not first or last vertices



Step 1: The vertices are derived from selected vertex 1 to last vertex

Step 2: The vertex from first vertex to selected vertex 2 are derived

Firstly, the vertices are derived from object 1 and then from object 2. Since the contour is drawn in the order of vertices that are saved in the database, it is necessary to maintain correct order of the vertices.

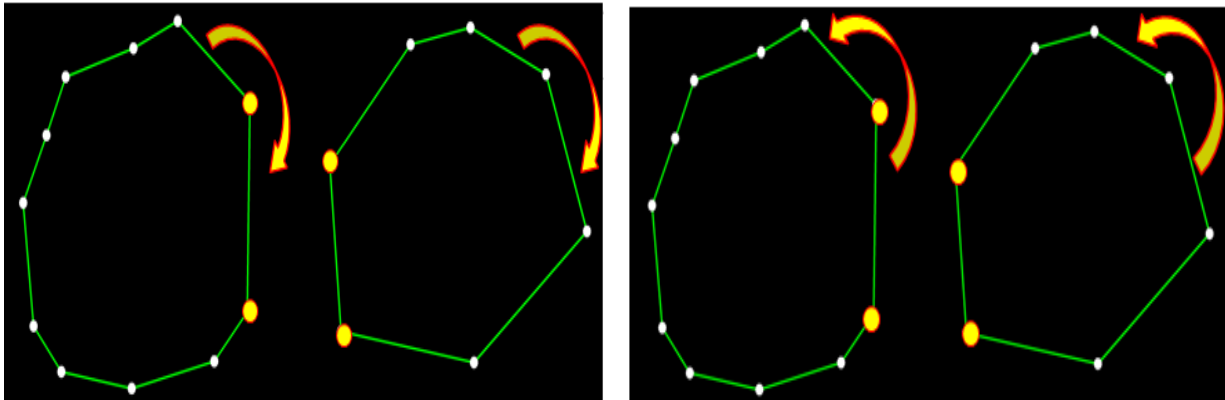


Figure 4.17 Merged contours

#### 4.5.1 Complexity with Merging of Contours

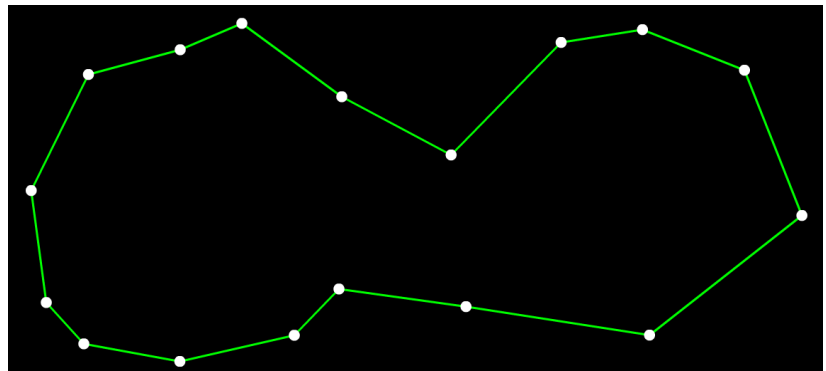
Contours are represented by polygons in FireFly. Currently based on above algorithm, merging is different if the polygons are drawn in same direction or opposite direction. On the UI it looks the two polygons drawn in opposite direction look. But at the backend these are two different scenarios. FireFly stores all the vertex coordinates in the ascending order as it is drawn. This order is used while drawing a polygon on the screen every time. There are two ways to merge two polygons between any given 2 pairs of vertices if they don't have any vertices in between selected vertices in each pair. Figure 4.18(a) shows two polygons which were drawn in clockwise direction and Figure 4.18(b) shows two polygons which were drawn in anticlockwise direction. Here the result is Figure 4.18(c) as both are in same direction so the algorithm works fine. Figure 4.19 shows the same operation done on two polygons drawn in opposite directions. Figure 4.19(a) shows two

polygons drawn in opposite direction. Figure 4.19(b) shows the final results after merging. Here results become different as the polygons were drawn in opposite directions.



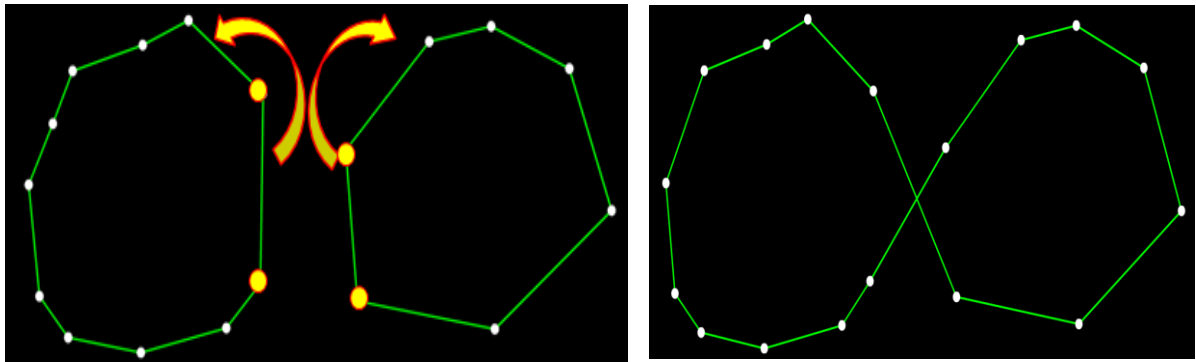
(a) Both polygons in clockwise direction

(b) Both polygons in anticlockwise direction



(c) Result of merge in both cases

Figure 4.18 Merging of two polygons with same orientation

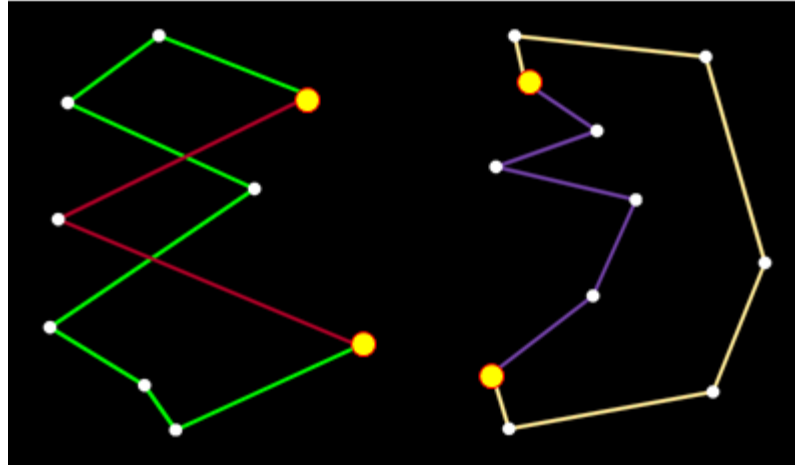


(a) Polygons in opposite directions

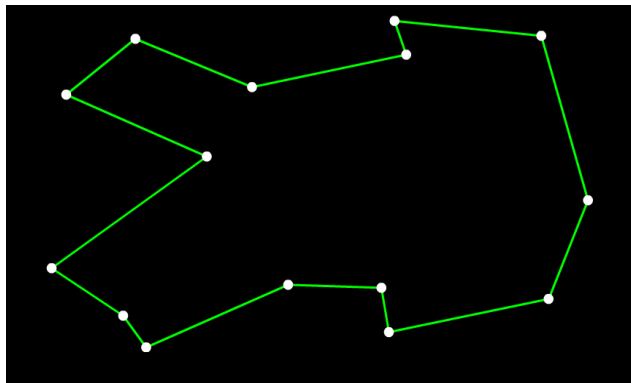
(b) Results after merge

Figure 4.19 Merging of polygons drawn in opposite directions

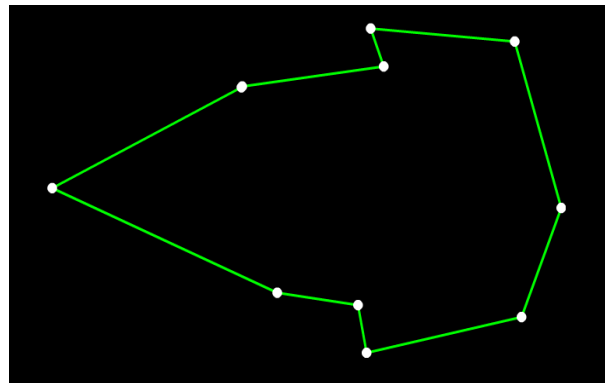
Complexity increases if we have points between selected pair of vertices. As shown in Figure 4.20(a) there are four segments that can be selected, this selection can produce 4 different kinds of shapes as show in Figure 4.20(b), (c), (d), (e).



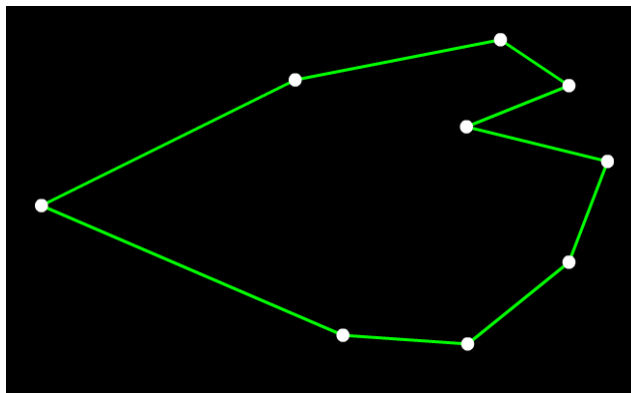
(a) Merging options for complex polygons



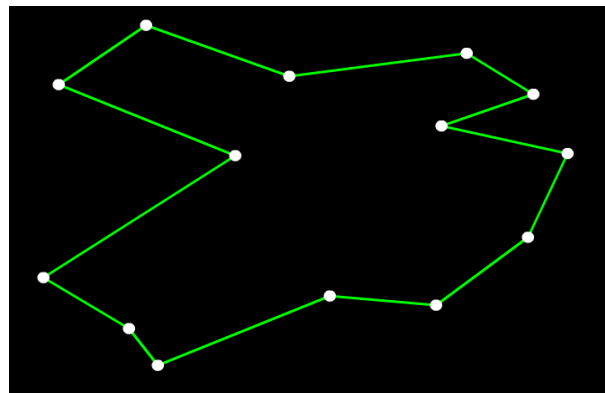
(b) Joining two outer segments



(c) Joining inner in 1 and outer in 2



(d) Joining both inner segments



(e) Joining outer 1 and inner 2

Figure 4.20 Merging in complex polygons

The problem here that the orientation of the polygon can't be determined. As the polygons can be drawn in either direction, determining its orientation only with the help of vertex coordinates is challenging. One way would be to let users decide which segments to choose but this may require a lot of mouse clicks. Currently, with the help of Firefly we can generate polygons only with outer segments and the inner segments are ignored. The inner join would be a cross if the Polygons are drawn in different direction and join would be straight with no cross if the polygons are drawn in same direction. Our future work would be to support all the possible cases.

Steps for Open Segment:

Step1: 'LMB click 1' marks the connecting point in the first object.

Step2: 'LMB click 2' marks the connecting point in second object.

The connecting of a segment can be done in 4 ways. The resulting shape is dependent upon the vertices that are selected to be joined.

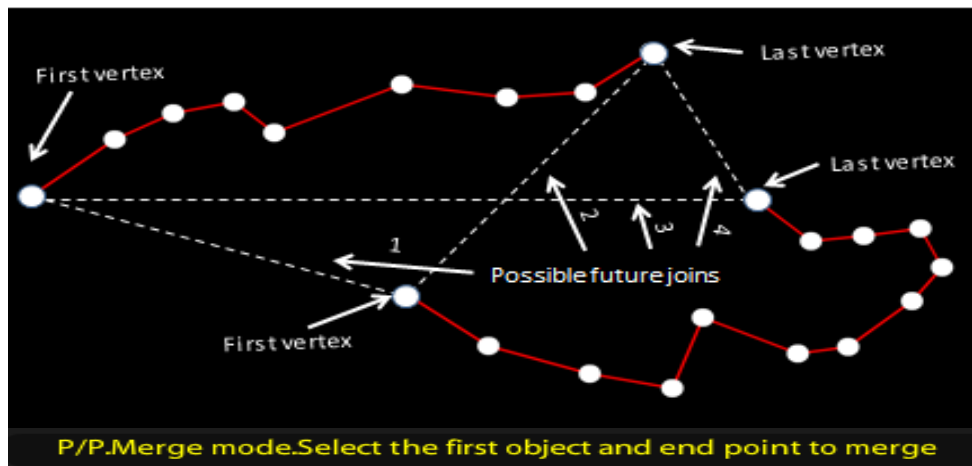


Figure 4.21 Merging of segments

1. Joining first vertex of object 1 to first vertex of object 1 as shown in Figure 4.22

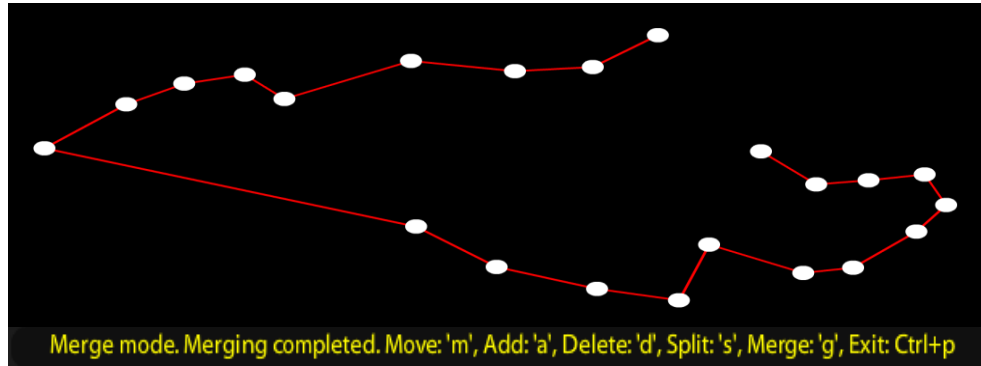


Figure 4.22 Results after merge operation with join1

2. Joining first vertex of object 2 and last vertex in object 1 as shown in Figure 4.23

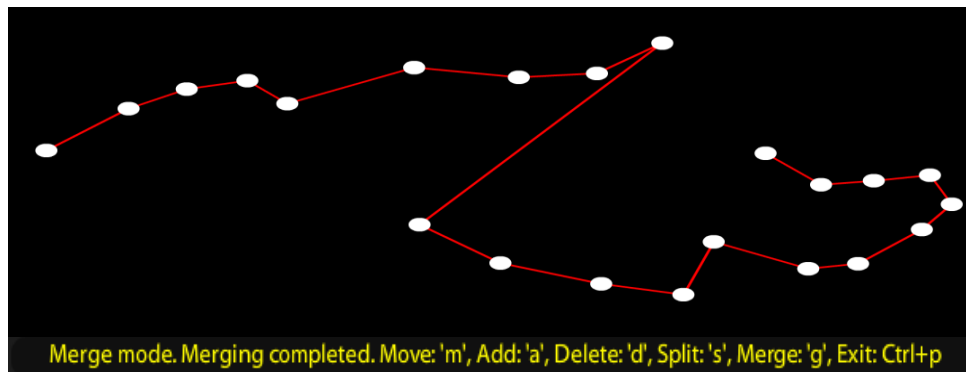


Figure 4.23 Results after merge operation with join 2

3. Joining first vertex of object 1 and last vertex in object 2 as shown on Figure 4.24

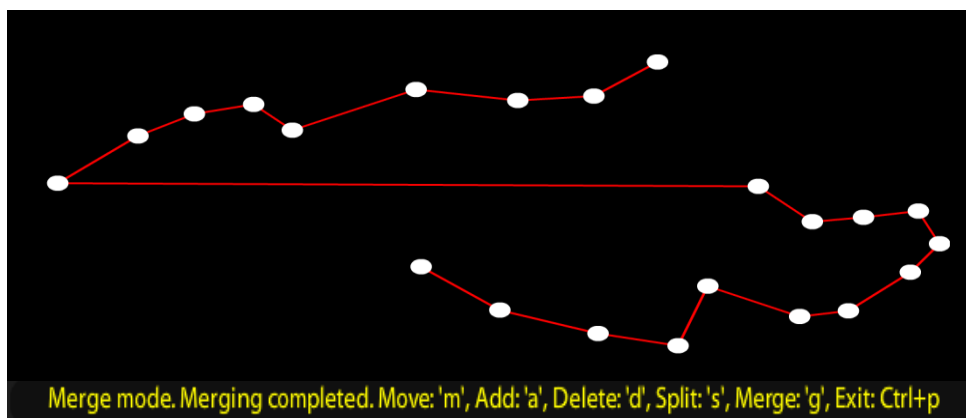


Figure 4.24 Results after merge operation with join 3

4. Joining last vertex in object 1 and last vertex in object 2 as shown in Figure 4.25

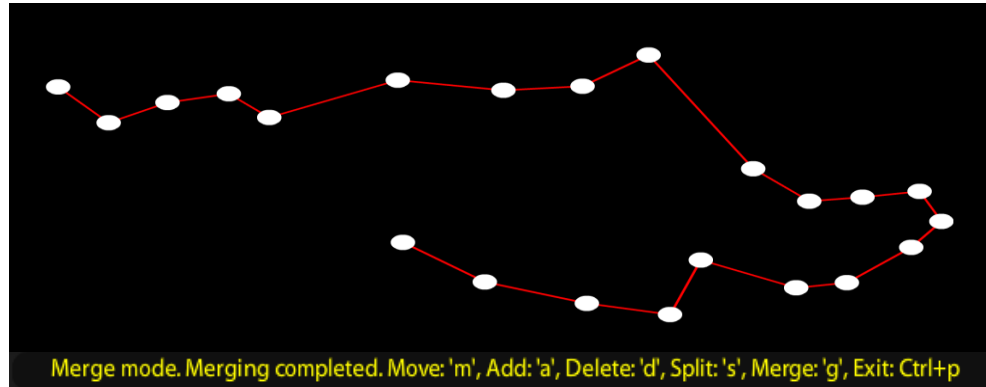


Figure 4.25 Results after merge operation with join 4

Backend operation for open segment

An open segment can only be joined at either the first vertex or the last vertex. Depending on the vertex that is selected by the user, different backend operations might be taking place.

Scenario 1: First vertex of Object 1/Object 2 and first Object 1/Object 2. The resultant shape will be same in both the cases however, the format in which it is stored in the database changes

Step1: The vertices from first to last are derived from the first object

Step2: After the last vertex of first object, the vertices from first to last are derived from the second object.

Depending upon which object is chosen first, the first and last vertices are decided.

Scenario 2: First vertex of object 1 and last vertex in object 2

Step 1: The vertices from first to last of Object 1 are derived

Step 2: The vertices are derived from last to first (in reverse order) at the end of last vertex of first Object. The first vertex of object 2 becomes the last vertex

Scenario 3: Joining first vertex of object 1 and last vertex in object 2.

Step 1: The vertices from Object 1 are derived from first to last

Step 2: The vertices form Object 2 are derived from last to first

Scenario 4: Joining Object 1 last vertex to Object 2 last vertex

Step 1: The vertices from Object 1 first to last is derived

Step 2: The vertices from Object 2, last to first is derived (in reverse order)

## 4.6 Cases Involving Multiple Edit Operations

Sometimes the user may require a figure which can't be achieved by just one operation. Multiple operations can be done one after another for achieving these figures. Some of the cases have been discussed below. The vertices marked with yellow represent the active vertices. These vertices are either deleted, added or can be a merging or splitting point in the contour or segments.

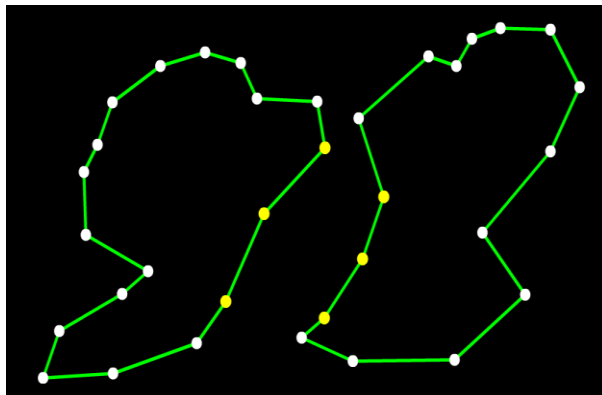
### 4.6.1 Closed Contours

Case 1:

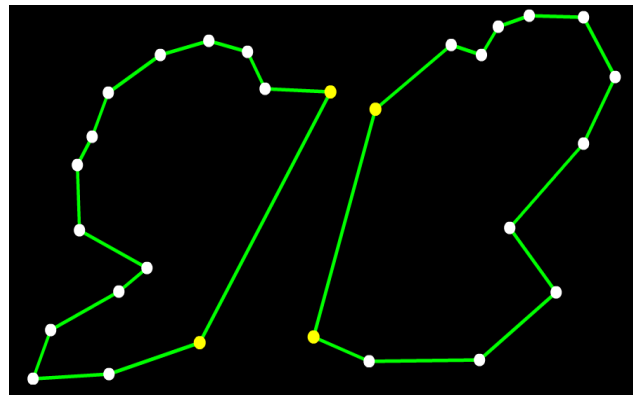
Step 1: Delete the points. Figure 4.26(a) shows the vertices to be deleted marked as yellow.

Step 2: Join the objects. Figure 4.26(b) shows selected vertices in yellow. The two contours are joined along these vertices. Figure 4.26(c) shows the figure after joining of two contours.

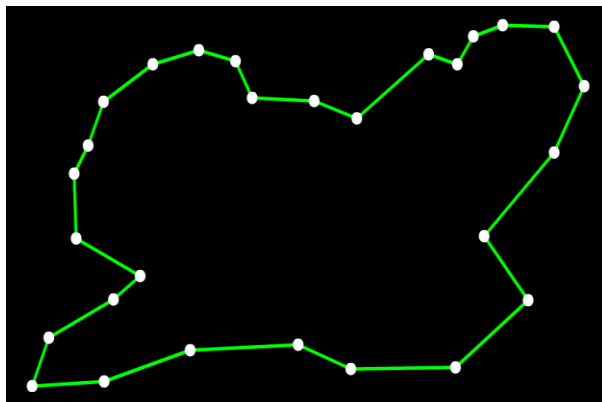
Step 3: Add vertices to get final Figure. Figure 4.26(d) shows the final figure after insertion of new vertices. The vertices marked in yellow are the new vertices added.



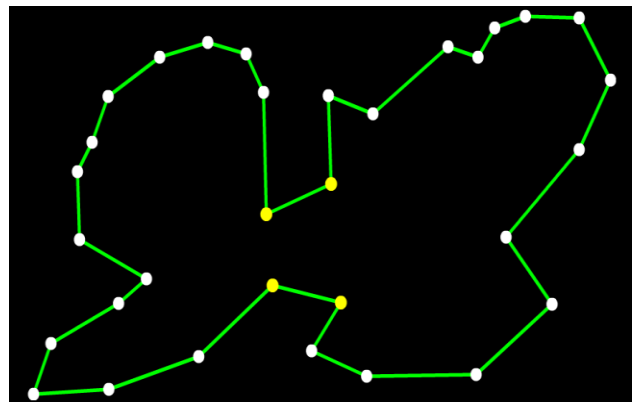
(a) Initial contours



(b) Contours after deletion of vertices



(c) Joining two contours



(d) Single contour after insertion of new vertices

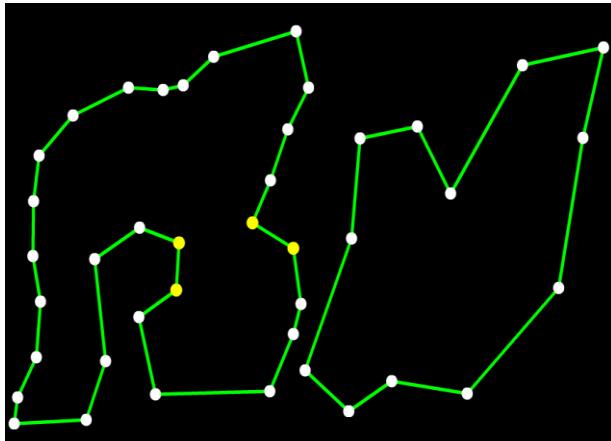
Figure 4.26 Closed contours- case 1

Case2:

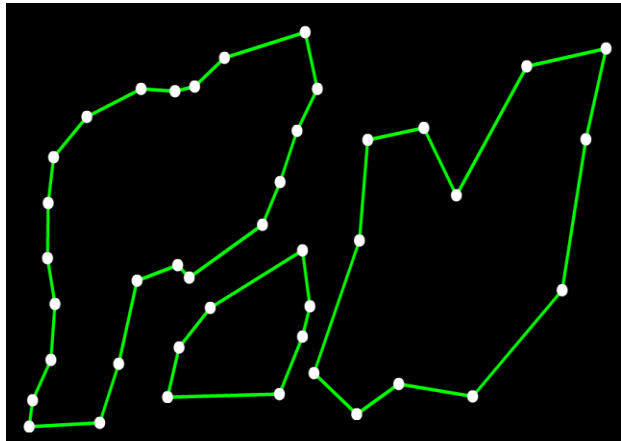
Step 1: Split object 1. Figure 4.27(a) shows 2 pair of vertices along which the first contour can be split

Step 2: Delete object 3. Figure 4.27(b) shows a new contour to be deleted. Figure 4.27(c) shows the contours after deletion of unwanted contour and two pair of points along which contours can be joined

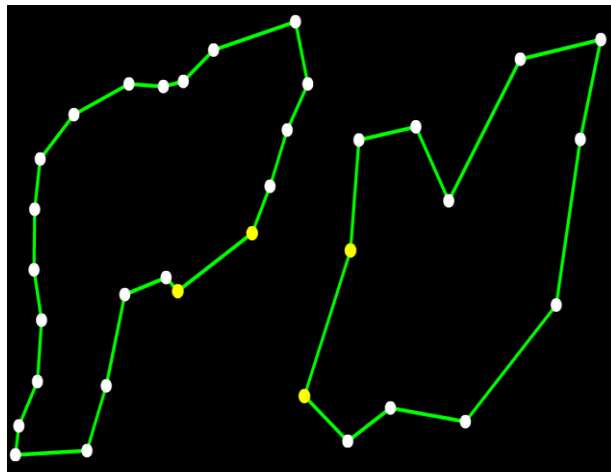
Step 3: Join remaining contour. Figure 4.27(d) shows the final contour after merging



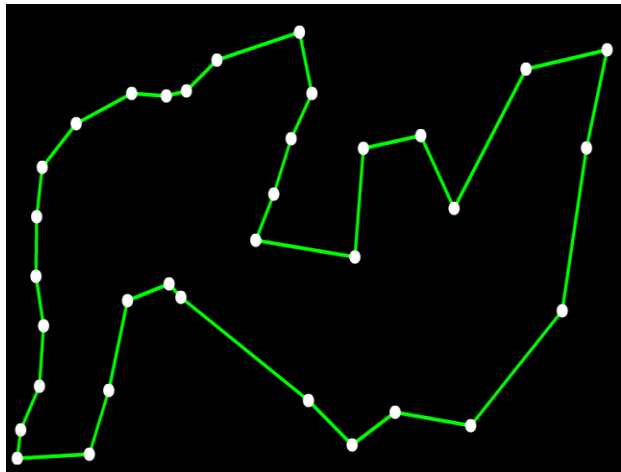
(a) Original Contours



(b) Contours after splitting of object 1



(c) Contours after deleting new contour



(d) Single contour after merging

Figure 4.27 Closed contours- case 2



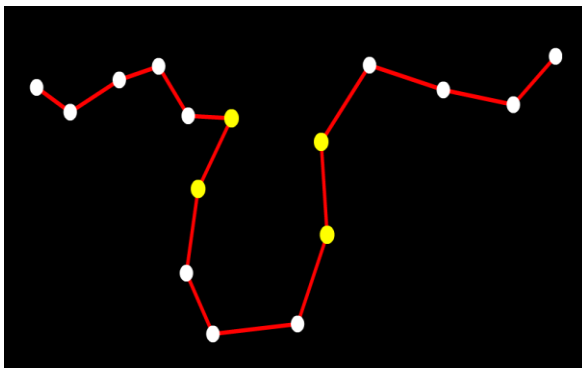
## 4.6.2 Segments

Case 1:

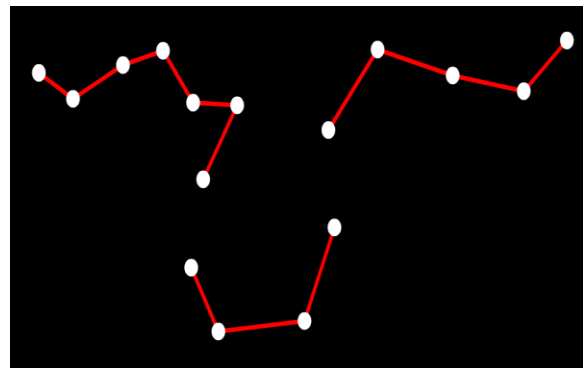
Step 1: Split object 1 into three different objects by using split operation. Figure 4.28(a) shows two pairs of vertices along which the segment has to be split. Figure 4.28(b) shows segments after split

Step2: Delete the new segment at bottom. Figure 4.27(b) shows the segments after deleting of the bottom segment. It also shows the vertices along which the segments canbe joined

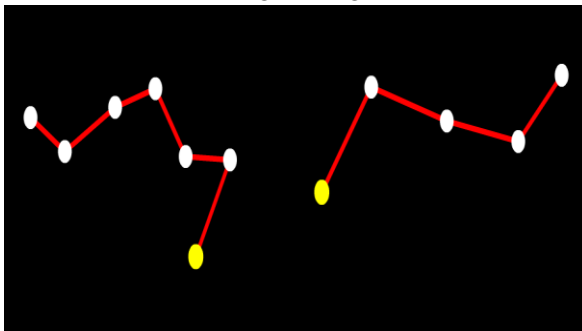
Step3: Merge the two segments. Figure 4.28(d) shows the final figure after merging.



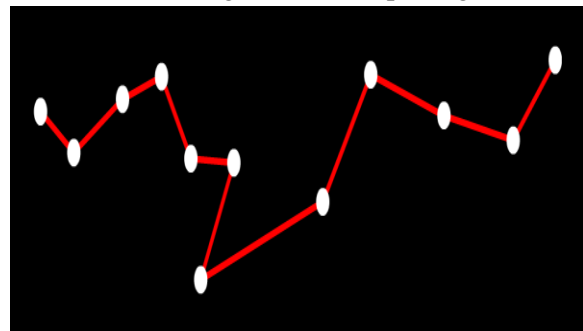
(a) Original segment



(b) Segments after splitting



(c) Segments after deleting new segment



(d) Final figure after merging of segments

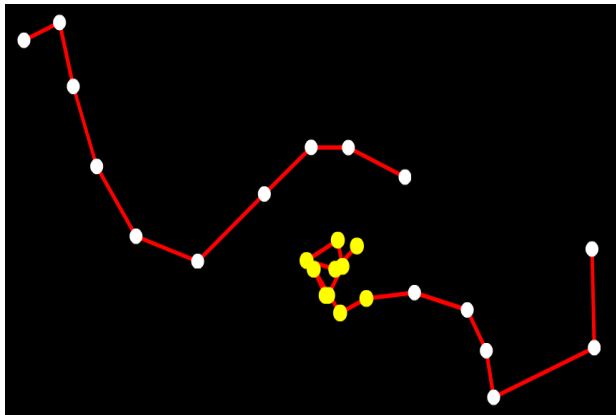
Figure 4.28 Segments- case 1

Case 2:

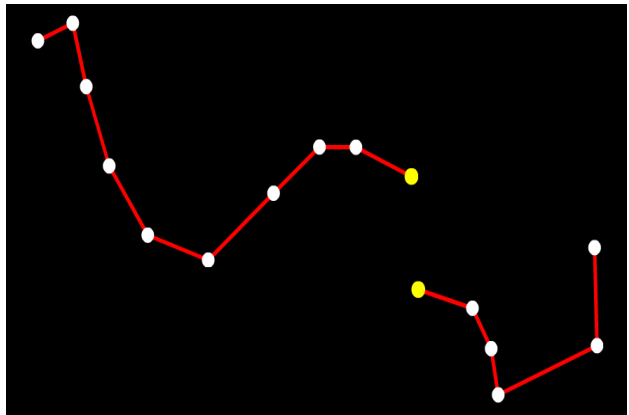
Step 1: Delete unwanted vertices. Figure 4.29(a) shows the vertex to be deleted marked as yellow. Figure 4.29(d) shows the segments after deletion of vertices and also the points along which the segment have to be merged marked as yellow.

Step 2: Merge the segments. Figure 4.29(c) shows the segments after merging

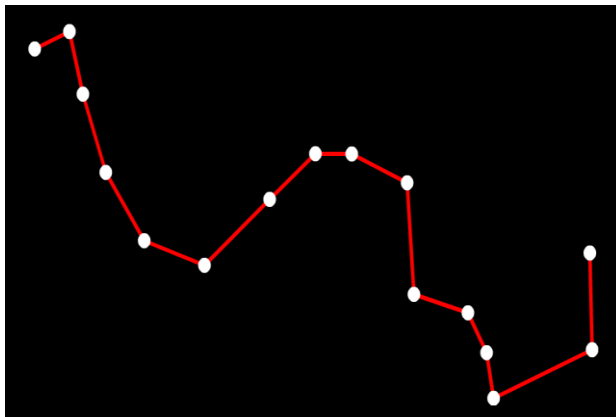
Step 3: Insert new vertices. Figure 4.29 (d) shows the single segment after insertion of new vertices.



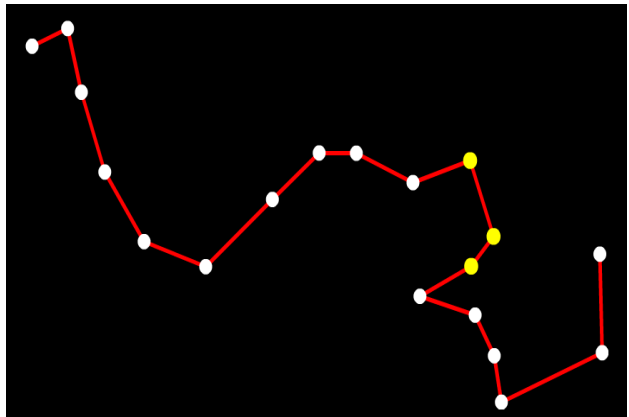
(a) Original segments



(b) Segments after deletion of vertices



(c) Segment after merging



(d) Final segment after adding of vertices

Figure 4.29 Segments- case 2

## 4.7 Protection and Editing Mode

Dragging of an object in FireFly was handled by left mouse drag event. The users would often drag objects accidentally, and correcting it was time consuming. Various solutions were suggested to fix the above-mentioned problem and the final solution chosen was to use ‘Shift+LMB (Left Mouse Button) to move objects.’

Dragging LMB has always been used to move the image around in the workspace. Being an essential feature this couldn’t be overwritten.

Several other options were taken into consideration:

- a. Scrolling Mode- A separate mode for moving objects. The user had to continuously switch between the drawing and scrolling modes. In order to draw and edit at the same time, the user had to make multiple switches using shortcut keys and mouse events. A larger programming effort was required to implement this. The user still required a lot of mouse motion to switch between modes, which made this method inefficient.
- b. Ctrl-P to translate polygon- the user would use this shortcut to enter into polygon mode to move a polygon. A polygon can be moved by a simple LMB drag but at the same time it would stop other LMB drag options for the image, other objects, and edit points. This method also required multiple mode changes, which didn’t solve the problem completely.
- c. Ctrl + LMB drag – A general solution to move all objects in FireFly. This way LMB-drag will move the image; LMB-click can select the object but will not move the drawn objects accidentally. Ctrl + LMB drag can move the object and not image.

‘Ctrl’ key in Windows doesn’t map the same in Mac machines. Thus, ‘Ctrl’ had to be replaced by the ‘Shift’ key. Finally, ‘Shift + LMB’ drag was chosen as default option for moving objects.

# Chapter 5

## Web-based Supervised Image Segmentation

### 5.1 MATLAB Interfacing

Microvasculature analysis required a tool that could not only be used for editing and marking in vessels images but also can be linked with MATLAB to provide a quick interface for analysis. Hence, we developed a semi-computerized Dura mater laminae analysis system [25] for fluorescence microscopy images of brain tissues. Figure 5.1 shows vessels image in Firefly.

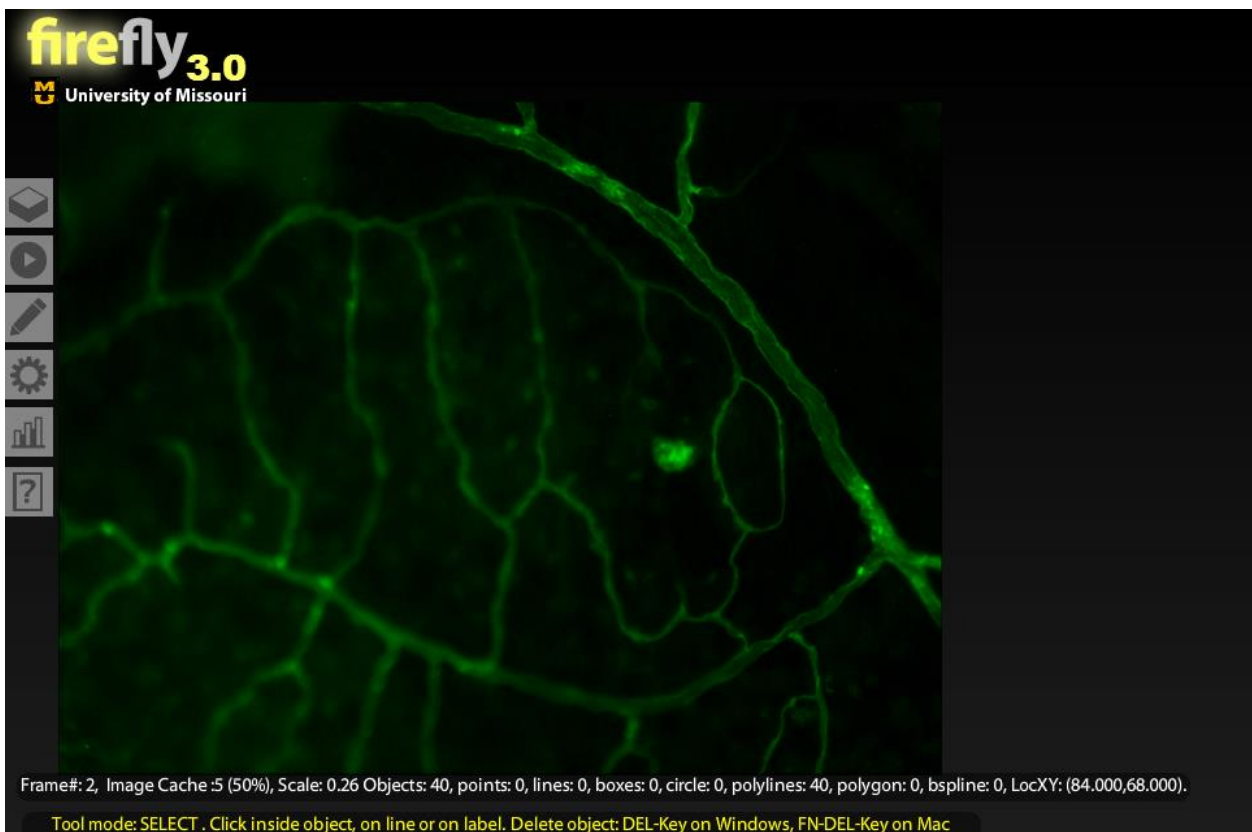


Figure 5.1 Vessels image

Processing images with MATLAB on the local system requires heavy memory and usage space on the disk, which makes the system slow. On the other hand running MATLAB and processing the image on the server would make the process faster and prevent wastage of local memory.

One way was to run the algorithm separately manually and load the data in the database through PHP controllers, which would have been done on developer's side and user would be notified after everything was done. User won't have any control and would totally depend on developer every time. Bridging the gap between the client side and MATLAB was necessary as it would provide flexibility and ease of execution directly from the client side. Also, the user would be having the control over MATLAB executables.

CGI calls are sent in Firefly through the client UI which is written in Flex. The user has freedom to select and process any image. The client machine is free from this execution process as processing occurs on the server. The results produced are loaded in FireFly. Simultaneously, the results are transferred through HTTP to the client side and displayed on the interface which can further be edited and processed by the user.

#### 5.1.1 Architecture of CGI

Our web application is based on Apache server with a Linux background. With the PERL CGI interfacing we could add a new module within the existing architecture without disturbing any frameworks on the client or server side. The HTTP calls are sent through Flex on the client side. On the server side two layered scripting is used to run the executables. Flex calls CGI PERL script, which calls another shell script and the shell script calls MATLAB executables. The data is exchanged through query strings through HTTP. The data between PERL, Shell script and MATLAB is interchanged using text files as shown in Figure 5.2. This result from MATLAB executable is further read by PHP controller and saved into database and also sent to the client side for visualization simultaneously.

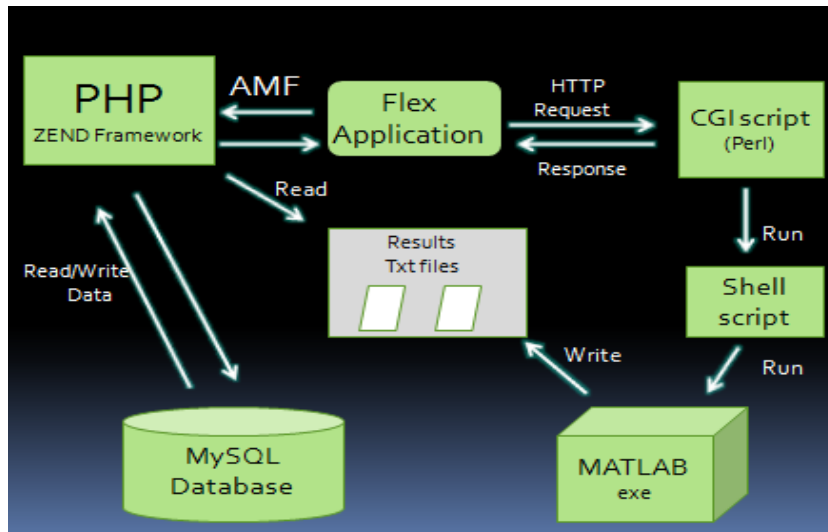


Figure 5.2 Architecture of CGI

### 5.1.2 Study of Different Technologies

Web developers have variety of ways to exchange the data dynamically over the web. These methods involve writing little scripts in one of the scripting languages. These scripts differ in two ways:

1. The location of the script
2. The execution of the script

The following table summarizes different technologies:

Method	Script Location	Script Execution	File extension
CGI	CGI-bin folder on the server	Server	.cgi
PHP, ASP	HTML	Server	.php, .asp
JavaScript	HTML	On the client side by the browser	n/a
Java (Servlets)	Server	On the client side	n/a

Table 5.1 Comparison of different technologies [28]

The script code for PHP and ASP are embedded in the HTML code. The source code is executed at the server and the result is pasted at the same place of the script when it is transmitted to the client.

The process is somewhat similar with CGI [28] scripts also. The server sees the tag with the CGI script; it executes the server-located script file. It pastes the result at the place of the tag and then sends this file. A common example is a hit counter script. The execution of the script will be incrementing the counter and the result are put in the HTML and sent on the client side and displayed at the place where SSI directive tag was located.

Browser provides an option to enable or disable JavaScript and Java by radio buttons or check boxes. But this isn't true for PHP or CGI. The reason behind this is as JavaScript and Java are executed on the browser (or not, if it is disabled). Browser has no idea about PHP or CGI. It just gets pure HTML.

Developers working with “frontend” web pages and who are accessing backend databases normally use the embedded script method (PHP, ColdFusion, ASP (i.e. client/server Web applications)). Our application does something similar, it's based on Client/Server model. Due to certain security and cross browser calling issues, running scripts through PHP services became difficult. FireFly is based on two different technologies. The flex on the client side calls PHP script. The HTML page consists of SWF file rather than PHP script. Using CGI became a big advantage as scripts are stored in separate files. Scripts can be stored separately in CGI-bin folder and can be executed just by a GET request.

### 5.1.3 CGI (Common Gateway Interfacing) through Flex in FireFly

Due to our high interactive requirement there are continuous calls to server. By using URL Loader class we were able to handle the requests efficiently on the flex side. The parameters were interchanged through query string and the call were sent to the server by URL loader method.

```
V = new URLLoader();
```

```
var vr:URLRequest=new
URLRequest("http://FireFly.cs.missouri.edu/services/public/index.php/do?frameNumber="+frameBuffer.currentFrame+"&annotationID="+annotationID);
```

#### 5.1.4 PERL Scripting Language for CGI

Common Gateway Interface (CGI) is a method to generate web content by executable files. It is mostly written in a scripting language. Practical Extraction and Report Language (PERL), which was developed by Larry Wall, has been used in FireFly to run the MATLAB executables. PERL can be run on variety of platforms, such as Windows, Mac OS, and UNIX. Large projects written in PERL include cPanel, Slash, Bugzilla, RT, TWiki, and Movable Type. PERL as a CGI scripting language became very popular and some of the high-traffic websites that use PERL extensively include Amazon.com, bbc.co.uk, Priceline.com, Craigslist, IMDB, LiveJournal, Slashdot and Ticketmaster. It is also an optional component of the popular LAMP technology stack for web development, in lieu of PHP or Python.

#### 5.1.5 MATLAB Executables and MCR

Using Math Works application deployment products, the recoding of MATLAB algorithms can be avoided. This reduces errors and maintenance becomes easier. As you develop in MATLAB you can easily develop and deploy an executable, which can be standalone or software components. MATLAB builder products work with MATLAB Compiler to create components for standard use with Java, .NET, or Excel. The components can be deployed easily on operating systems, which are supported by MATLAB.

The MATLAB Compiler Runtime (MCR) is a set of libraries, which helps the MATLAB standalone executables to run on machines that do not have MATLAB, installed. MATLAB, MATLAB Compiler, and the MCR together enable you to create applications or software components quickly and securely. Since we just needed an environment that can run the executables, MCR was installed in the CGI bin folder.



### 5.1.6 Invoking Shell Script and MATLAB Executables from the GUI

The user interface hides the complications on the server side and it displays a message “Running Analysis”, which indicates the script is running on the server. Figure 5.3 shows the message when the user clicks Run Analysis button in the Segmentation Panel. When the user clicks, the request is sent to the CGI script in CGI-bin. CGI script runs the shell script which executes the MATLAB executable. The image is processed and segmentation results are written on server which is later fetched on client side for visualization.

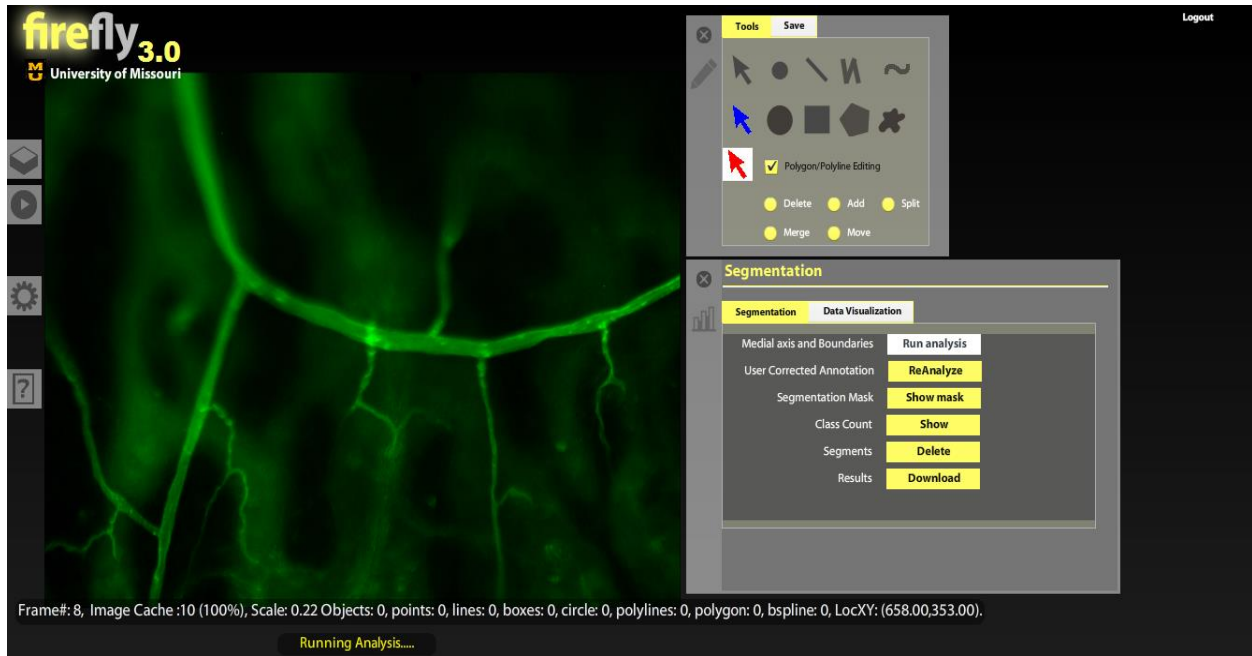
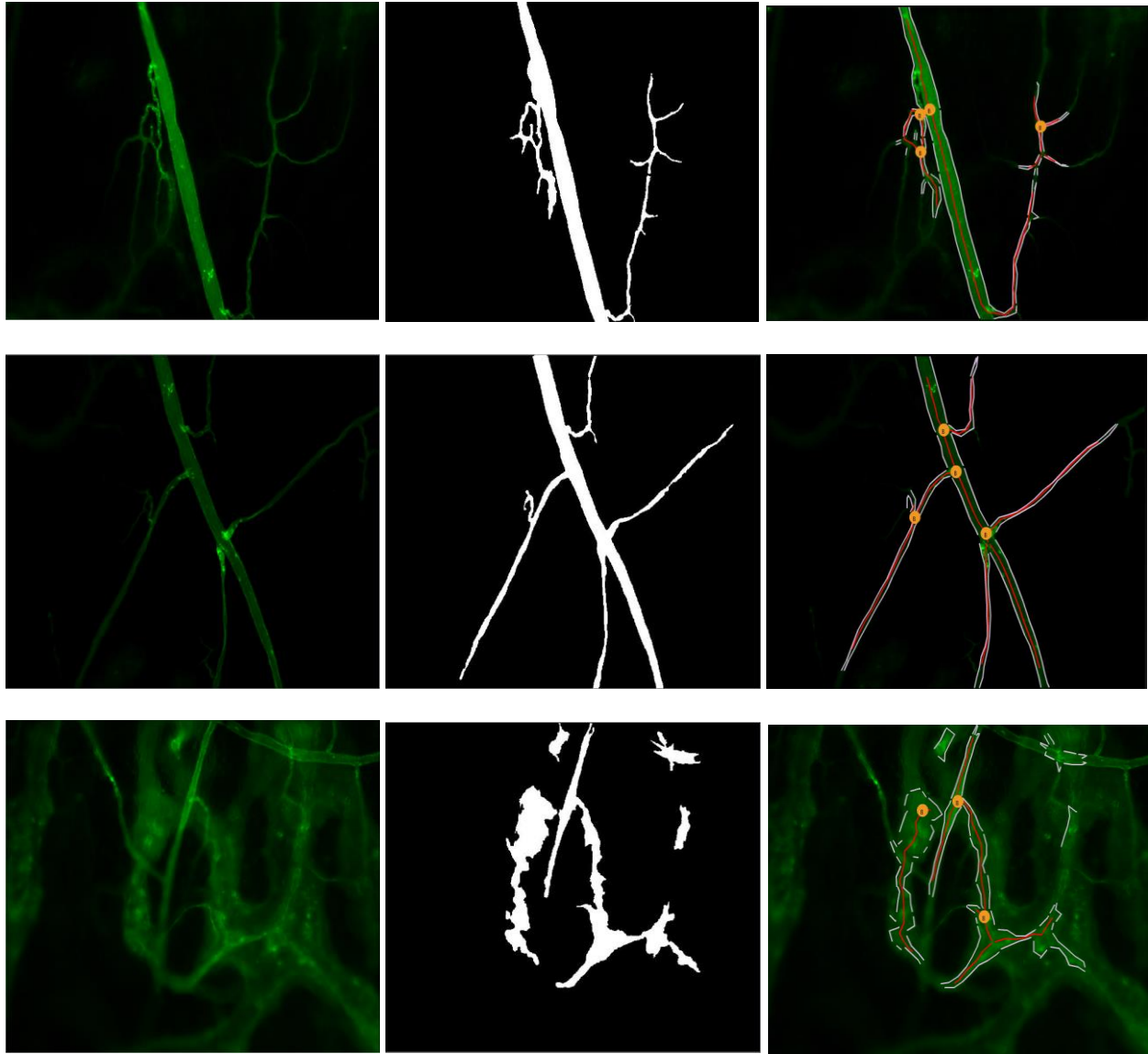


Figure 5.3 Running MATLAB executable in FireFly

## 5.2 Segmentation Results

Currently Firefly uses the segmentation described in [27]. Figure 5.4 shows the segmentation results for some of the Vessels images. When the MATLAB executable is called the algorithm produces a black and white segmentation mask as a result of processing. Figure 5.5(a) shows the original vessels images. Figure 5.4(b) shows the segmentation mask for the vessel images. The masks are further processed to extract the graphs; these graphs are shown in Figure 5.4(c) which are in the form Medial Axis, Boundaries and Bifurcation Points. These graphs are represented in FireFly with the help of Polylines and point objects.



(a) Original images

(b) Segmentation masks

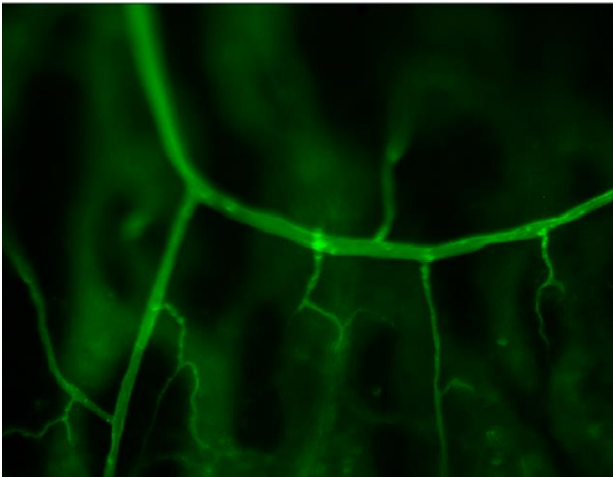
(c) Vessels graph

Figure 5.4 Segmentation results

### 5.3 Alpha Blending of Images

In the context of imaging, alpha values signify transparency with values ranging from 0 to 1, with 1 being opaque and 0 being transparent. Alpha channels carry transparency information for each image.

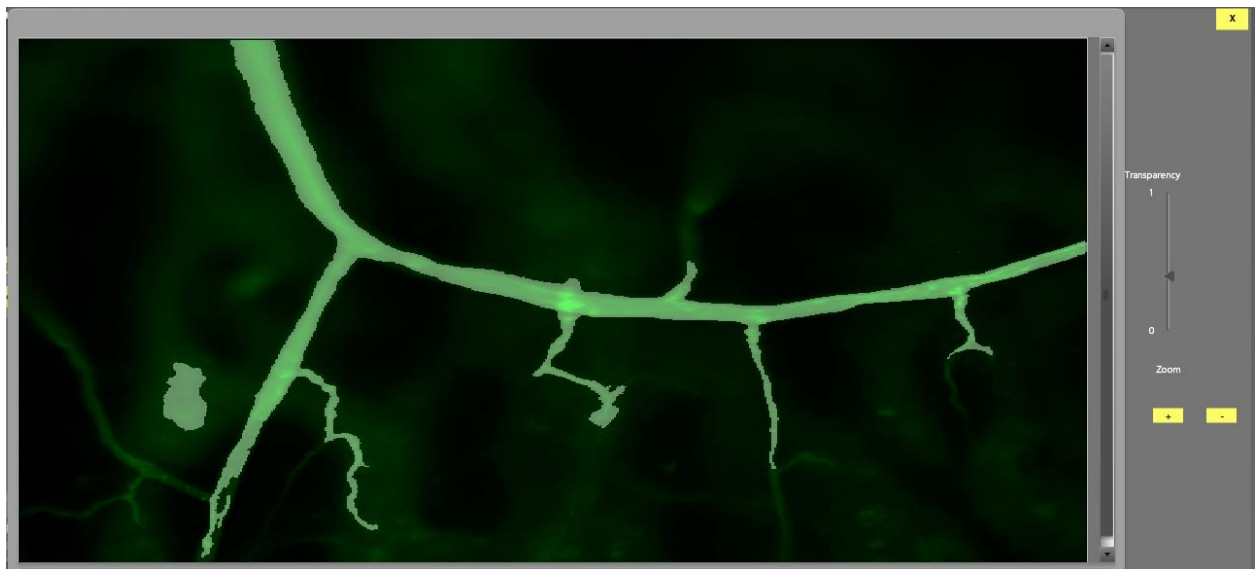
The segmentation mask generated from MATLAB are mostly black and white images. To test the accuracy, a new window containing the actual image and result image can be opened from the Data Analytics panel. This window contains the mask placed on top of the original image. As shown in Figure 5.5 the two images are fused together and displayed in a single panel. The vertical scroll bar on the right can be used to change the alpha value of mask, which changes the transparency of the mask. In Figure 5.5(c), the result image is shown with an alpha between 0 and 1.



(a) Original image



(b) Segmentation mask



(c) Alpha blending panel

Figure 5.5 Alpha blending

## 5.4 Loading Data through PHP Controllers

Loading of data is done through PHP controllers. The data after processing of vessels images contain three different kinds of objects: Boundary, Medial Axis and Bifurcation Points. Boundaries, medial axis are represented by polyline and Bifurcation Points are displayed using point objects. A separate controller was written to read this data for vascular images.

ReadSegController reads each line from the text files generated and inserts into database. Once the results are loaded, the response handler refreshes the screen buffer with new objects. The user can perform various editing operations on the results. Once a user has completed the editing process, another MATLAB executable can be called through GUI. By clicking on ReAnalyze button, analysis could be run on server on the existing segmentation results. This executable generates values for Tortuosity, Angles and Curvature which is then visualized in form of graphs as shown in Figure 5.7.

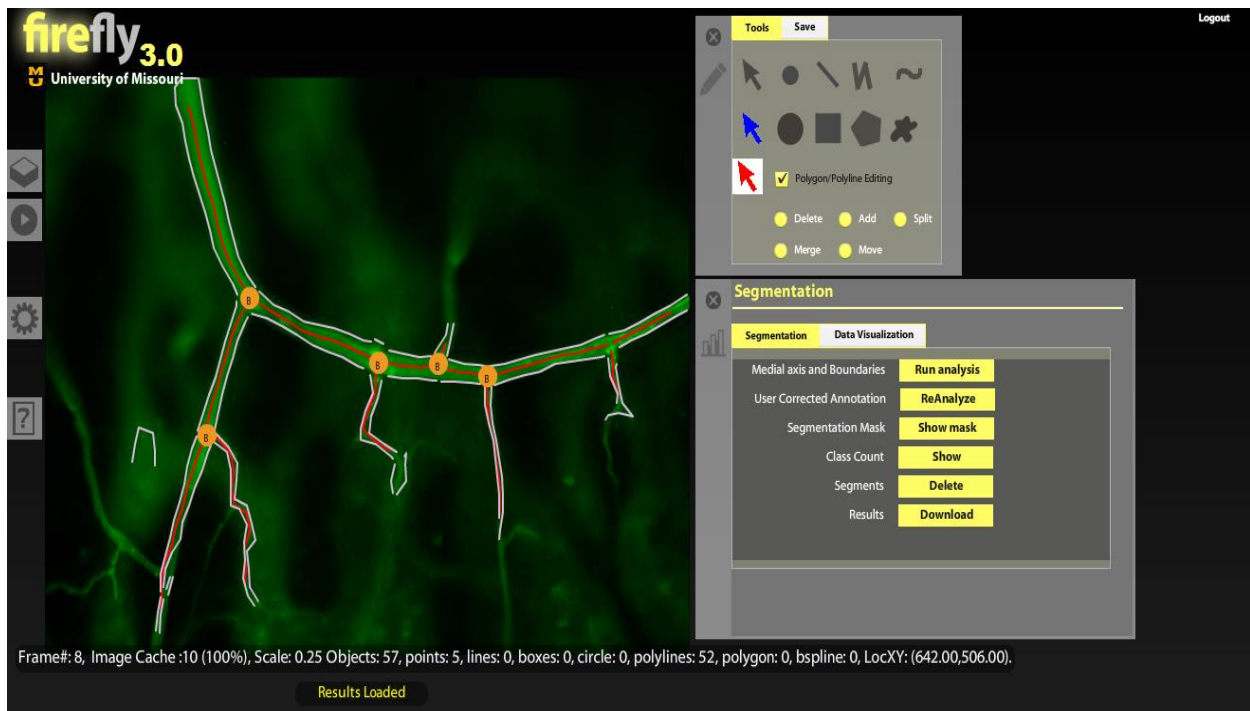


Figure 5.6 Image with processing result

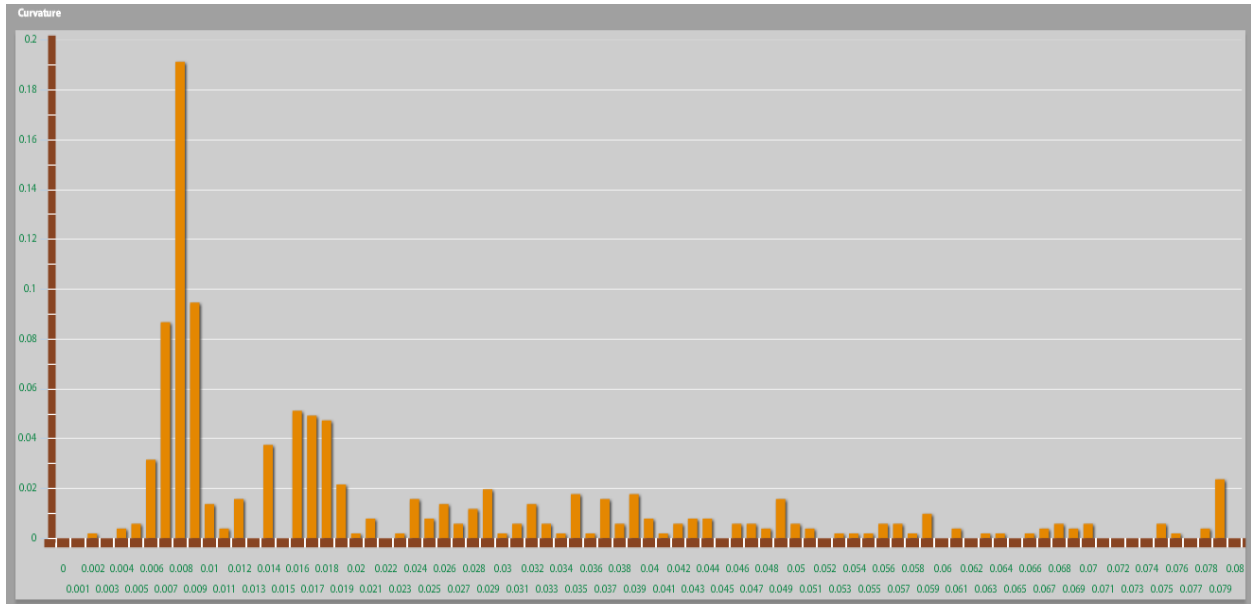


Figure 5.7 Graph analysis

# Chapter 6

## Data Analytics and Visualization

Data Visualization is a new and promising field in Computer Science. It uses Computer graphics to show the patterns, trends, and relationships among the data sets. With the evolution and collection of huge data sets, there is always a need of some novel way to represent and visualize the data. Due to variations in the data sets, a data specific approach is always important. Visualization is graphically presenting the data that helps in analysis of the information in a better way. It's a process by which, we transform numbers, relations, concepts, processes etc. into a visual form, which can be perceived and understood by human eyes. By visualization, we emphasize on the relation between the objects rather than a single data value. In the words of Friedman (2008) the "main goal of data visualization is to communicate information clearly and effectively through graphical means. It doesn't mean that data visualization needs to look boring to be functional or extremely sophisticated to look beautiful. To convey ideas effectively, both aesthetic form and functionality need to go hand in hand, providing insights into a rather sparse and complex data set by communicating its key-aspects in a more intuitive way. Yet designers often fail to achieve a balance between form and function, creating gorgeous data visualizations which fail to serve their main purpose to communicate information". Effectiveness and expressiveness are the two main criteria by which we can analyze our visualization. If we are able to express our relations clearly then we are effective. With new technologies we use animations and images to visualize the big data sets which were not effectively visualized by simple graphs.

Need for data visualization?

Human perception is always better when we see a picture rather than big numbers. From a well-drawn picture, it is much easier to find the trends and relations. Data Visualization takes the load from numbers to pictures, which not only saves time but also helps in making quick decisions. With emerging new researches we have new and even bigger data sets so we cannot use the old traditional graphs but would have to come up with some new interactive way of visualizing the data sets.

## Problem

One of the biggest challenges was to use a technique based on the client server model. Visualization had to be performed on the client side, and data had to be fetched from the server side. The visualization had to be intuitive and also dynamic in nature. There were various toolkits and online kits available, which were explored and some of them are discussed below.

### 6.1 Study of Toolkits and Online kits

#### A. Google Chart API [13]

Good Chart API, a tool that offers visualization for dynamic charts is very robust. It works well with any browser that supports SVG and VML. The API supports just about any line graphs, bar charts, maps and even QR codes. It is a good tool for someone who is not looking for a lot customization and is comfortable with using the Google 'look'. However, since it is generated on the client side, it often creates problems for devices that do not support JavaScript.

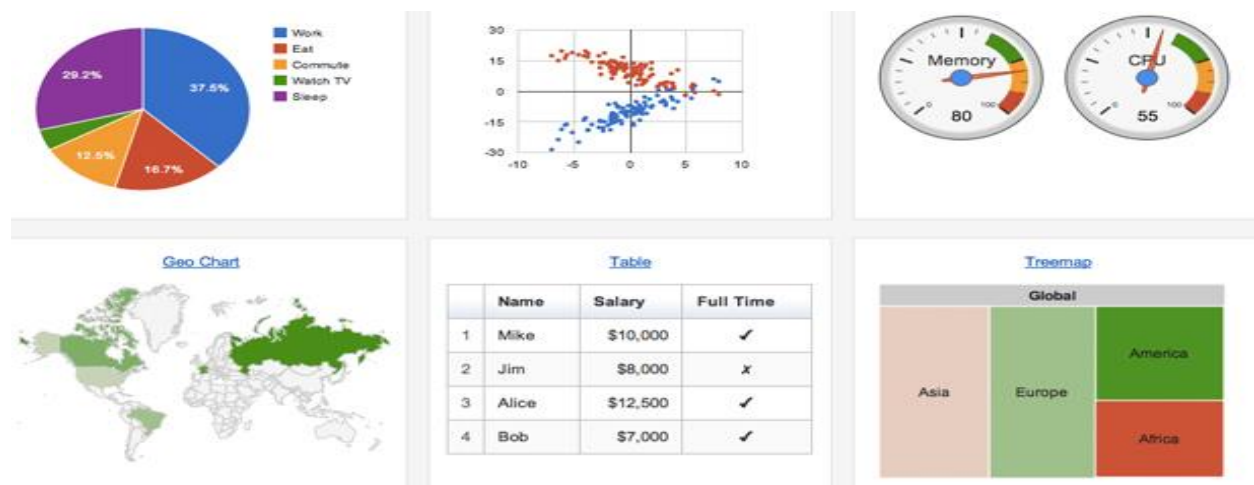


Figure 6.1 Google chart API [13]

#### B. FLOT [14]

Another tool with a great library for line graphs and bar charts is FLOT. FLOT works well with all browsers that support canvas. A nice feature about FLOT is that you have access to plenty of callback functions, so you can easily run your own code and style the results when readers hover, click, mouse out etc. This feature gives much more flexibility than other charting packages;

however, there is a steeper learning curve. FLOT is a jQuery library, and if you're already familiar with jQuery, callbacks, styling and behavior of the graphics can be easily manipulated.

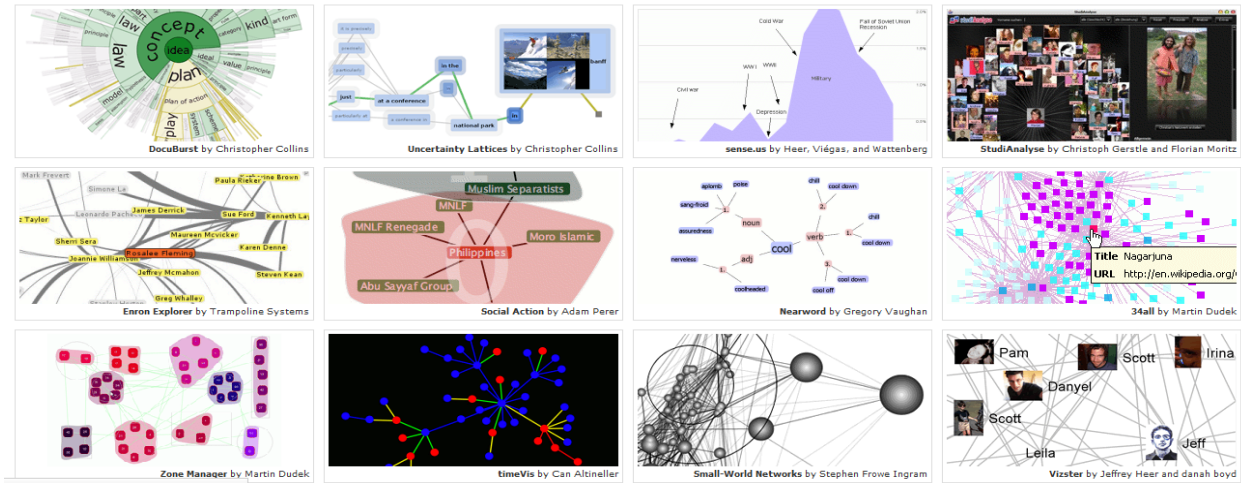


Figure 6.2 FLOT [14]

### C. Raphael [15]

Another great library for creating charts and graphs is Raphael.

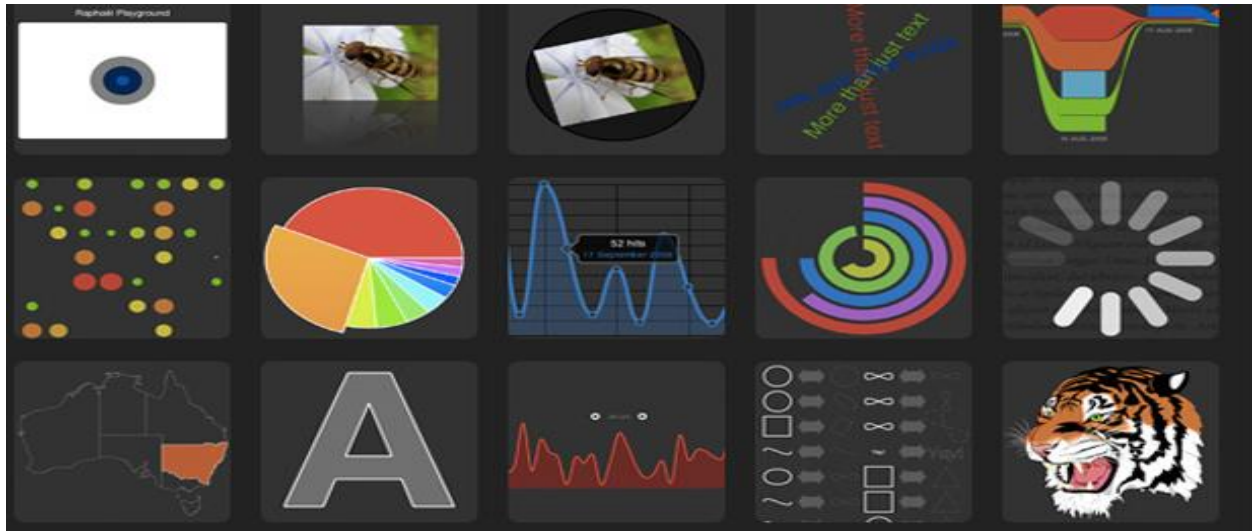


Figure 6.3 Raphael [15]

Unlike other libraries, Raphael focuses on SVG and VML as output, which has its own pros and cons. Since, SVG is a vector format, the results look great at any resolution; however, it creates a DOM node for each element and can be slower than creating rasterized images via canvas. The advantage of using Raphael is that one can easily interact with each DOM element and attach



events, just like HTML. There are plenty of demos available on the website to demonstrate how Raphael can create common charts and graphs and since it can also render arbitrary SVG, it has the ability to create some very complex visualization as well.

#### D. D3JS [16]

(Data-Driven Documents) is a JavaScript library that supports SVG rendering. D3 is a great tool to create simple to complex visualizations such as Voronoi diagrams, tree maps, word clouds, circular clusters to name a few.

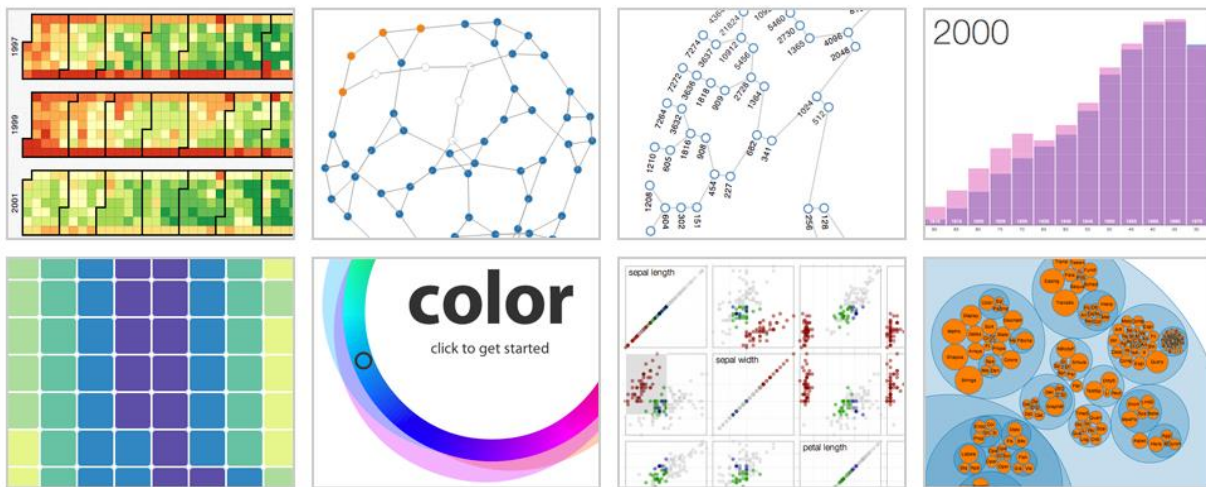


Figure 6.4 D3JS [16]

#### E. Flare [17]

Flare is a tool that runs on Adobe Flash Player and is an Action Script library, which is used to create visualizations from basic charts, complex interactive graphics, visual encoding, animation etc. Another benefit of using Flare is that it features modular design that enables the developer to customize visualization techniques.

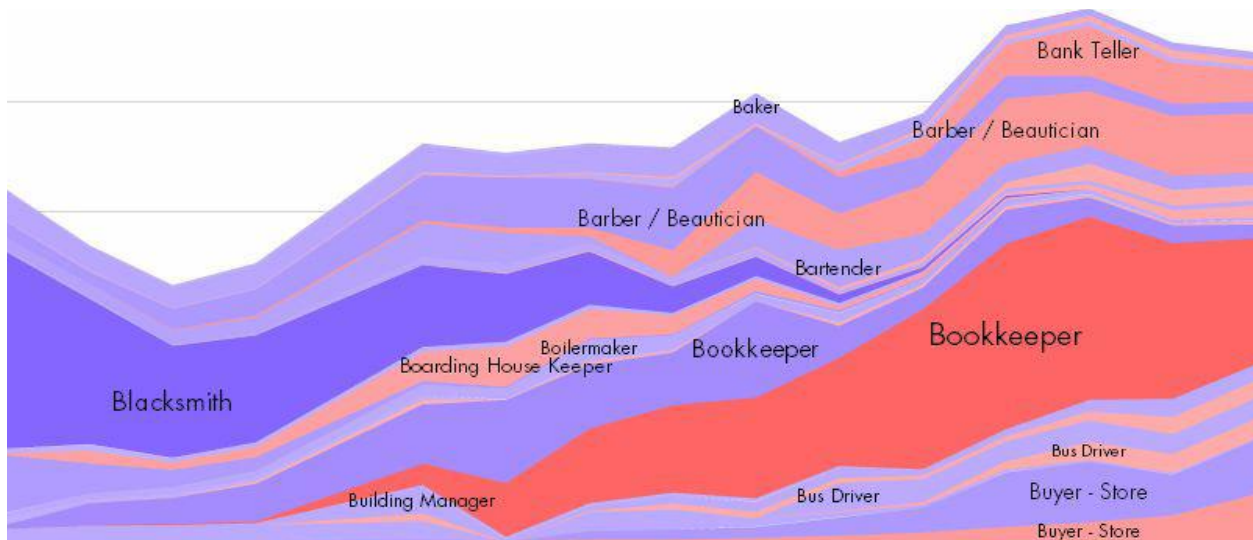


Figure 6.5 Flare [17]

## 6.2 Data Visualization in Flex 3

The main purpose of Data visualization is to enable the user to present data in such a way that it not only simplifies but also makes room for data interpretation and data relationship. A type of data visualization, which creates two-dimensional representations of data, is charting. Flex supports the most common 2-D charting types. Flex also allows the user to have a great control over the appearance of the charts as well. Flex provides some basic charts and with the classes it provides we can convert these graphs to an interactive graph.

Single data series can be shown on a simple chart, where a series is a group of related data points. Flex allows you to not only create chart types but also customize them. The controls to customize are located in the `mx.charts` package. Cartesian charts are charts that typically represent a set of data points in 2-D space in rectangular form. All chart controls are subclasses of Cartesian Chart class with the exception of Pie Chart class. Pie Chart class on the other hand is a subclass of the Polar Chart class, which represents the data points in the circular space.

## Events and Effects of Chart [18]

When a user clicks/double clicks on a chart control, a chart event is triggered. These events are of type Chart Event. The Chart Event is part of the charts package, which are imported in the appropriate classes into the mx.charts.events package.

You can select the data points by either writing a program or by the following ways described below-

1. Mouse selection: By hovering the mouse pointer and clicking on the left mouse button over a data point.
2. Keyboard selection: Use the keys on the keyboard to select one or more data points.
3. Region selection: By drawing a rectangle on the chart, this defines the range, and selects all data points within that range.
4. Programmatic selection: Programmatically select one or more data points using the chart selection API.

All chart controls inherit basic charting characteristics from the Chart Base class.

<b>Chart type</b>	<b>Chart control class</b>	<b>Chart series class</b>
Area	AreaChart	AreaSeries
Bar	BarChart	BarSeries
Bubble	BubbleChart	BubbleSeries
Candlestick	CandlestickChart	CandlestickSeries
Column	ColumnChart	ColumnSeries
HighLowOpenClose	HLOCChart	HLOCSeries
Line	LineChart	LineSeries
Pie	PieChart	PieSeries
Plot	PlotChart	PlotSeries

Table 6.1 Flex data visualization table [18]

## 6.3 Data Visualization in FireFly

### 6.3.1. Vessels Graph

Figure 6.6 shows the Curvature graph for Vessels. This chart is plotted by using columns chart in Flex 3. The text files generated by MATLAB are read and data is sent to the client side. These results are plotted using the column chart in mx.chart package. Vessels analysis contains Angles, Curvature and Tortuosity. Any of these options can be chosen form Data Visualization Tab. The value selected from drop down will generate the graph for that image as shown in Figure 6.6.

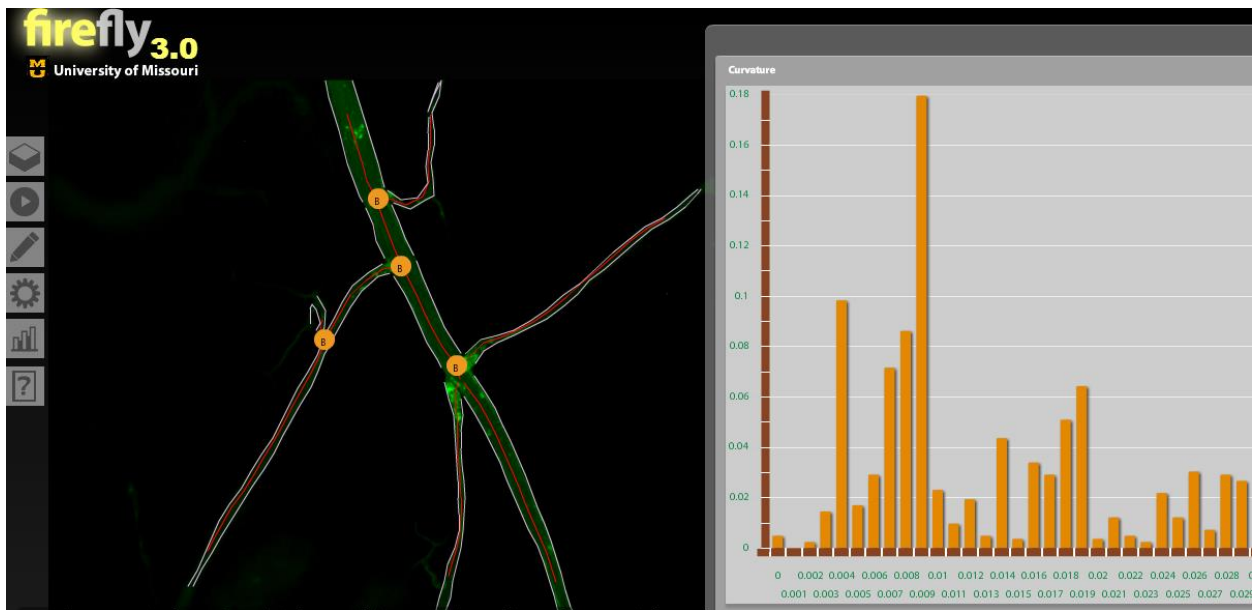


Figure 6.6 Vessels with curvature graph

### 6.3.2 Malaria Count Graph

Figure 6.7 shows Malaria Count Graph. This column chart shows the total number of objects in each class computed individually and represented in the form of columns. Malaria dataset contains four classes Parasitemic, Uninfected, Other and Parasite Outside Cell. The graph in Malaria dataset is used to visualize number of infected cells. This graph can also be utilized in other datasets to visualize total number of objects in each class for an image.

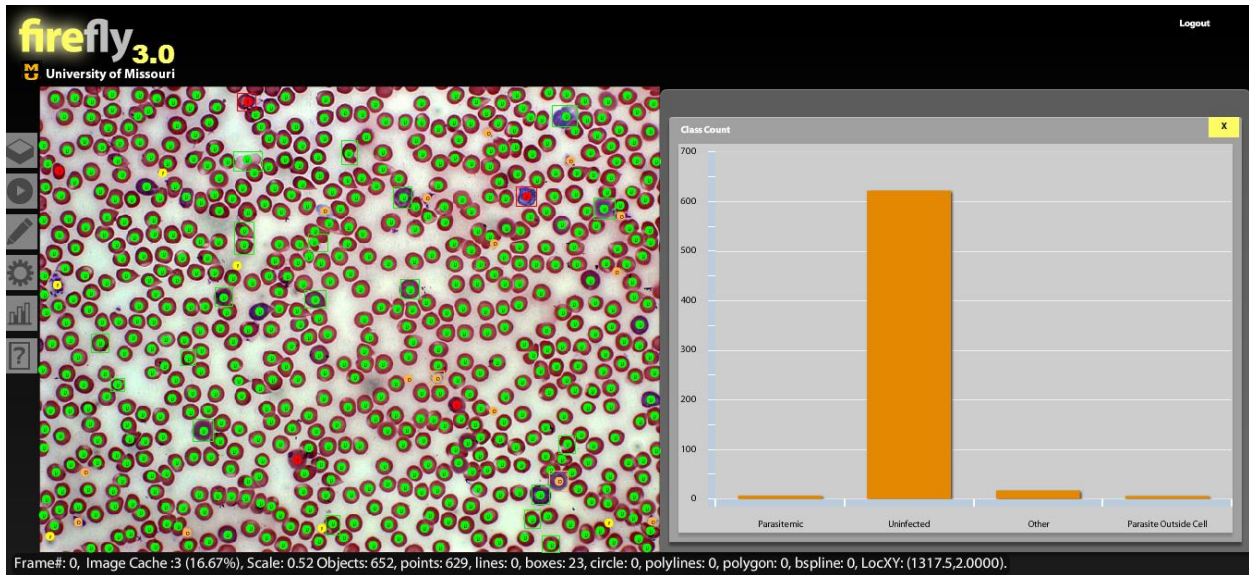


Figure 6.7 Malaria image with malaria count column chart

### 6.3.3 Data Grid and Scatter Plot

Even though there was a need for large number of features to visualize, graphical analysis linked with the numerical data was the main requirement. To do so, Scatter Plot in the Flex 3.0 with the data grid was used. Figure 6.8 shows how one can select any ID in the data grid and the corresponding values in the graph are highlighted. Due to different variations needed for the studies, six different plots had to be plotted. The Columns in the data grid are selectable and one click on the name sorts the whole grid according to that particular column. The points in the graphs are also selectable. By selecting any point, we can see the highlighted ID in the Data grid. This can help in finding the outliers in the graphs. We show different graphs with different axis values, the graphs automatically change its axis scale as we plot different values.

Currently this graph shows different parameter like center, x, y coordinates for a box. Our future work would be to extend this feature for all types of objects.

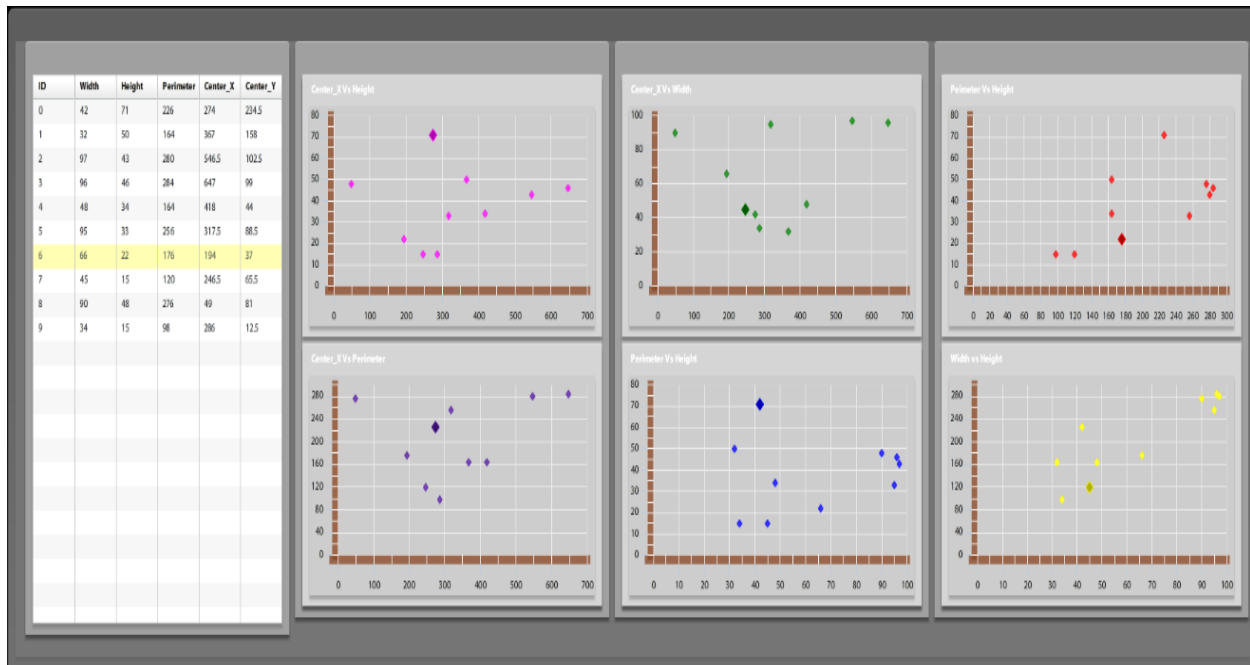


Figure 6.8 Data grid with scatter plot

### 6.3.4 Bar Graph with Pie Chart

Bar graph in combination with Pie chart helps us to identify the number and percentage at the same time. Figure 6.9 shows bar graph that can be converted to pie charts. The user would just click on the bar graph and drag it to the other panel to convert it into a pie chart. A Pie chart assists in determining the ratio of the particular value in comparison to other values. With the bar graph, features can be computed and visualized and by Pie chart is used to compare two or more features at once.

When the user points their mouse pointer on any of these bar graphs they can see the data tips which tell the values of that particular data point. Figure 6.9 shows an example of Heights Column. The user can select button in the Pie Chart if the user does not want to drag each one by one. The Pie Chart is also interactive if you click on one part that part moves out and you can see others as they were before.

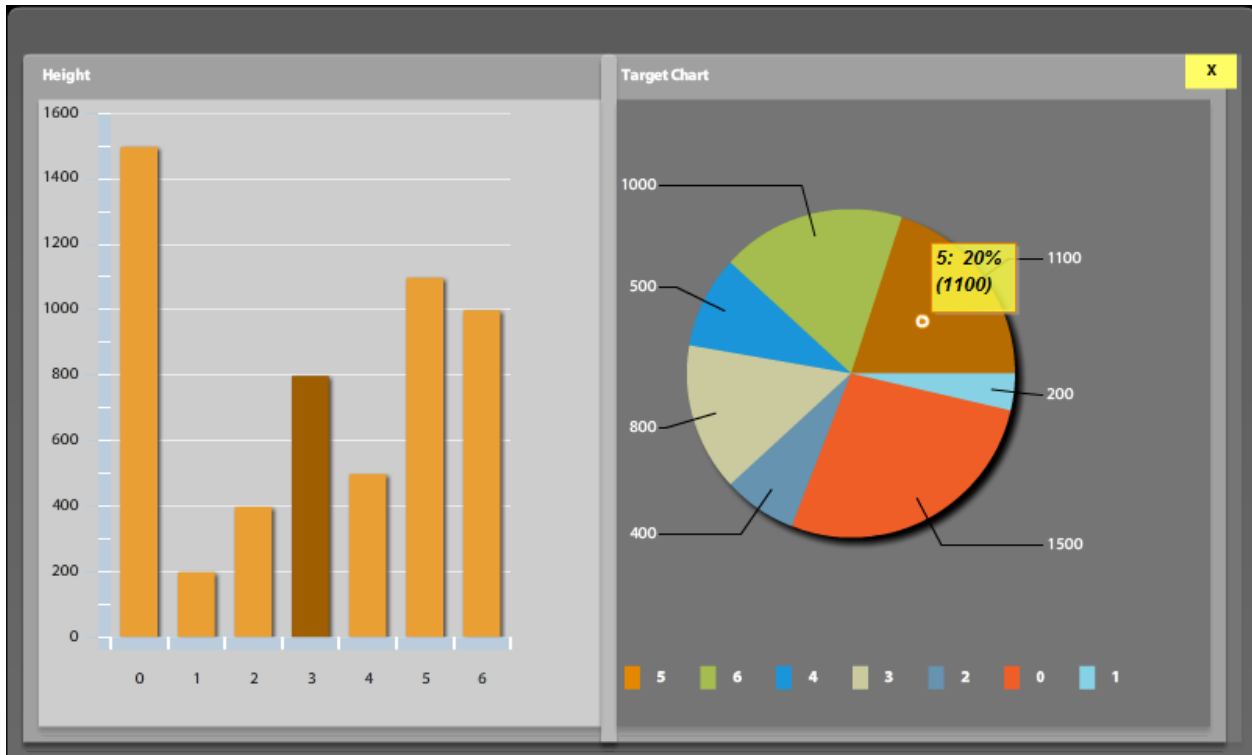


Figure 6.9 Bar graph and pie chart

### 6.3.5 Zoomable Line Chart

The variance of features along a timeline can be studied using this line chart that is zoom-able. The user can easily select an area to zoom. The axis adjusts dynamically depending on the zoom level. As shown in Figure 6.10 the square shows the area to be zoomed and Figure 6.11 shows the area zoomed.

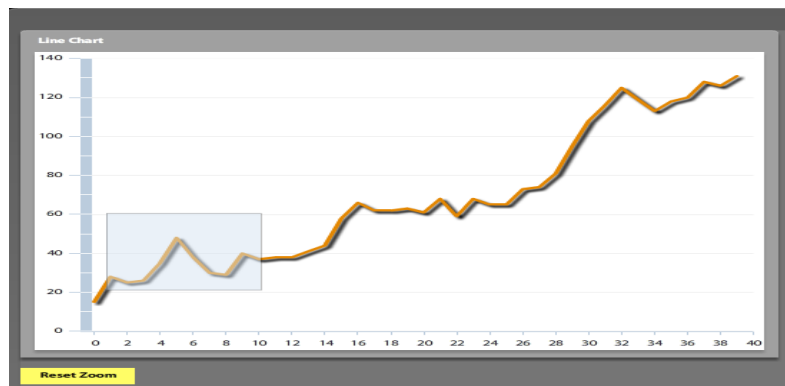


Figure 6.10 Zoomable line chart

This kind of approach not only helps to study the graph at macroscopic level but also at microscopic level. 'Reset Zoom' button will reset the zoom at the initial point and the user can again start selecting the portion of the line.

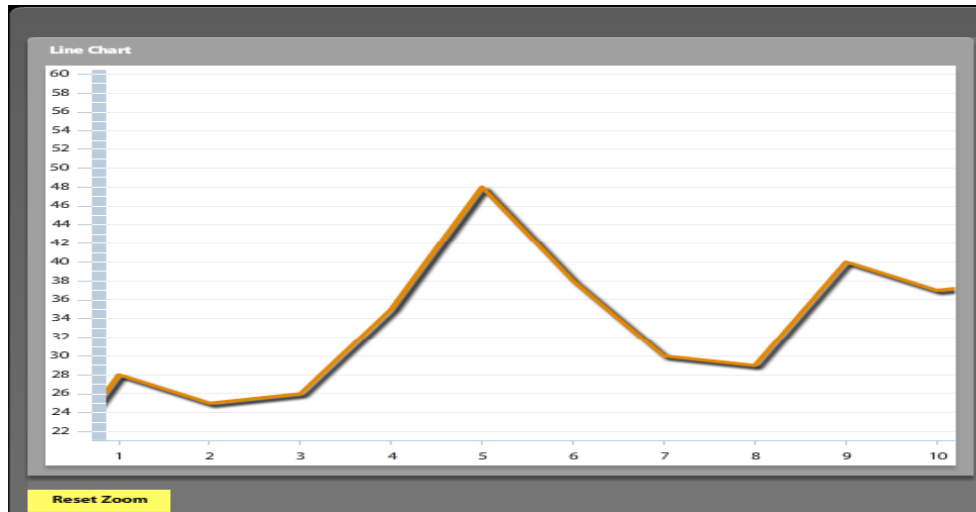


Figure 6.11 Zoomed portion of line chart



# Chapter 7

## Optimizing Web-Based Performance

### 7.1 Dataset Locks

FireFly uses a single database to support multiple users, which makes maintaining data coherency a challenge. In order to protect data from being overridden and to provide multiple users access to the same data set, data lock mechanism was introduced. When selecting an annotation, its lock status can be checked under the ‘Edit’ column.

AnnotationID	AnnotationTitle	Perm	Edit
7	UPS Filiz	R/W	Open

Figure 7.1 Annotation with R/W access

Figure 7.1 shows an example of an unlocked/open annotation. The ‘Perm’ column indicates the permission/security granted to the user once they enter the dataset. The ‘Perm’ value shown in Figure 7.1 indicates that the user has Read/Write (R/W) permissions, in other words, the user has editing capabilities. If the dataset is locked for editing, the user will get a ‘Read Only (R)’ permission status. Once the user that has locked the dataset for editing logs out, the lock is released in the database and other users can log in with an ‘R/W’ access.

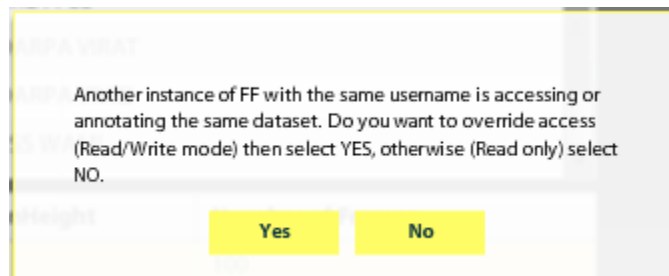


Figure 7.2 Alert box if another instance of same user is logged in

If one user hasn't logged out from the dataset and tries to log into the same dataset on the same machine with a different browser or on a different machine, an alert message is displayed. Figure 7.2 shows the alert message displayed to the user. An option is given to the user to log in 'Read/Write mode' or 'Read' only mode. This data lock mechanism has helped us maintain the integrity of our data and allowing several users to work successfully at once.

## 7.2 Internet Explorer vs. Other Browsers

### 7.2.1 Conflicts with Shortcut Keys

FireFly uses short keys to access certain features, which not only help reduce GUI space on the user screen but also provide alternate GUI options to the users. Some of these short cut keys have conflicts in Internet Explorer. However, FireFly runs as expected in other browsers such as Firefox, Chrome, Safari, etc. Table 8.1 lists the shortcut keys that conflict with Internet Explorer browser keys. These keys were selected since they relate to the meaning of the feature and it's easier for the user to remember.

Keyboard Short Keys	FireFly	Internet Explorer
F1	Debug info window	Help and support
Ctrl + T	Enable track mode	Opens new tab
Ctrl + J	Joining the tracks	View downloads history

Table 7.1 FireFly vs. Internet Explorer

The debug info window can now be accessed in Internet Explorer using 'Ctrl + F1' keyboard short cut. To enable/disable track mode a user can click on the blue arrow in the tools panel.

### 7.2.2 Cache in Internet Explorer

When a web service is accessed for the first time through Adobe Flex, the first set of data is received correctly, however, when another request is made to the same data, IE does not receive the most recent data. This issue has been identified to exist only in Internet Explorer and not in other browsers. The cause of the problem is that Internet Explorer maintains a cache of its web service results. When a request for data is sent via Flex, Internet Explorer tends to get it via the cache instead of requesting from the Web Service. A work around to this problem is to change the

settings of Internet Explorer. The user can access ‘website data settings’ via Internet Explorer settings. Figure 7.3 shows Website Data Settings in Internet Explorer with 4 available options:

- Every time I visit the webpage
- Every time I visit Internet Explorer
- Automatically
- Never

To successfully implement the work around and operate FireFly on Internet Explorer, the user must select ‘Every time I visit a webpage.’ By selecting this option, the request for data will be directly sent to the web service each time instead of the IE cache.

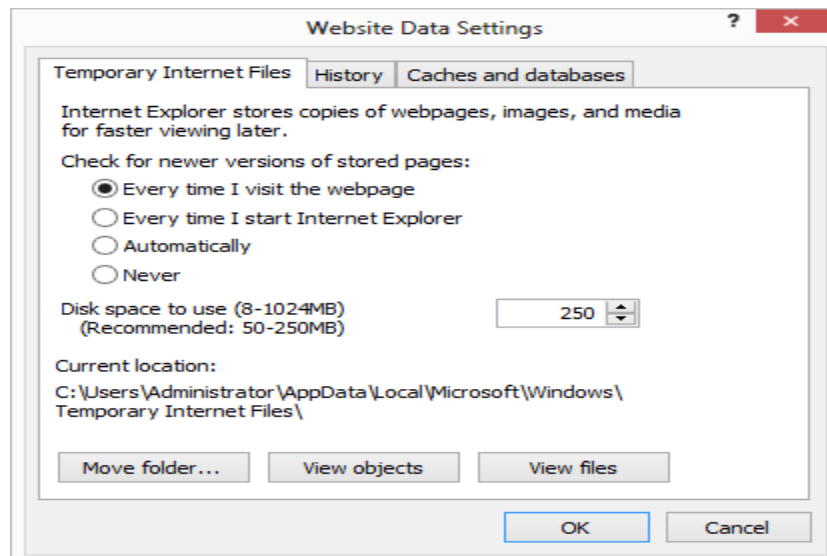


Figure 7.3 Website data settings

### 7.3 User Interface Improvements for Biomedical Image Annotation

The data set such as lung x-rays was far more demanding than the other datasets primarily because it required support for large lung boundaries and a better user interface to handle the masks. The lung X-ray labelling was a difficult task and was done as a combined effort by radiologists and researchers both at the University of Missouri-Columbia and the National Institute of Health. To support the business requirements, several enhancements to FireFly’s user interface had to be made.

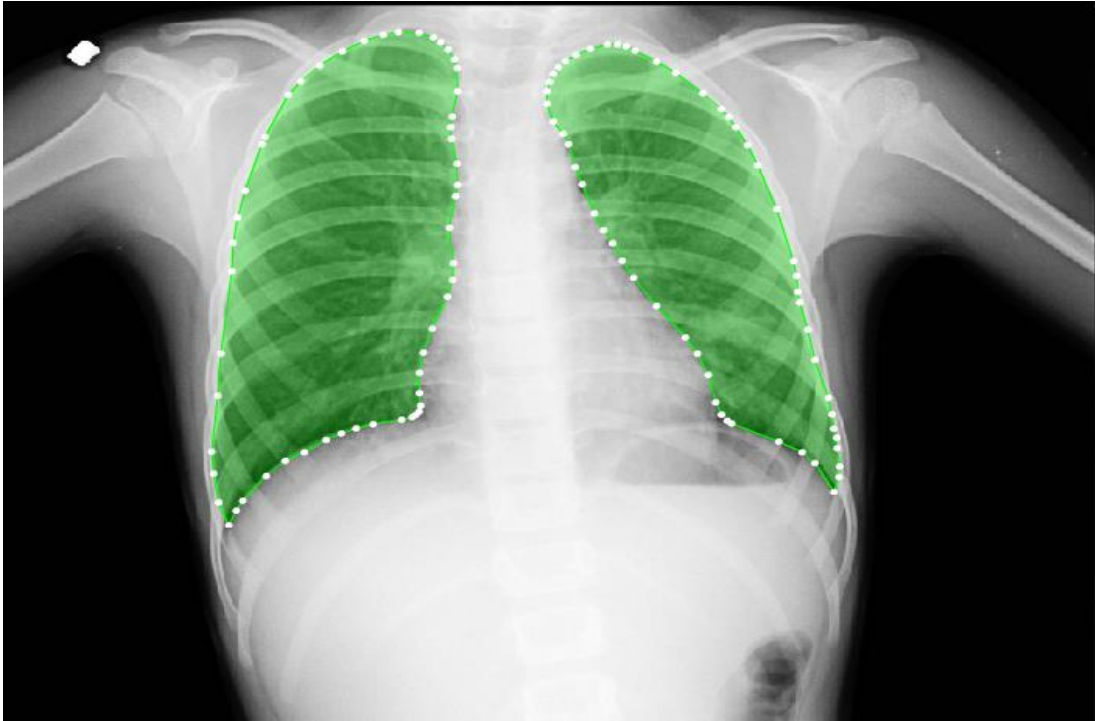


Figure 7.4 Lung X-ray with the mask

The enhancements to the user interface were as followed-

- 1) Change in Class Label Colors- Class labels were changed to white text on a black background (instead of a colored background) with a small colored square signifying the class label color. This made multiple labels in a complex image a lot easier to be read. Intensive research was done on web color brewers to choose a unique range of colors for each class type.
- 2) Object Auto Complete & Auto Save - Advancing to the next frame in the middle of drawing mode didn't automatically save the annotations, which was causing problems to the user. Few users wanted the tool pointer to revert back to selector/pointer tool each time they advance to the next image while others wanted to still be in polygon, polyline or spline drawing mode. Eventually, completing the object and auto-saving it when a user advances to the next image resolved the issue. In the new frame, the user continued to be in polygon drawing mode and needed to press the 'Esc' key to exit from the polygon/polyline/spline drawing mode and revert to the selector/pointer mode.

- 3) Select/Unselect Checkbox- With multiple classes, it was often tedious and time consuming for the user to select and unselect classes one at a time, especially when labeling a large number of objects. As a solution, a new checkbox was introduced to allow the user to select all or unselect all objects at once.
- 4) Info Message Window Relocation: The info message window was relocated from center top to bottom left of the screen. This window is used to display messages related to the current operations. While labeling the X-rays, the placement of the info message window was determined to be an obstacle by the users as it had to be moved each time the user tried to mark something. It was relocated down on the left corner to provide a better view to the user.
- 5) Undo Option- The user being in polygon drawing mode used to accidentally drop an object in order to adjust the image. A quick shortcut for removing the object was added. Ctrl + Z is used to undo last operation in case of object drawn. If a user has dropped multiple objects he can do Ctrl + Z as many number of times he wants and the objects will be erased in the decreasing order it was drawn. The user cannot undo anything once it has been saved in the database.
- 6) Object Glow- Each time an object was selected, it used to glow. However, this glow was removed from all drawing objects and was added to the point object.

Support of lung boundaries was a challenging task in the beginning. Since each lung had many points on the screen and drawing, editing and displaying had to be done in real time. With continuous feedback from researchers and radiologists at University of Missouri and National Institute of health, FireFly evolved to be a valuable tool in marking the masks and labeling X-rays.

## 7.4 Incremental Save vs. Bulk Save

FireFly uses Loader max to load the data. FireFly v2 supported a bulk save and retrieval. This saving technique worked well for datasets with smaller number of objects in the frame. However, this became a problem when FireFly was being used for larger datasets such as cell count of Malaria cells. These images as show in Figure 7.5 had more than 500 objects on each frame.

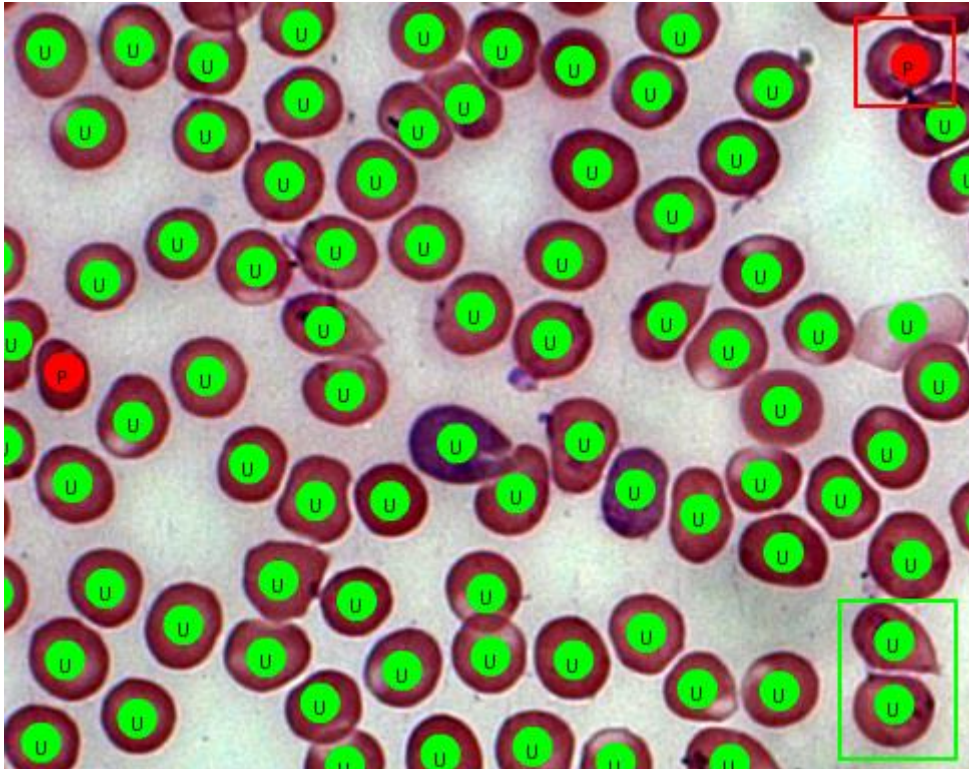


Figure 7.5 Malaria cell count

Each save would update all the objects in the database even the ones that weren't changed. For small operations like changing colors, it took more than a minute to be saved in the database. Moreover, each class change operations were being directed to save operations in the database instead of just changing in the local cache. Similarly, the delete operation would delete the object from the database and update all the objects on the screen. These bulk saves caused issues when the user tried to draw or edit any objects on the screen. The response received used to refresh the entire screen, which would also lose any unsaved changes. This caused inconsistency and the user wasn't even notified as to which objects were saved and which weren't. A prompt solution was

required to enhance the saving mechanism and after multiple discussions the following solution was proposed-

1. Operations like dropping new objects, deleting an object or changing any attributes of the object such as class must not update the database at that instance and the changes must only be saved in the local cache and screen buffer.
2. Dirty bits have to be associated with each object which would signify a change in the object. This change can be anything from changes in coordinates to attributes.
3. Saving only the objects, which were drawn, deleted or changed into the database and not update all the objects. This process was called incremental save.

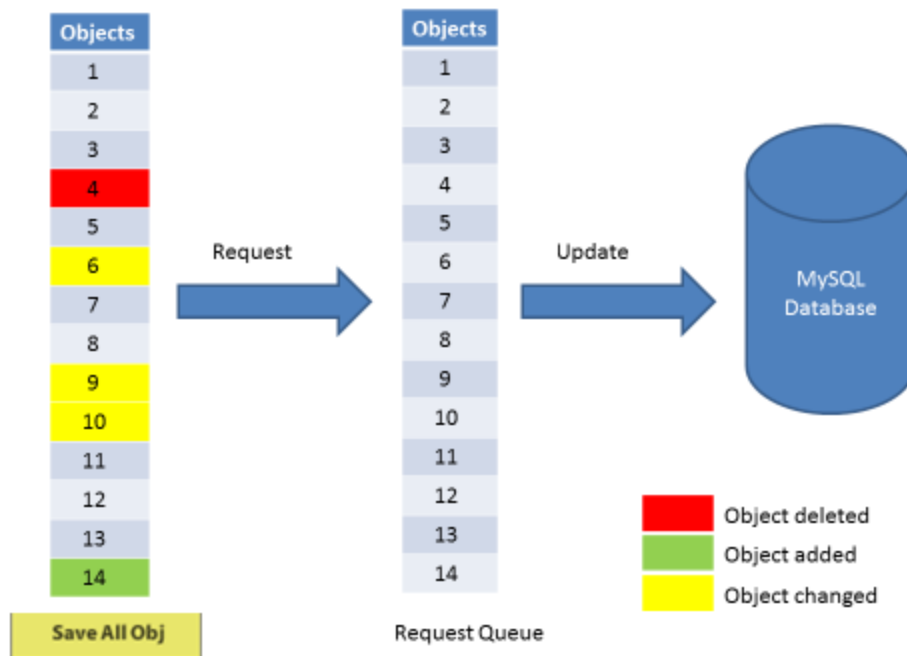


Figure 7.6 Bulk save

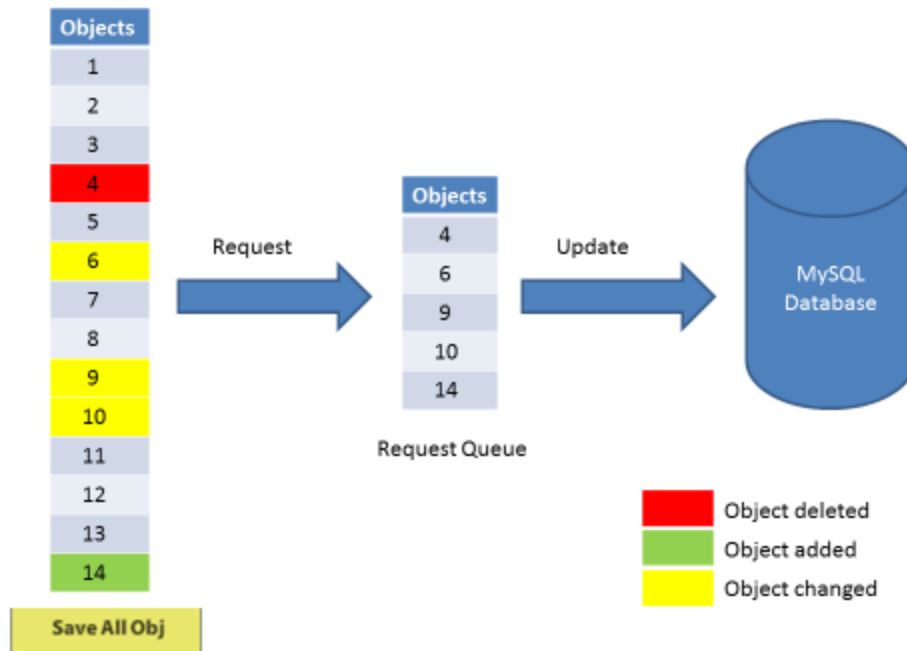


Figure 7.7 Incremental save

The solution was achieved by updating the saving mechanism on both the client and server side. On the client side, each object was associated with a change bit. This change bit was a Boolean variable that signifies that the attributes or dimensions of the object have been changed. The delete bit would be set if that object is deleted from the screen. The new objects were marked with `markedObjectID=0`. When any of these operations happened, each object was marked with the change and the screen was updated. When the user moved to the next frame with the auto save option on or did an explicit save; only the objects, which were changed, were saved in the database. This reduced the number of objects from more than 500 to 2 or 3 being transferred by HTTP. This reduced time remarkably for any database update. The operation that took more than a minute to be saved was now done in 2 or 3 seconds. Figure 7.6 shows Bulk save, where all the element are sent as a request queue to the server and updated in the database. Figure 7.7 shows the Incremental save in which only the objects which have been changed, deleted or added are sent as a request queue to the server for the update. This change was a major enhancement to FireFly's performance and it enabled users to function and correct ground truth faster and efficiently.



# Chapter 8

## New Intuitive GUI Tools

Flex gives us a freedom of three coordinate [26] system.

1. Global coordinates- The coordinates system with respect to workspace in Adobe Flash player is called Global coordinate system. The origin of the coordinate system is located at the upper-left corner of the stage (workspace of the application). With the MouseEvent class, the X, Y coordinates which can be accessed are the actual coordinates of the stage of the application and not screen coordinates.
2. Local coordinates- The coordinate system with respect to the component is called the Local Coordinate system. The origin is located at the upper-left corner of the component. Flex transforms coordinates internally to global coordinates while drawing each component. Any object can be drawn using its local coordinate system.
3. Content coordinates- The content coordinates system includes all the component content. Even though if any area is not visible it will be still in the coordinate system and can be seen by using scroll.

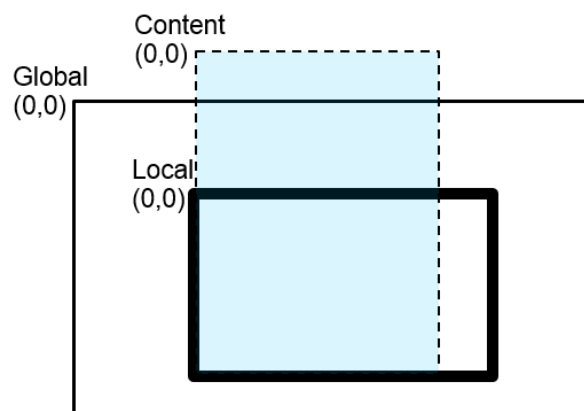


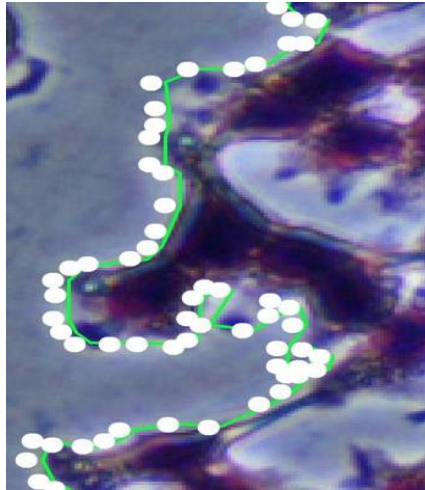
Figure 8.1 Coordinate system in Flex

Each object and image is treated as a different component in FireFly. FireFly works with local coordinate system for individual components. In the case of an image we take the coordinates based on the local coordinate system for the image. This allows us to move the image freely on the screen without caring about the zoom level and position of image on the screen. MouseX and MouseY function always provide the coordinates of the image not the screen.

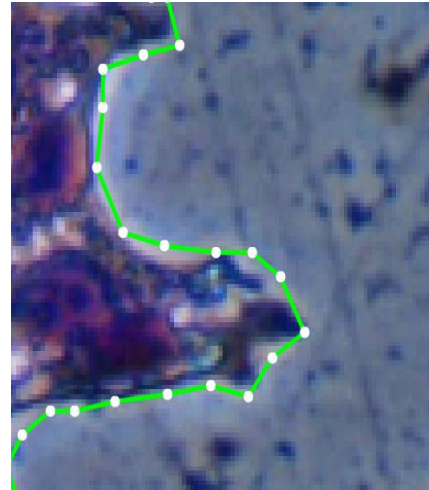
For drawing any object on an image, the first coordinate of the object is taken as origin and the object is drawn using the local coordinate system. As Flex takes care of transformations in the background, the object doesn't have to be transformed if the image is moved or zoomed. In the database, the image coordinates are stored instead of the local coordinates. After all the operations are done, the offset is added to the local coordinates to convert into image coordinates. These image coordinates are stored in the database.

## 8.1 Creeping of Polygons and Polylines

All the GUI elements in FireFly have been implemented using these transformations. Elements like polygons and polylines involve a series of summation of the delta difference between the consecutive coordinates. Polygons and polylines existed in FireFly without any editing options. To make these objects editable interactive edit points had to be added on exact vertex positions similar to boxes and lines. These edit points provided an interface for the user to change polygons and polylines once they have been created. As the edit points are moved the two lines connected to this point are redrawn. This feature proved to be very beneficial in marking lung boundaries and vessels. But during the course of marking the wound boundaries, the edit points were found to be shifted from their actual positions i.e. they were not at the vertex and were displaced by few sub pixels. This issue proved to be a bottleneck in marking the boundaries as every time the user clicked on the edit point the whole shape was redrawn and shifted by few subpixels. To solve this problem whole implementation of polyline and polygon had to be changed and made consistent with the interactive edit point calculation. Figure 8.1 (a) shows the displaced points from the object and Figure 8.1 (b) shows the correct placement of the points after change in implementation.



(a) Creeping of points



(b) Correct position of points

Figure 8.2 Creeping of points in wound dataset

Figure 8.3 shows the earlier code for the calculation of the delta between coordinates and then drawing other polylines. The code was found to be complex and due to unnecessary calculations a small value was being added which became significant after multiple additions. This made the actual line to be shifted by few sub pixels. On the other hand the edit points were plotted based on the calculation shown in Figure 8.4. The whole code for polylines and polygons was rewritten and modularized. Figure 8.4 shows the method to calculate the distance and Figure 8.5 shows the method to draw the polyline based on the delta which is already calculated. A separate method for calculation of distance reduced code at several places. This method can be called whenever the objects have to be redrawn or edit points have to be placed. The calculated delta distance has the same value now as from both the places the same function is being called. Hence, there is no shifting or creeping of points.

```

if(_redrawFlag){
var temp:VOFreeFormPoly= VOFreeFormPoly(_markedObject);
this.graphics.clear();
if(!(temp.closed)){
for(var i:uint=0;i<temp.points.length;i++)
{
k=i;
if(i+1 != temp.points.length)
{
//draw a red line between each vertex
this.graphics.lineStyle(poly_thick, _classLayer.color, 1);
if(i==0){
this.graphics.moveTo(0,0);
this.graphics.lineTo(temp.points[i+1].x-temp.points[i].x,
temp.points[i+1].y-temp.points[i].y)
ptx=temp.points[i+1].x-temp.points[i].x;
pty=temp.points[i+1].y-temp.points[i].y;
}
else
{
this.graphics.moveTo(ptx,pty);
for(var j:int=k+1; j>0;j--)
{
sumx=sumx+(temp.points[j].x-temp.points[k].x);
sumy=sumy+(temp.points[j].y-temp.points[k].y);
k--;
}
ptx=sumx;
pty=sumy;
sumx=0;
sumy=0;
this.graphics.lineTo(ptx,pty);
}
}
}
}
}
}

```

Figure 8.3 Old code for calculating and drawing

```

if(_redrawFlag){
var moPoly:voPoly= voPoly(_markedObject);
this.graphics.clear();
if(!(moPoly.closed)){
deltaPoints[0].x=0;
deltaPoints[0].y=0;
for(i=1;i<temp.points.length;i++){
deltaPoints[i].x=deltaPoints[i-1].x+(moPoly.points[i].x-moPoly.points[i-1].x);
deltaPoints[i].y=deltaPoints[i-1].y+(moPoly.points[i].y-moPoly.points[i-1].y);
}
}
}
}

```

Figure 8.4 New code for calculating distance

```
for(i=0;i<deltaPoints.length-1;i++){
    this.graphics.lineStyle(poly_thick,_classLayer.color,1);
    this.graphics.moveTo(deltaPoints[i].x, deltaPoints[i].y);
    this.graphics.lineTo(deltaPoints[i+1].x,deltaPoints[i+1].y);
}
```

Figure 8.5 New code for drawing

## 8.2 Additional Annotation Objects

FireFly supported drawing objects like box, lines and points, which were used for object annotations. Polygons and polylines did not provide the user with any editing options, which made it difficult for the user to mark or change any vertex. The only option was to delete the whole object and redraw it.

Taking into consideration the growth and support of datasets with a variety of shapes and figures, new GUI elements had to be implemented. The new features implemented were:

- Circle
- Polygon (with editing option)
- Polyline (with editing option)
- Free Form
- Curve

### 8.2.1 Circle

This drawing tool is used for annotating circular objects. The Bresenham's algorithm is used for drawing a circle in FireFly. The algorithm is as follows:

Bresenham's circle algorithm calculates the locations of the pixels in the first 45 degrees. It assumes that the circle is centered at the origin. For every pixel (x, y), it calculates and draws a

pixel in each of the 8 octants of the circle:

```
PutPixel(CenterX + X, Center Y + Y)
PutPixel(CenterX + X, Center Y - Y)
PutPixel(CenterX - X, Center Y + Y)
PutPixel(CenterX - X, Center Y - Y)
PutPixel(CenterX + Y, Center Y + X)
PutPixel(CenterX + Y, Center Y - X)
PutPixel(CenterX - Y, Center Y + X)
PutPixel(CenterX - Y, Center Y - X)
```

Interactive editing of the circle was an important requirement for marking the objects. Figure 8.6 displays a circle drawing with a center point and a point on the circumference. Each time the points are moved, the circle is updated. A new table was created in the database to store the center and circumference points.



Figure 8.6 Circle drawing with edit points

## 8.2.2 Polygons

Simple polygon drawing has been supported for a long time in FireFly. But once a polygon was drawn it can't be changed or edited. For any small changes the polygon needs to be deleted and redrawn. This wasn't a feasible technique especially in the case of lung boundaries as shown in Figure 8.3. Hence, polygon drawing was extended with new editing features. With this editing feature now, the vertices can be moved anywhere on the screen providing a rubber band drawing effect. Besides the moving vertex, the remaining vertices are fixed and don't move. Vertex movement causes interactive changes on the connected edges only. As the points are moved, the edges are interactively erased and redrawn without moving any of the other vertices. This provided a rapid approach to mark the ground truth for several objects in multiple X-ray images. For drawing any line in Flex `moveTo(x, y)` and `lineTo(x1, y1)` methods are used. The first click is taken as origin in the local coordinate system, so we have to move to  $(0, 0)$  and then draw a line to the distance computed (difference of the actual coordinates). Drawing a polygon is summation of multiple `moveTo` and `lineTo` functions. Multiple lines are drawn one after another and finally the last vertex is joined to the first vertex to complete a polygon.

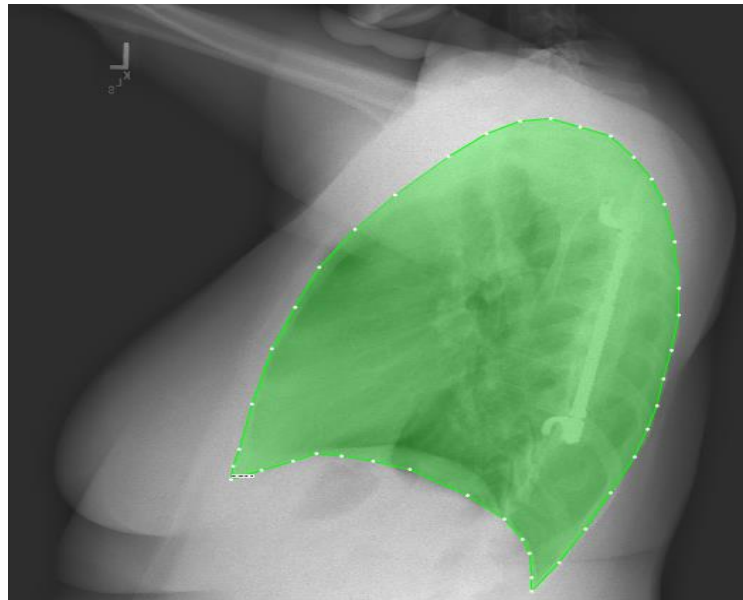


Figure 8.7 Polygons used for marking lateral side of lungs

### 8.2.3 Polylines

Similar to polygons, an enhanced version of polylines with more editing options was required. Implementation of polylines is very similar to polygons with the basic difference being the join between the last and first vertex. Polyline also uses the same concept of `moveTo` and `lineTo` functions as polygons. Polylines are being used to mark boundaries and medial axis in vessels, as shown in Figure 8.8. Polylines also support rubber band drawing, which means that each time a vertex is moved; interactive changes are made along the edges. As the vertices are moved, the edges are interactively erased and redrawn without moving any of the other vertices. This provided a rapid approach to mark the ground truth for several objects in multiple vessel images.

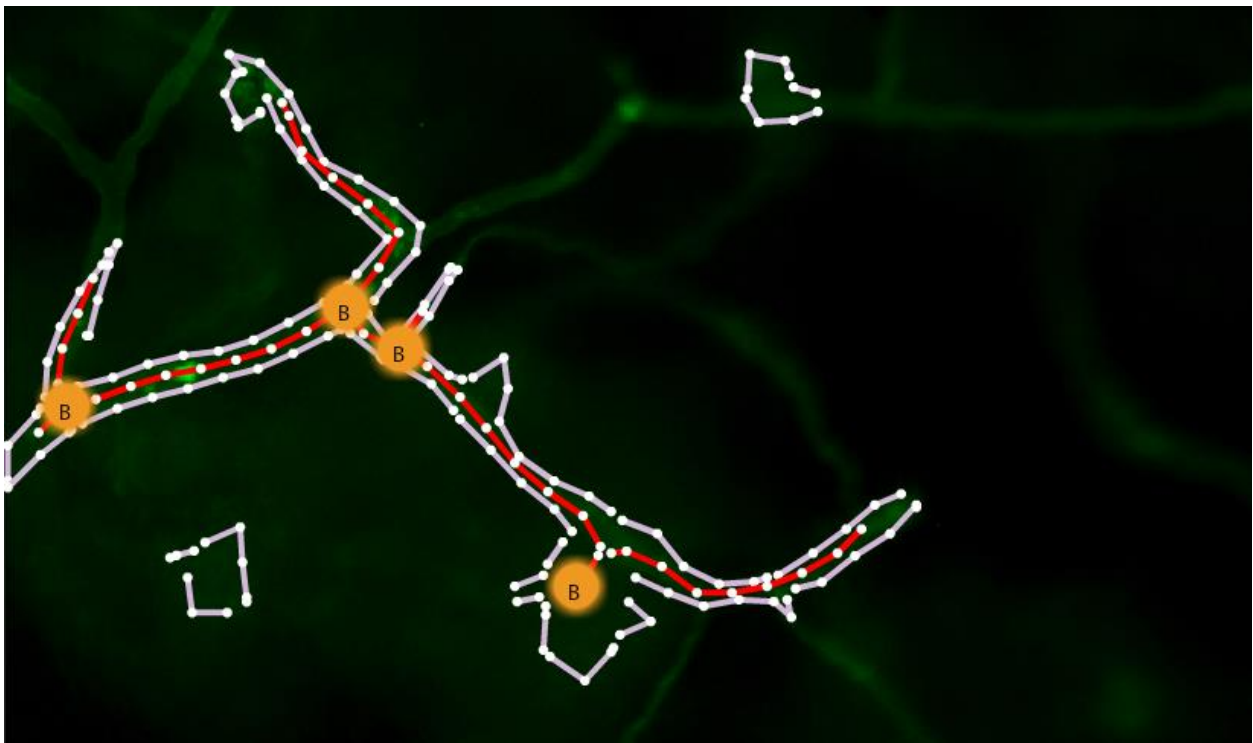


Figure 8.8 Polylines in vessels image



## 8.2.4 Curve

This type of drawing tool is useful for marking curled objects, as number of points is reduced along the curved surfaces. Curves are represented by a polynomial equation. Interactive curve editing was a challenging task since each new point changes the polynomial equation totally. Another challenge was database storage, as only coordinate points are stored and not the constants.

An approach similar to polylines and polygons was applied and a three point Bezier curve was implemented. The first and last clicks became the fixed points and the second point was the control point for the curve. Instead of having a single curve, multiple Bezier curves were plotted next to each other. The control point in the middle is used to interpolate the curve. The user can interactively interpolate and change the bending of curve by moving the control point. As shown in the Figure 8.9, the curve consists of 5 curves joined together to represent one curve.

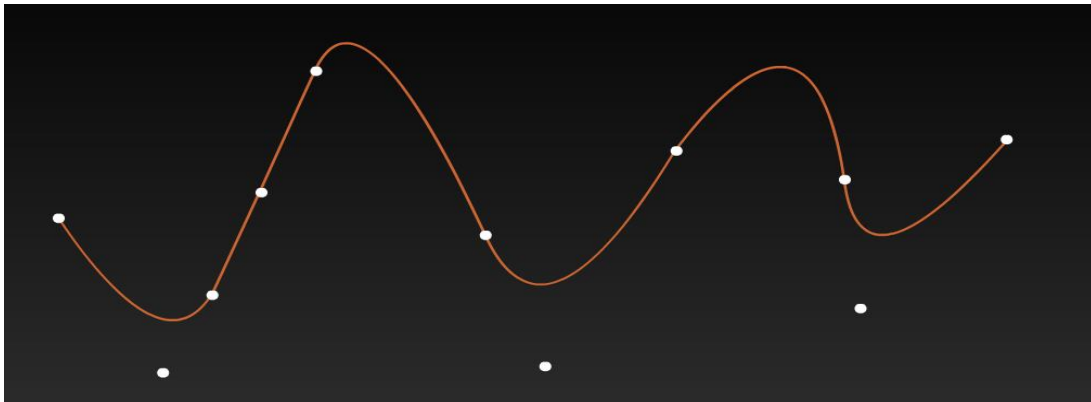


Figure 8.9 Curves

## 8.2.5 Free Form

This type of drawing tool is used to draw free form shapes. However, a disadvantage of using this tool can be the total number of points that are required for irregular shapes. Each of these points has to be stored in the database. The drawing algorithm samples the point along straight lines, which are then stored in the database. Sampling helps to reduce points however, for larger and irregular shapes it can be extremely time consuming. Moreover, editing is difficult because of the presence of voluminous points. The algorithm for free form drawing is similar to the algorithm

for polygons. Instead of click event the drag event from the mouse is captured to place the points. These points are connected through straight lines to give the final figure.



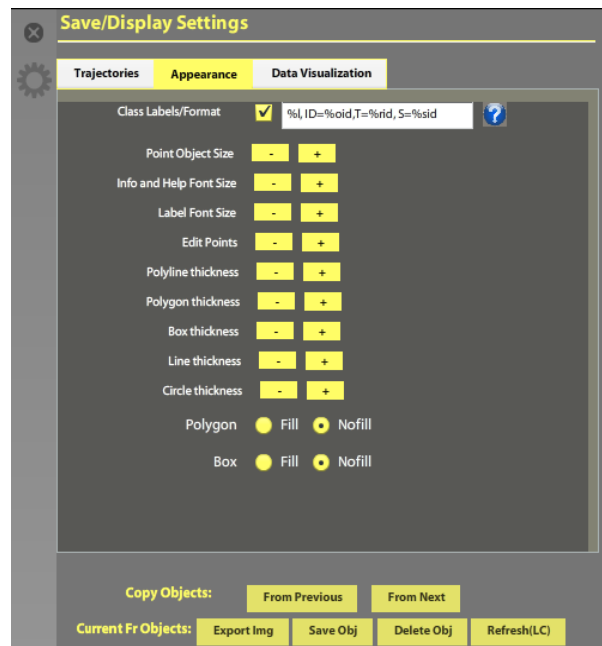
Figure 8.10 Free form

### 8.3 Redesign of Save/Display Panel

Several features had to be added which were related to the object's appearance. This led to the redesigning of the panel. Instead of a single tab, the features are separated under 2 different tabs as shown in Figure 8.11(c). At an intermediate stage Data Visualization tab was also added as shown in Figure 8.11(b). Later this tab was removed and added in the Segmentation panel.



(a) Version 2.0



(b) Version 2.5



(c) Version 3.0

Figure 8.11 Save/Display panel- version updates

Figure B.8 shows a more detailed view of Save/Display panel. The description of old features has been already described under P. Madala's Thesis [21]. The new features added with their functions are:

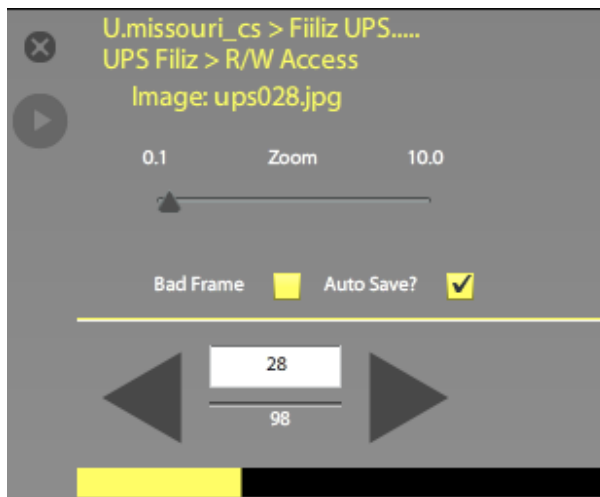
Appearance:

- a. Edit Points: Used to increase or decrease the size of the edit points
- b. Line Thickness: Used to increase or decrease the thickness of a line in line objects
- c. Polygon Thickness: Used to increase or decrease the thickness of boundaries of the polygons
- d. Polyline Thickness: Used to increase or decrease the thickness of the polylines
- e. Box Thickness: Used to increase or decrease the thickness of sides of the boxes
- f. Circle Thickness: Used to increase or decrease the thickness of circumference of circles

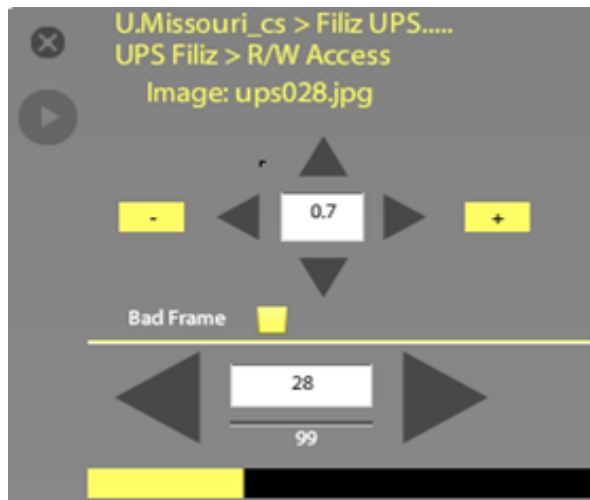
- g. Boundary fill/ no fill for Polygon: These provide an option to make interior of a polygon filled or empty with a semi-transparent color.
- h. Boundary fill/ no fill for Box: These provide an option to make inside of a box filled or empty with a semi-transparent color.

## 8.4 Redesign of Frame Advance Panel

Frame Advance Panel consists of features relative to zoom, buffer and movement of frames. The zoom functionality, initially consisted of just a scroll bar, which was replaced by an input box and two + and – buttons. The input box shows the zoom level, which the user can also directly input. + and – button represents zoom in and zoom out functionality and are dependent upon the zoom level. Ctrl + and Ctrl – can also be used to zoom in and out.



(a) Version 2.0



(b) Version 3.0

Figure 8.12 Frame advance panel- version updates

The four direction buttons indicate image movement. If the user wants to move the image left, right, up and down during editing, they can use one of these buttons for image movement.

## 8.5 Redesigning of Tools Panel

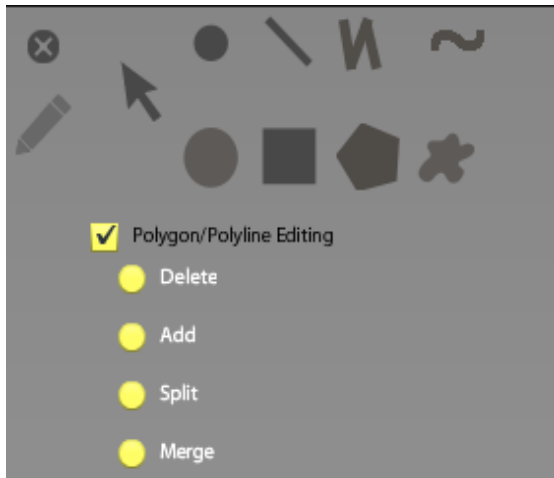
Tool Panel consisted of all the drawing tools. But with recent addition of polygon and polyline editing, several new modes were introduced which required a complete redesigning of the panel with some new features. Figure 8.9(a) shows the tool panel in Version 2. Figure 8.9(b) shows tool

panel in Version 2.5 with new P/P editing options. Figure 8.9(c) shows the Version 3.0, with editing and different mode select options. The red and blue arrow indicate track and polygon/polyline editing mode. The database operations were moved from Save/Display panel to the tools panel as a separate tab Save.

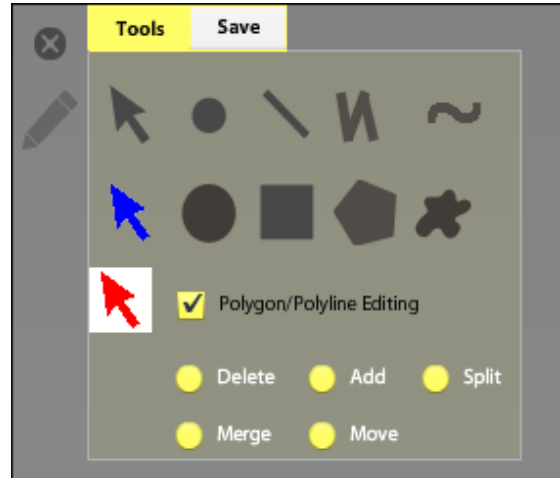
The tool tab contains all the tools with P/P editing tools. The save tool has all the options to save, delete and retrieve objects from the database. More detailed view of the tools panel has been shown Appendix B.



(a) Version 2



(b) Version 2.5



(c) Version 3.0

Figure 8.13 Tools panel version updates

# Chapter 9

## Import/Export for Image Analysis

FireFly currently supports 4 different file formats. FireFly being a more generic tool had to adapt to different kinds of file imports and exports. Initial versions supported .csv files, which were used to load data for cell tracking. With new datasets the need for new formats evolved. FireFly mainly exports/imports data for two major features: tracking and segmentation

Both features required different fields and data, which meant they had to be exported in different formats. The export format was also dependent on the requirements from users of various projects. Initially, the segmentation results for Lungs and Microvasculature image sets were exported in two different formats, however, this later evolved to a more standard ROI format to support multiple user need.

### 9.1 KW 18 Format

KW 18 format is primarily used for tracking purposes. The results from different tracking algorithms or ground truth are exported in this format, which is further imported by FireFly to load the data in its database with the use of controllers. FireFly exports the data in the same format. The GUI for the file to be downloaded is in the attribute panel as shown in Figure 9.1. The data is transferred using AMF (Asynchronous Message Format) to the client side, which is written into a text file on user's local machine.

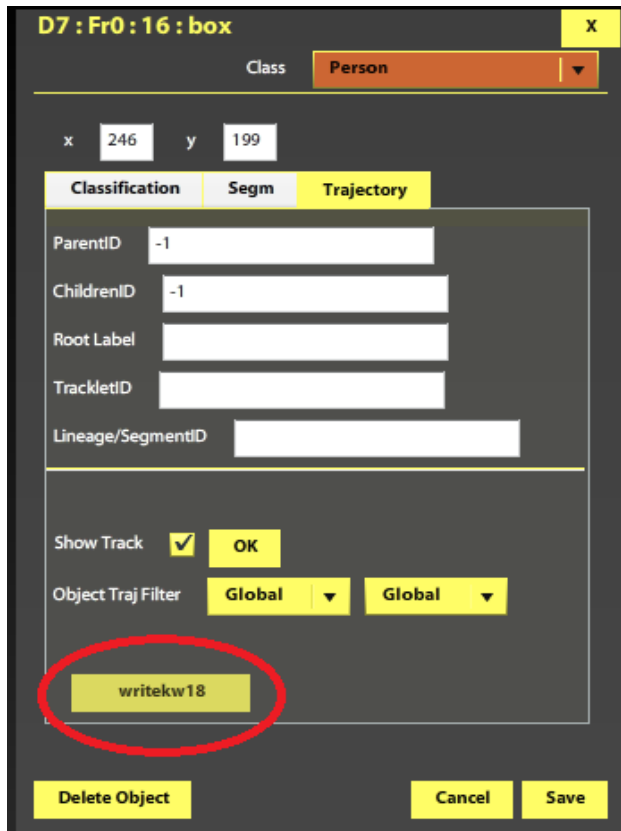


Figure 9.1 GUI for writing KW 18 file

Field details for KW18 format:

No.	Field	Description
1	Track-ID	Unique ID for each track
2	Track-Length	Length of the track i.e. number of frames the track exists.
3	Frame-number	Frame number where the track starts
4-5	Tracking-plane-loc (x, y)	Location of the plane
6-7	Velocity (x, y)	Velocity of track object
8-9	Image-loc (x, y)	Location of the image
10-13	Imgbox (TL_x, TL_y ,BR_x, BR_y)	Coordinates of bounding box (top left and bottom corner)
14	Area	Area of the box

15-17	World-loc (x, y, z)	Camera location
18	Timestamp	Time-stamp during image acquisition.
19	Object-type-id	ID to classify the object
20	Activity-type-id	ID to classify activity

Table 9.1 Details of KW18 format

```

MU_ARGUS_GT_V2Registered_no_neg_manually_corrected_10_06_2011_freq_1kw18
1 # 1:Track-id 2:Track-length 3:Frame-number 4-5:Tracking-plane-loc(x,y) 6-7:velocity(x,y) 8-9:Image-loc(x,y) 10-13:Img-bbox(TL_x,TL_y,BR_x,BR_y) 14:Area 15
2 0 40 0 1381.4549999999999 914.1920000000000 0 0 1381.4549999999999 914.1920000000000 1368 890 1394 938 1231.37280000000026 0 0 0 0 -1 -1
3 0 40 1 1378.9549999999999 953.1920000000000 0 0 1378.9549999999999 953.1920000000000 1365 929 1392 977 1280.72448000000017 0 0 0 1 -1 -1
4 0 40 2 1379.625 975.7305000000000 0 0 1379.625 975.7305000000000 1366 950 1392 1000 1309.7091299999913 0 0 0 2 -1 -1
5 0 40 3 1377.9549999999999 999.0605000000000 0 0 1377.9549999999999 999.0605000000000 1364 974 1391 1023 1309.7091299999995 0 0 0 3 -1 -1
6 0 40 4 1376.625 1024.068 0 0 1376.625 1024.068 1363 999 1389 1048 1309.8414799999988 0 0 0 4 -1 -1
7 0 40 5 1375.9549999999999 1049.1900000000000 0 0 1375.9549999999999 1049.1900000000000 1362 1025 1389 1073 1280.61860000000041 0 0 0 5 -1 -1
8 0 40 6 1373.79 1073.4000000000000 0 0 1373.79 1073.4000000000000 1361 1048 1386 1098 1259.76080000000023 0 0 0 6 -1 -1
9 0 40 7 1372.7950000000000 1099.4000000000000 0 0 1372.7950000000000 1099.4000000000000 1360 1074 1385 1124 1259.26600000000028 0 0 0 7 -1 -1
10 0 40 8 1371.7950000000000 1124.4000000000000 0 0 1371.7950000000000 1124.4000000000000 1359 1099 1384 1149 1259.26600000000028 0 0 0 8 -1 -1
11 0 40 9 1370.625 1151.1900000000000 0 0 1370.625 1151.1900000000000 1357 1127 1383 1175 1280.6185999999932 0 0 0 9 -1 -1
12 0 40 10 1371.79 1178.0699999999999 0 0 1371.79 1178.0699999999999 1359 1153 1384 1202 1259.76080000000023 0 0 0 10 -1 -1
13 0 40 11 1370.1199999999999 1203.73 0 0 1370.1199999999999 1203.73 1356 1178 1383 1228 1359.71039999999902 0 0 0 11 -1 -1
14 0 40 12 1370.9549999999999 1230.0599999999999 0 0 1370.9549999999999 1230.0599999999999 1357 1205 1384 1254 1309.73560000000018 0 0 0 12 -1 -1
15 0 40 13 1369.4549999999999 1257.1900000000000 0 0 1369.4549999999999 1257.1900000000000 1356 1233 1382 1281 1231.27100000000005 0 0 0 13 -1 -1
16 0 40 14 1369.4549999999999 1282.1900000000000 0 0 1369.4549999999999 1282.1900000000000 1356 1258 1382 1306 1231.27100000000005 0 0 0 14 -1 -1
17 0 40 15 1368.79 1307.95 0 0 1368.79 1307.95 1356 1282 1381 1333 1287.7667999999999 0 0 0 15 -1 -1
18 0 40 16 1367.625 1335.0699999999999 0 0 1367.625 1335.0699999999999 1354 1310 1380 1359 1309.73559999999907 0 0 0 16 -1 -1
19 0 40 17 1367.625 1361.1900000000000 3.1390939237593481e-056 2.811009477669641e+076 1367.625 1361.1900000000000 1354 1337 1380 1385 1280.61859999999932 0 0 0 17 -1
20 0 40 18 1366.9549999999999 1386.6199999999999 2.5791724086912201e-020 -1.4543336299621233e+147 1366.9549999999999 1386.6199999999999 1353 1361 1380 1411 1338.8526
21 0 40 19 1366.9549999999999 1412.74 -2.3138987829034578e+170 8.1335696018598281e+098 1366.9549999999999 1412.74 1353 1388 1380 1437 1309.73560000000018 0 0 0 19 -1
22 0 40 20 1364.9549999999999 1439.74 1.8836568724513329e-213 3.0713878555405591e-022 1364.9549999999999 1439.74 1351 1415 1378 1464 1309.73560000000018 0 0 0 20 -1
23 0 40 21 1365.9549999999999 1465.0699999999999 1.632612384402691e-129 3.298975127274377e-096 1365.9549999999999 1465.0699999999999 1352 1440 1379 1489 1309.735600
24 0 40 22 1364.9549999999999 1490.74 -3.0859544541923594e-058 -1.9724680409329236e-083 1364.9549999999999 1490.74 1351 1466 1378 1515 1309.73560000000018 0 0 0 22 -1
25 0 40 23 1363.9549999999999 1516.74 -1.910557714570696e-287 1691987423.9097543 1363.9549999999999 1516.74 1350 1492 1377 1541 1309.73560000000018 0 0 0 23 -1 -1
26 0 40 24 1364.1199999999999 1542.4000000000000 -1.9520873998256005e-261 -3.7502809553011369e-048 1364.1199999999999 1542.4000000000000 1351 1517 1376 1567 1259.766
27 0 40 25 1363.2849999999999 1567.4000000000000 3.6705614785661661e-076 1.5186489900813552e-223 1363.2849999999999 1567.4000000000000 1350 1542 1376 1592 1309.73566
28 0 40 26 1363.45 1592.51 4151744919.0475645 -1.5221293460778299e+094 1363.45 1592.51 1349 1568 1377 1616 1329.4824000000004 0 0 0 26 -1 -1
29 0 40 27 1362.9540000000000 1618.0500000000000 663974546776160500 1.8921090071877772e-143 1362.9540000000000 1618.0500000000000 1349 1593 1376 1642 1309.73560000

```

Figure 9.2 KW-18 file

## 9.2 Lung Boundaries Format

Ground truth for lungs consisted of two main objects in the X-ray- left and right lung. The user needed only the coordinates from their ground truth. Two different text files for each lung were written. This format was chosen as it was easier to parse and presence of unwanted fields was removed.



Figure 9.3(a) contains the coordinates for the left lung and Figure 9.3(b) contains the coordinates for the right lung. FireFly reads and stores the coordinates in the order it was written in database. The drawing algorithm also fetches and draws the objects in the given order of points.

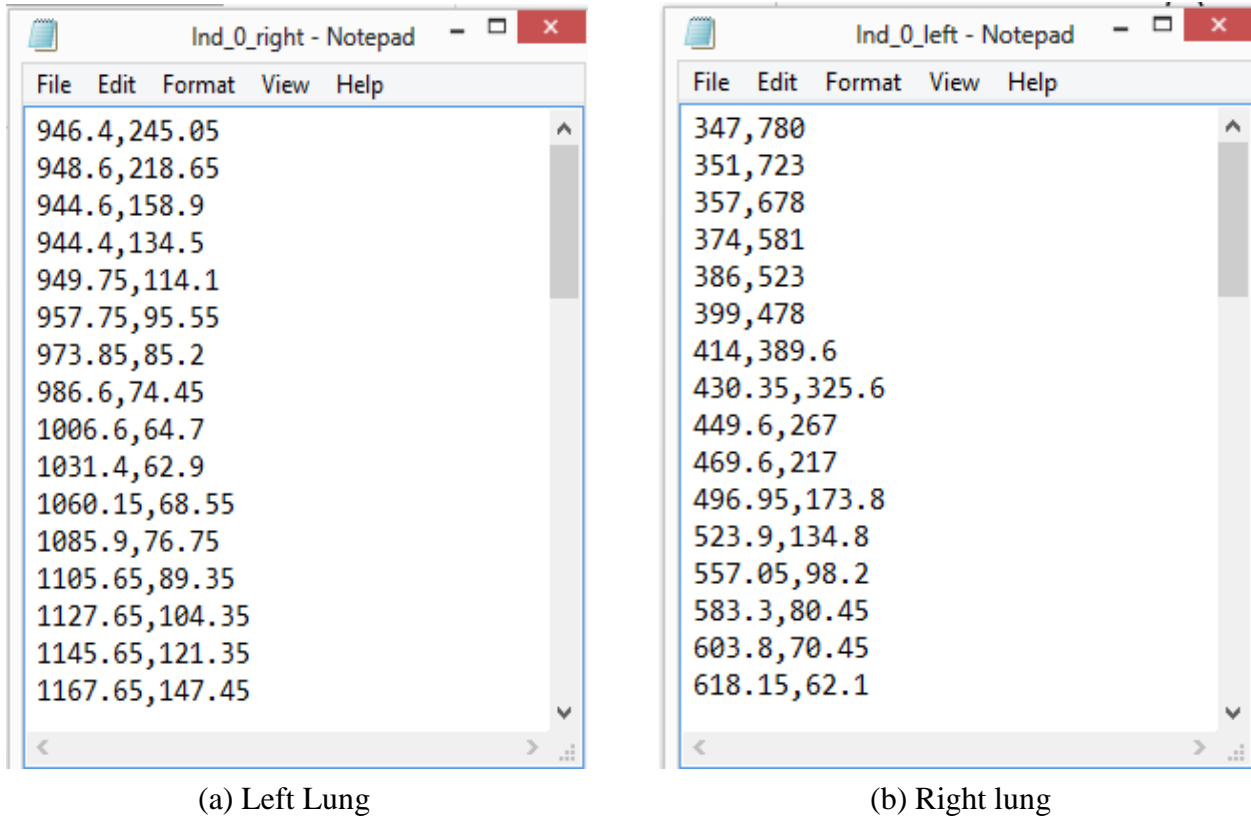
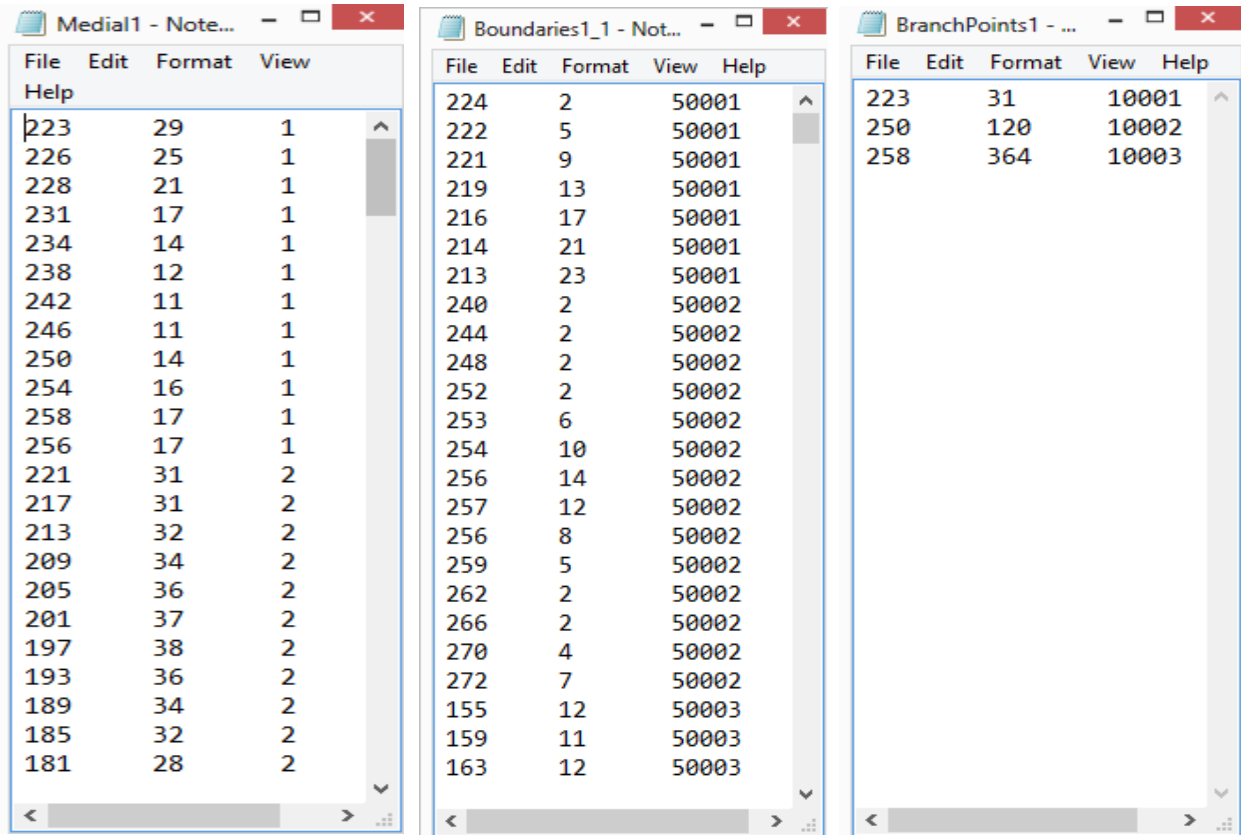


Figure 9.3 Lung coordinates

### 9.3 Microvasculature File Format

The file support required for Vasculature analysis was similar to lung segmentation. The main difference was presence of multiple objects instead of two objects. The objects were segments that were results from MATLAB executables. These results had to be loaded in the database and displayed in FireFly as polylines and points.

It was imperative to be able to distinguish between two objects in this file format. The points are drawn in the same order as they are saved in the database. The coordinates were written in the same order with assigned unique object ID's. As shown in Figure 9.3(a), (b), (c), the first two columns contain the coordinates and the third column contains the unique object IDs, which are used for identification of objects in FireFly. The files were divided into three types, based on class type- Boundaries, Branch Points and Medial Axis.



(a) Boundary

(b) Medial Axis

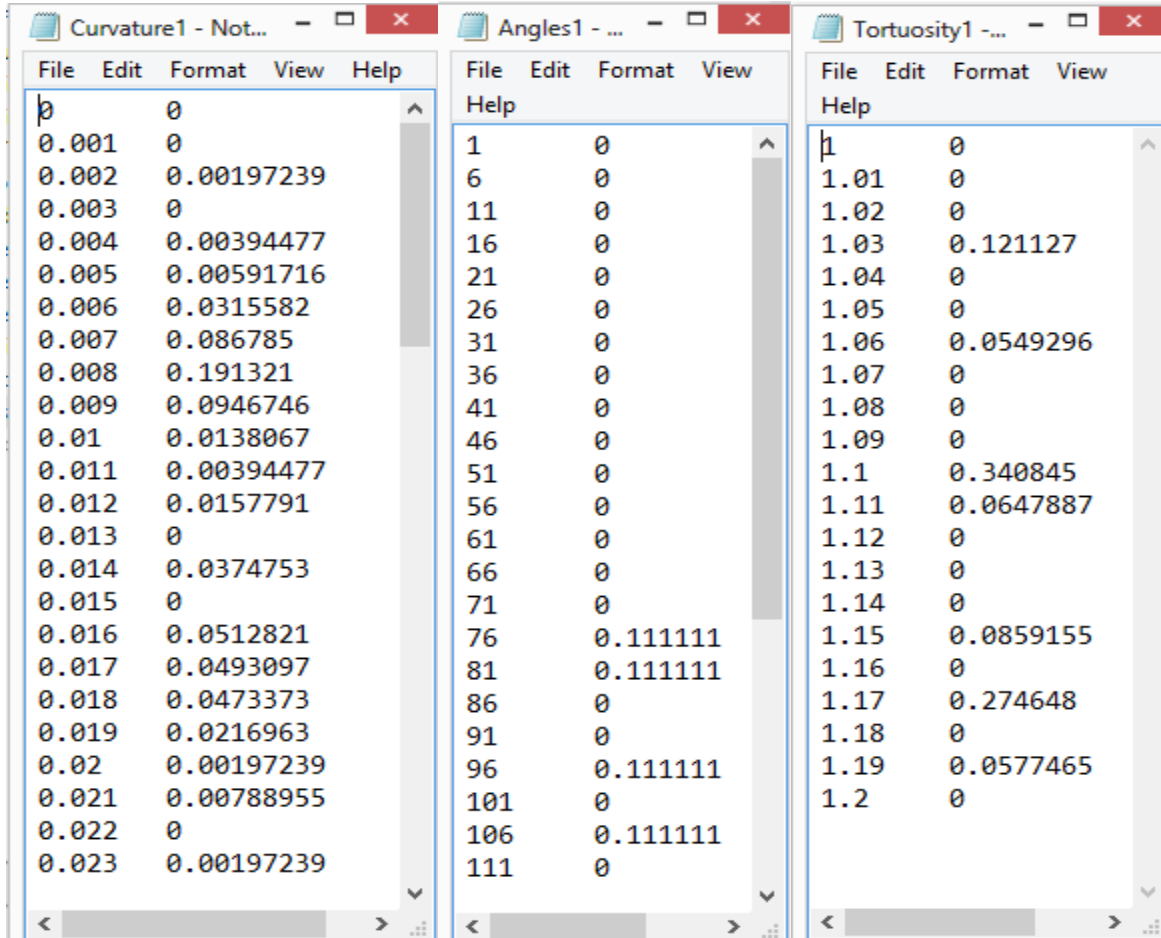
(c) Branch Points

Figure 9.4 Microvasculature file format

## 9.4 Data Analytics File Format for Microvasculature

The dataset that required this file format was Vessels. In vessels the data came from different species of mice before and after treatment. There was a need to study the Curvature, Angles and Tortuosity. To meet the requirement, data analytics feature was developed in FireFly. This feature allows users to run data analysis and generate the results. The results are further displayed in FireFly in the form of graphs by using one of the data visualization tools. The file format created

to study the curvature; angles and tortuosity are shown in Figures 9.4(a), (b) and (c). The first column in the file denotes the number for the angle, curvature, tortuosity and the second column denotes the associated value, which gave the user a quick way to read and plot the graph in FireFly.



(a) Curvature

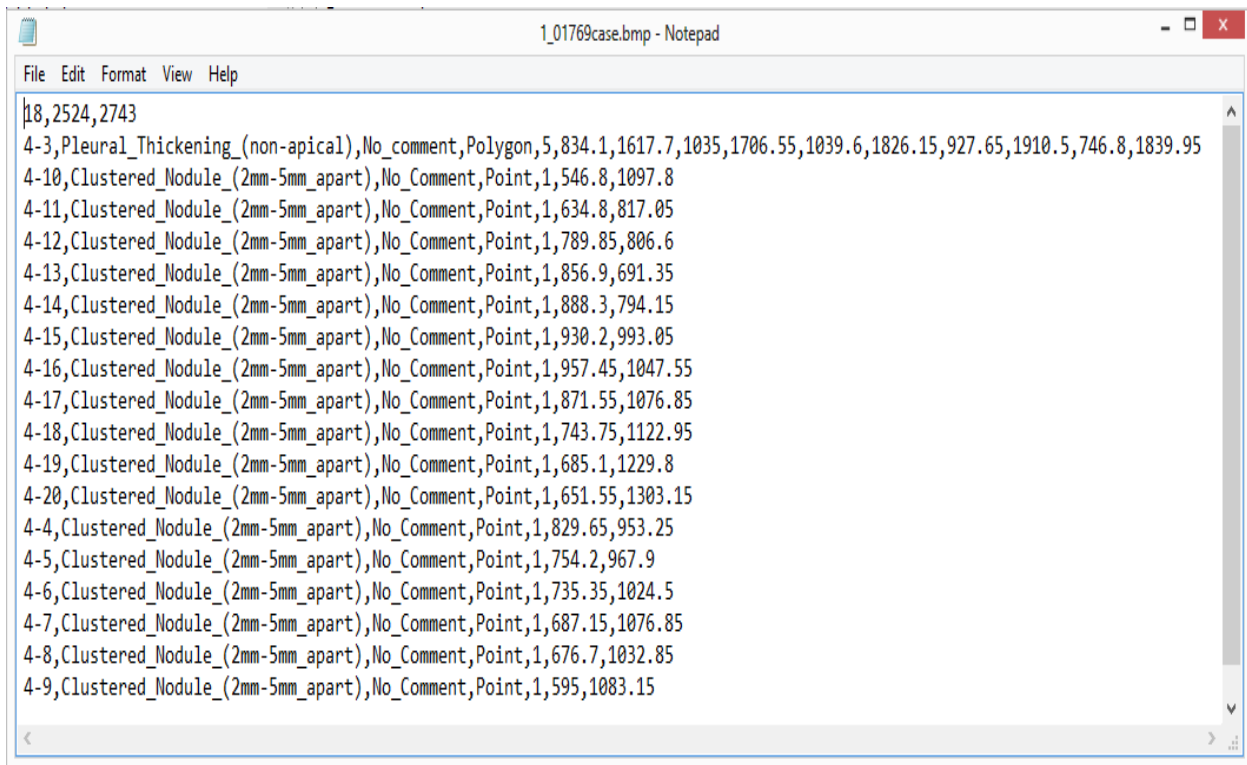
(b) Angles

(c) Tortuosity

Figure 9.5 Data analytics format

## 9.5 Region of Interest (ROI) Format

FireFly supports multiple techniques of exporting data. However, for every new format, change was required to the existing file formats. A uniform file format was needed, which would fulfil all the requirements for segmentation results. File import was more time consuming as the file was written in a vertical format rather than horizontal format. To insert a 10-point object, 10 lines had to be scanned, which was reasonable for lesser number of objects. But a more scalable format was needed for data with multiple objects in same frame. We supported files based on specific type of objects. The ROI format allowed us to support any kind of object with any kind of data. Figure 9.6 shows the file, it comprises of a minimum of 7 columns with the exception of the first row, which only contains 3 elements or columns.



```
18,2524,2743
4-3,Pleural_Thickening_(non-apical),No_comment,Polygon,5,834.1,1617.7,1035,1706.55,1039.6,1826.15,927.65,1910.5,746.8,1839.95
4-10,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,546.8,1097.8
4-11,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,634.8,817.05
4-12,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,789.85,806.6
4-13,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,856.9,691.35
4-14,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,888.3,794.15
4-15,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,930.2,993.05
4-16,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,957.45,1047.55
4-17,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,871.55,1076.85
4-18,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,743.75,1122.95
4-19,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,685.1,1229.8
4-20,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,651.55,1303.15
4-4,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,829.65,953.25
4-5,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,754.2,967.9
4-6,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,735.35,1024.5
4-7,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,687.15,1076.85
4-8,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,676.7,1032.85
4-9,Clustered_Nodule_(2mm-5mm_apart),No_Comment,Point,1,595,1083.15
```

Figure 9.6 ROI format file

The file is named based on the image name instead of the frame number, for example, 1\_01769case.bmp. The first row contains number of objects, resolution height and resolution width. All the rows besides the first row have the following fields in each column-

- a. Frame-Number-ObjectID: This field is a combination of frame number and ObjectID. Combined together, it gives a unique ID to the object outside FireFly.

- b. Class- It defines class of the object.
- c. Comment-Displays comments saved with respect to the object during labelling or marking of ground truth.
- d. Object type- It indicates if the object is box, circle, line, point, polygon, polyline, free form, and b-spline.
- e. Total number of points- Depending upon the object type, this field signifies total number of x, y coordinates written after this column for this object. For example,

Box: 2- the top left and bottom right coordinates of the box.

Circle: 2- the center and the edit point coordinate.

Point: 1- the coordinates of the point

Line: 2- the coordinates of two endpoints.

Polygon: n- the number of points in the polygon

Polylines: n- the number of points in the polyline

Freeform- n- the number of points in the freeform

B-spline: n- the number of points in b-spline

The point coordinates are written in x, y format. For example, a box will be denoted as  $x_1, y_1, x_2, y_2$ . A polygon will be denoted as  $x_1, y_1, x_2, y_2 \dots x_n, y_n$ . This format gives us a preview of the number of objects in the file and only a fixed number of lines would have to be scanned. It also gives us the information of number of coordinates, which helps us to split the coordinates and process in a much faster and efficient manner.

## 9.6 Downloading Results

Besides the direct link of KW18 formats, FireFly provides a feature of downloading all the results stored in the server. Every annotation has its file stored separately. A PERL script zips the folder including the subdirectory. The zipped folder is transferred over HTTP and downloaded on the client's machine.

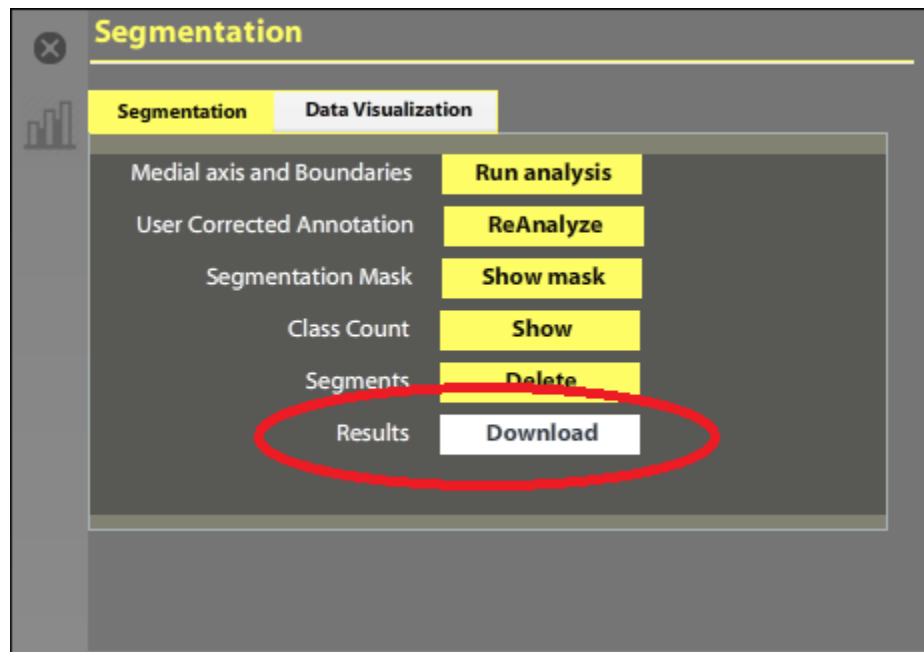


Figure 9.7 GUI for downloading results

# Chapter 10

## Conclusion & Future Work

### 10.1 Summary

Rich Internet Applications (RIAs) facilitate physical, audible and visual interaction. FireFly fulfills these goals by supporting labelling, segmentation, visualization, classification and tracking. After studying various technologies for interfacing, PERL was used to bridge the gap between FireFly and MATLAB. It has been extended to generate automated segmentation results by interfacing through CGI. This feature helped us to run our analysis on microvasculature images and extract graphical analysis on medial axis, boundary and branch points. Data analytics was an important feature, as it gave a platform to analyze and visualize data in the form of graphs. The current segmentation editing (add, delete, split and merge) and labeling features are being used by the National Institute of Health for generation of ground truth for lung segmentation [5] and labeling chest X-rays. NIH is also utilizing FireFly's capabilities for malaria infected cell counting. With the support of various importing/exporting formats, it is easier to work and transfer data to other applications. Additional GUI tools have also made FireFly a more user-friendly application for image annotation.

The new data set lock saving and other GUI changes has increased FireFly's performance drastically. It allows researchers and doctors to efficiently collaborate by having more control of the system with individual accounts and sharing capabilities.

The researchers and doctors can now work collaboratively and solve more complex problems; they also have more control of the system with the user specific annotation levels. This is helping them to work concurrently on the same images but with different set of annotations. Some of the future works that can help Firefly to be more efficient are described below-

## 10.2 Future Work

### 10.2.1 Comparison of Different Segmentation Results

FireFly currently supports segmentation and segmentation editing. However, a separate page to view and compare different segmentation result will be useful. Currently the segmentation algorithms can be run from FireFly and the resulting masks are stored on the server. These resulting masks from various algorithms can be compared and visualized over the web in a better format to provide a better feedback to the researchers for their algorithms.

### 10.2.2 OMERO Interface

OMERO handles a wide variety of formats. The data can be loaded from their servers directly. It maintains a central repository of images. By creating an interface, the data can be retrieved and loaded in the UI. This will provide a better place for data storage and also an easy access to large variety of files. OMERO has identified itself to be one of the biggest libraries and a standard tool for biological microscopic images. It will be a useful enhancement for FireFly to interface with OMERO's backend server to gain access to these images.

### 10.2.3 Big Tiff File Support

The current image loader in FireFly does not support the tiff format. It only supports gif, png and jpg. FireFly version 1 supported pss files, which were high-resolution images. These files were converted to jpg tiles on the server and a respective tile was sent to the client. It used open zoom library to display the tiled images. This library is no longer supported. In order to support big tiff files, an external tiling server is needed. This server will take in big tiff images and tile them depending on the zoom level. The client can interact with the server by requesting a particular tile. Once that tile is transferred to the client by HTTP, it can be easily displayed. Deep zoom is another library that can be utilized to display the tiles on the client side.

### 10.2.4 UNDO Function

Currently FireFly supports undo function in only one case: when user has dropped an object and the object is not saved in the database. The user can perform undo function by pressing Ctrl + Z. The objects are deleted in the decreasing order of their creation. A separate data buffer was



created to store each new objectID for the objects dropped, which was later used to delete the objects from the screen buffer. Currently no temporary buffer is maintained in FireFly. All the functions are handled directly without any intermediate storage. For e.g. if the user is drawing a polyline, we cannot undo a single line segment, instead the whole object has to be undone. The approach to solve this problem would be, by introducing a temporary storage buffer, which will store all the steps sequentially. This storage buffer has to also maintain the information of services. Any operation would have to be stored in local cache before sending a service call to PHP services for database update. This would mean that prior to performing any operation; all the details must be stored in a temporary data structure as we may lose all the information when the cache is refreshed with the updates. For e.g. to undo a delete function, the same object with the same information has to be inserted with all the existing properties back in the database. This would create a local storage problem as FireFly is a web based application and only a minimum amount of data must be stored on the local machine. A selective undo approach can also help in reducing the number of steps and complexity. Selective undo will mean only doing undo till the updates are not saved in the database. It will also save storage space in the temporary storage buffer as it will save the data for only a single image.

#### 10.2.5 Single Attribute Panel to Query Information for any Annotation Object

Attribute Panel currently holds the attributes and tracks information of the object. Each object has its own Attribute Panel. Double clicking the object opens the attribute panel for that particular object. Currently for each object a separate Attribute Panel is opened. These Attribute Panels have to be closed each time the user switches the image. A single Attribute Panel is required to automatically update the attributes and track information. With a single Attribute Panel, the user would be able to control all the objects through one panel instead of several Attribute Panels. Since the tracks are also managed through attribute panels, a clever way of handling this situation has to be developed, as multiple tracks can be turned on and off through the Attribute Panel. This feature would provide users a more intuitive way of handling different objects and their track information.

#### 10.2.6 Triggering External (MATLAB, C++) Algorithms and Data Analytics/Cloud Computing

FireFly version 3 supports interfacing with MATLAB for specific datasets. Currently the parameters are exchanged through text files. In future, a more general approach to interface FireFly

with any MATLAB executables is required. This will require some the following points to be taken into consideration -

- a. Exchange of Data-The exchanging of data must be in a standard format like JSON/XML.
- b. API- A proper Application Programming Interface has to be provided through which the users can interface easily.

This will give a more generic approach to the users for interfacing. There is also a need for a GUI based uploading of executables, which is currently uploaded manually.

Another feature that can be added is running of executables on a public or private cloud. The user should be able to upload the data and executables on his personal or private cloud. FireFly can run the executable on his cloud and fetch the data through web and store in its database or store in user's database. This approach will make FireFly more scalable. Currently FireFly's services are hosted on MERU server but in the future to make it more scalable the services can be pushed to a public cloud. We can use cloud as Platform as a Service or can host our own cloud by utilizing Infrastructure as a Service. Each of these approaches would help us to enhance FireFly's functionalities.

#### 10.2.7 Conversion to Flex4

In comparison to Flex 3, the new features in Flex 4 are certainly much more advantageous. However, migrating to Flex 4 is a challenge. Features that were introduced in Flex 4 [9] are -

1. Integration with Adobe Catalyst- Adobe wanted to present the world with a new designer tool for creating Rich Internet Applications, Adobe Catalyst. While developing Flex 4, they mainly focused on creating a platform for this new tool. This new tool separated the work of designers and developers allowing them to continue working in their old ways while also efficiently collaborating.
2. Spark Component Architecture- Flex 4 introduces new spark architecture. The components model, core logic, skinning and layout are broken and they act autonomously. Most of the components from Flex 3 can be used in Flex 4 as the new model is built upon Halo model. UI component has been extended to make new classes. Each version of Flex has all the components needed to build an application like data-grid, buttons, layout containers etc.

and they are bundled together in libraries. The introduction of new Spark architecture is mainly going to lay out the foundation for Adobe Catalyst. The major change from the old Halo architecture is the introduction of separation levels. Spark affects package implements the animation effects. It allows more flexibility to create animation for random objects. The effects have been re-implemented in Flex 4 while maintaining backward compatibility with Flex 3 allowing the old effects to be used at the same time.

3. MXML 2009- MXML is a XML based language used for laying out components on the interface. Numerous changes have been introduced to support the new Spark architecture while also maintaining the support for Halo components. MXML 2009 continues to support the old core, skinning and layout along with the newer Spark components.
4. Improvements to View States- A simple state change used to produce change in the View state. It was introduced in Flex 2. This feature has been simplified in Flex 4 and is now easier to understand and use. IncludeIn and ExcludeIn can be used on the attributes to manipulate them based on the change of state.
5. FXG Support- with the FXG support in Flash player, the designers will be able to design the components in Catalyst or CS4 illustrator and it can be imported as it is without any modifications. This will assist the developers to use the components that were designed outside Flex. FXG is the declarative used by a designer to create components.
6. Skinning Enhancements- In order to modify the visual components independently, Spark component has now changed skins to control all visual aspects. The components can be modified outside the core logic as it resides outside the component core.
7. Updated Layout Model- The majority of containers used in Flex 3 like HBox and VBox, which were used for horizontal and vertical layouts, are not included in the Gumbo library. The layout is now handled through delegation.

Migrating from Flex 3 to Flex 4 has immense benefits, however, it can be difficult at the same time since FireFly is a developed application and all the current components are not supported in Flex 4. Migrating would be beneficial in the long run but will also need a careful approach. A different branch has been created in SVN codebase and slowly the components are being migrated.

10.2.8 Support for Mobile Computing Platforms like iOS, Android, etc.

Currently FireFly has been tested and used only on web browsers. It does run on tablets that Flash support on their browsers. In iPad’s case, the mobile version of Flash required high memory so is not supported. However, FireFly can be run using third party web browsers on iPads. Another way FireFly can be run on mobile platforms is by developing the client side in Android or iOS. Currently web services are written in PHP using Zend framework. FireFly’s data interchange is done using AMF (Asynchronous Message format). For mobile device app support, a different data format exchange will be required. Currently, the services are bound to flex calls.

By creating a wrapper of REST services on top of our existing services, we can support any kind of GET, PUT, POST and DELETE REST calls. These REST services will interact with our existing Zend services. This internal exchange between the existing services and REST services can be done in any format like AMF, JSON, and XML. A support of JSON or XML will enable us to interact with any mobile application. Although, the existing services can be utilized but the client side still has to be rewritten completely for Android or iOS. The tentative architecture is shown in Figure 10.1.

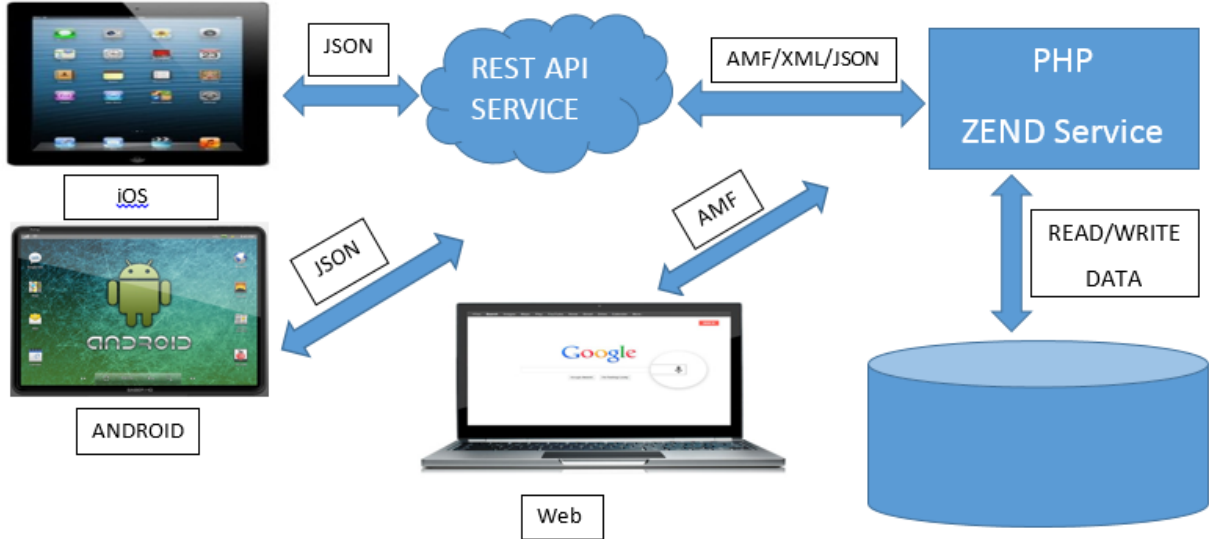


Figure 10.1 Support for mobile devices

# Appendix A

## Development and Debugging in FireFly

FireFly is built on Adobe Flex3, PHP, MySQL and PERL. It uses Adobe Flex 3 on the client side and PHP on server side. There are two development IDE's that can be used for developing in Flex3-

1. Adobe Flash Builder- It was previously known as Adobe Flex builder. Flash Builder was built on top of eclipse. It can be downloaded from:

[https://www.adobe.com/cfusion/tdrc/index.cfm?product=flash\\_builder](https://www.adobe.com/cfusion/tdrc/index.cfm?product=flash_builder)

2. Eclipse- It can be also used for Flex development with the flash builder plugin. Eclipse can be downloaded from:

<http://download.eclipse.org/eclipse/downloads/>

The plugin can be downloaded from:

<http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliosr2>

FireFly is compiled using Flex 3.5 SDK. Adobe Flash Builder already contains this SDK.

For Eclipse it can be downloaded from:

<http://opensource.adobe.com/wiki/display/flexsdk/Downloads>

Flash player is required to run the compiled SWF file. Latest version can be downloaded from:

<http://get.adobe.com/flashplayer/>

Links for tutorial and documentation on Flex:

- <http://www.adobe.com/support/documentation/en/flex/flex3.html>
- <http://blog.flexexamples.com>
- <http://cookbooks.adobe.com/flex>

FireFly's codebase is maintained using SVN. Currently FireFly v3.0 is in the HEAD version. Flash builder or Eclipse can be connected by enabling the SVN plugin in the IDE or connected directly through tortoise SVN.

The link shows how to install SVN in Flash builder.

[http://blogs.adobe.com/jasons/2010/03/installing\\_subclipse\\_in\\_flash\\_builder\\_4.html](http://blogs.adobe.com/jasons/2010/03/installing_subclipse_in_flash_builder_4.html)

## PHP

The IDE that can be used for PHP development is Adobe Dreamweaver. It can be downloaded from <http://www.adobe.com/products/dreamweaver.html>

With the Apache server the services can be hosted locally. Depending on the operating system LAMP, WAMP, XAMP can be used to host services locally. PHP code is currently deployed on MERU server. The code can be checked out locally connecting through SECUREFX FTP client. To direct the request to the local services instead of MERU the path in service-config-file has to be changed to local host.

## PERL

Eclipse PERL plugin can be useful for PERL development and it can be downloaded from <http://www.epic-ide.org/download.php>. Strawberry PERL is the MS windows version of PERL; it is already equipped with everything needed for running in a windows machine. It has been designed very close to PERL environment on UNIX systems. Apache server needs to be enabled before running any PERL scripts.

## Database

FireFly uses MySQL database to store all the information. Database can be connected through MySQL Workbench or Toad. Database resides on MERU and it can be connected by pawprint and password. For running scripts on MERU putty can be used. For loading images or any other files directly SECUREFX can be used.

## Debugging of FireFly

Firefox plugin firebug is a beneficial tool to test any GUI bug. This plugin throws an exception when it comes across any error on the browser. Charles debugger is used for testing and debugging AMF requests and responses. It is a monitoring tool used for observing different request and responses.

# Appendix B

## FireFly Manual

The home page of FireFly is shown in Figure B.1. The user can login by using his username and password.

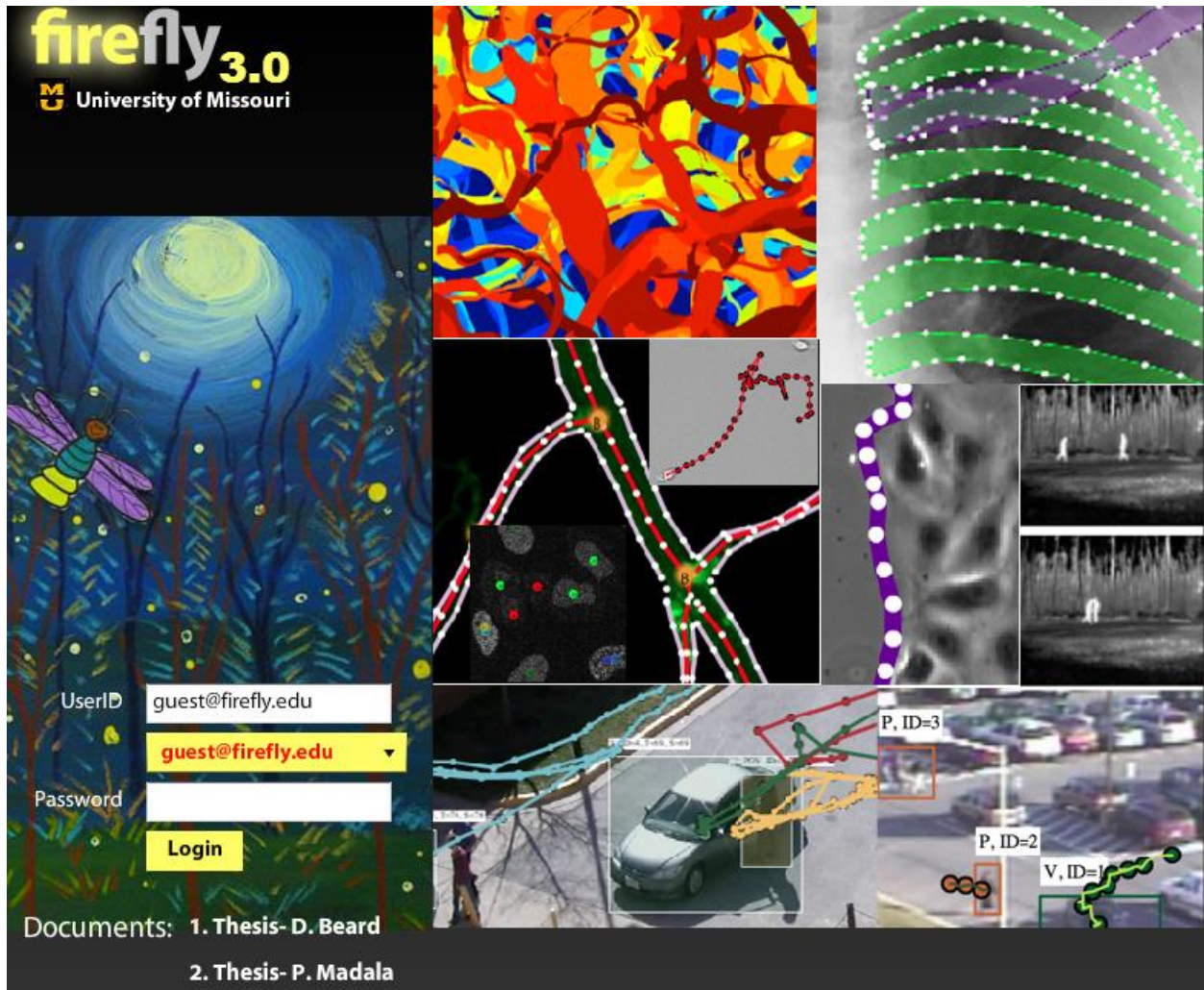


Figure B.1 FireFly login page



**firefly 3.0**  
University of Missouri

**Labs :**

Lab ID	Lab Name	Description
1	U.Missouri_cs	Biomedical
2	U.Missouri_cs	Surveillance

**Projects :**

ProjectID	Creator	ProjectTitle
5	1	Princeton Shaevitz Bacteria
10	1	Cornelison Satellite Cells
12	1	UMASS wadsworth Wound healing
16	1	NKTraining_original
17	1	Vessel_original

**ImageSet :**

Imageset	resolutionWidth	resolutionHeight	Number of Frames
Vessels_segments/	1360	1036	67
Vessels_n/	1360	1036	9
intact_mice1_012606_erbw	2993	2993	10
intact_mice1_012706_erbk	2993	2993	10

**Annotation :**

AnnotationID	AnnotationTitle	Perm	Edit
31	Vessel_test	R/W	Open

Figure B.2 User lab menu

Figure B.2 shows the User lab menu. It is divided into four parts- Lab, Projects, Imageset, and Annotation. The user must select the lab in order to view the associated projects. After selecting the project, the underlying image sets will be displayed. Each image set can have different set of annotations. User's R\W access depends upon the access level of the user in the dataset. If dataset is being used by any other user, it will be displayed as locked in order to prevent concurrent editing of the same dataset.

Figure B.3 shows the workspace in FireFly. It has six panels that are expandable and collapsible. It also has an attribute panel which is only displayed after a double click on the objects. The position of the image can be changed anytime using normal mouse movements. The panels are described below.



Figure B.3 Workspace

## Class Chooser Panel

This Panel contains all the classes which are contained in a given project. The user has to select one of the classes and start marking the objects on the screen. The objects on the screen will be marked with the color of the class selected. By default first class in the panel is selected. The check box on the left side shows the classes that are currently visible on the screen. Unchecking the check box will make the objects of that class disappear from the screen. All the classes can be made invisible by unchecking the show all check box. Checking show all will make them visible again. Count show button shows a column chart with number of objects in each class. This graph auto updates itself as we switch the frame.

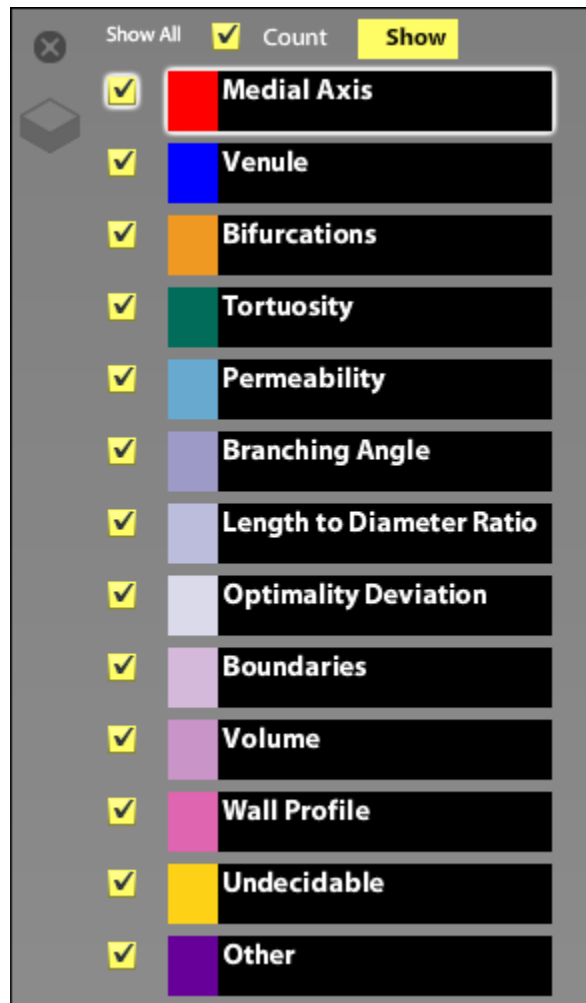


Figure B.4 Class chooser panel

## Frame Advance Panel

This panel contains all the movements, zooming options, buffer and other information related to frames. It contains the lab, project, annotation name and access level of the user at the top.

It has four direction buttons to move the image in four directions. The input box to set the zoom factor, the plus and minus buttons can be used to zoom in and out. Ctrl + and Ctrl - can also be used to zoom in and zoom out from the keyboard. It also contains another direction button at the bottom, which is used to navigate to next and previous frames. It also contains the input box in the middle which can be used as navigation to a specific frame. Left and right arrow on the keyboard can also be used for a advancing to different frames. The bar at the bottom shows the number of images buffered. Bad Frame check box is used to mark bad frames.

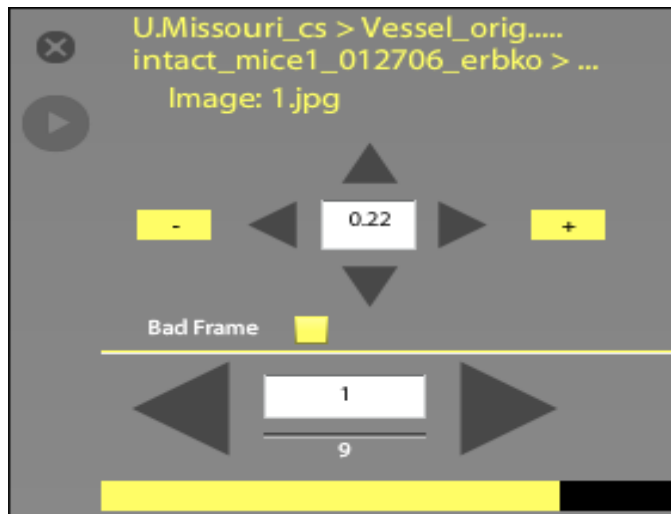


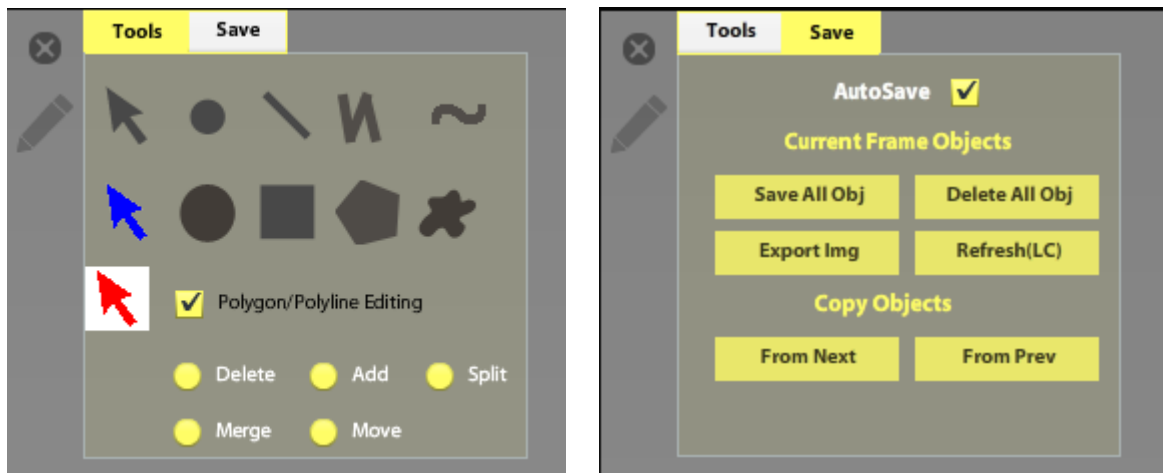
Figure B.5 Frame advance panel

## Drawing Tools Panel

It has two tabs Tools and Save. The tool tab has all the drawing tools. FireFly currently supports Point, Line, Box, Polyline, Polygon, Curve, Circle, and Freeform. The black arrow indicates default tool select mode. This mode represents no tool is selected and the user can move the image using the same mouse events. Red arrow indicates the Polygon/Polyline editing mode.

Polygon/Polyline editing checkbox enables the user to enter into edit mode. A user can also enter into this mode by pressing Ctrl+P. In this mode, user can edit the polygon/polyline by selecting one of the options add, delete, split and merge. The actions used in editing are done by using LMB click. P/P mode is described in more detail in Chapter 4. Blue arrow indicates tracking mode. Tracking is described in more detail in P. Madala’s thesis [21].

Save tab under Current Frame Objects contains buttons for saving, deleting all objects, exporting the image and saving on local machine. It contains a button Refresh (LC) to refresh cache from updated values in database. Under Copy Objects it contains for buttons copying annotations. It also contains an AutoSave checkbox, which is used to turn the mode on or off.

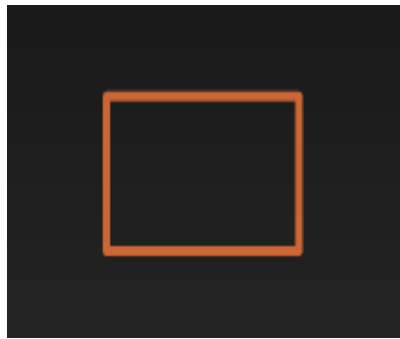


(a) Tools tab

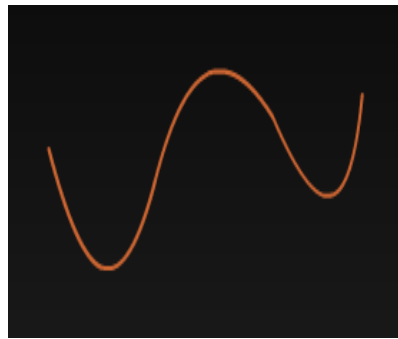
(b) Save tab

Figure B.6 Drawing tool panel

Figure B.7 shows all the drawing objects supported in Firefly. The drawing is interactive and are just drawn by using LMB click and drag events. For completing a Polygon, Polyline, Curve ‘C’ key is used.



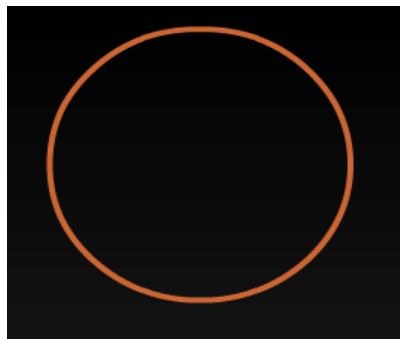
(a) Square



(b) Curve



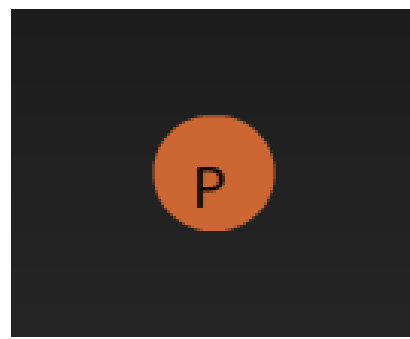
(c) Polyline



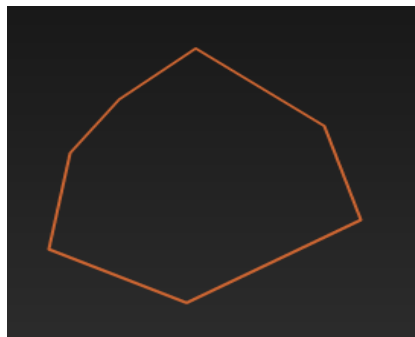
(d) Circle



(e) Freeform



(f) Point



(g) Polygon



(h) Line

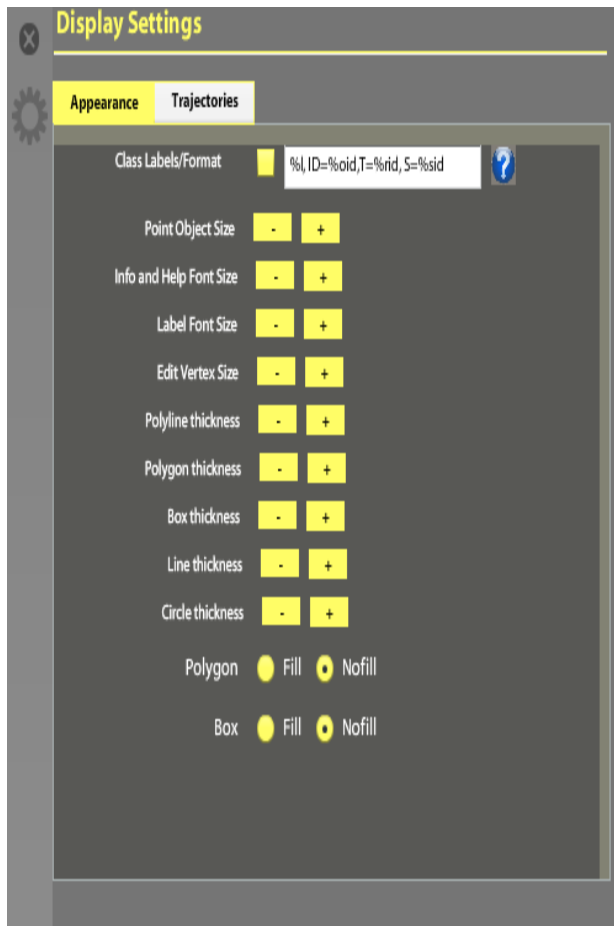
Figure B.7 Different drawing shapes in FireFly

### Display Panel

This Panel is divided into two tabs. Trajectories, Appearance. Trajectories contains all the functions related to trajectories such as number of points, sampling, head and tail, trajectory color,

trajectory thickness, trajectory points size. trajectory length etc, All the options are used for making changes in trajectory.

Appearance tab consists of functions related with the appearances. It contains class label-turning on/off, control thickness of- line, circle, polyline, polygons, boundary fill option for square and polygons. It also contains point object size, edit point size, class label/format and info and font size control.



(a) Appearance tab



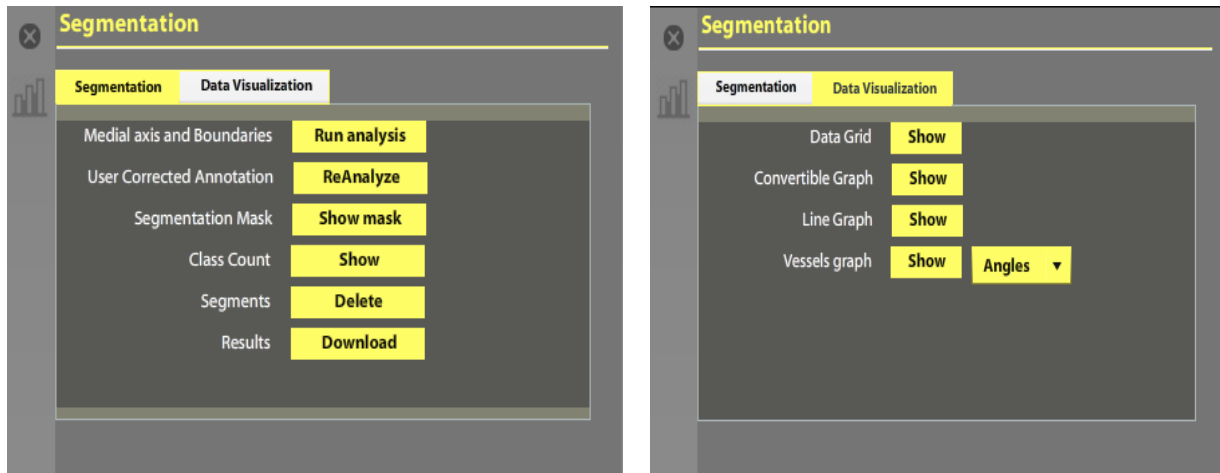
(b) Trajectory tab

Figure B.8 Save/Display panel

## Data Analytics Panel

This panel contains all the analysis options to be done on images and data. Running the analysis would invoke segmentation algorithm on the server and results are produced which are then loaded on image. Graph analysis computes different values for Angle, Curvature and Tortuosity and displays in the form of graphs. Segmentation mask displays the actual image and the segmentation mask on top of each other. By varying the alpha it can be utilized to monitor the accuracy of algorithm, Segments delete, deletes all the segments on the screen. Results download, downloads all the result on the system. Class Count shows count of objects in a class in the form of graphs.

Data Visualization contains the tools for visualization of data. FireFly currently supports bar graph, pie chart, line chart, data grid for visualization of data.



(a) Segmentation Analysis Tab

(b) Data Visualization Tab

Figure B.9 Data analytics panel



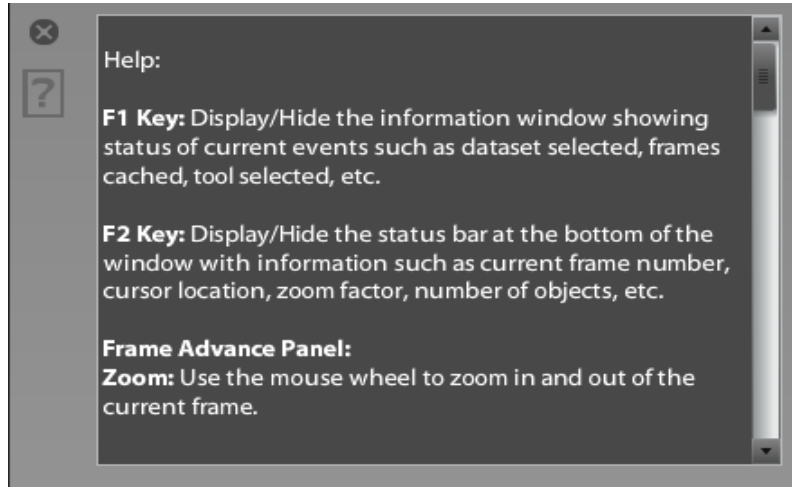
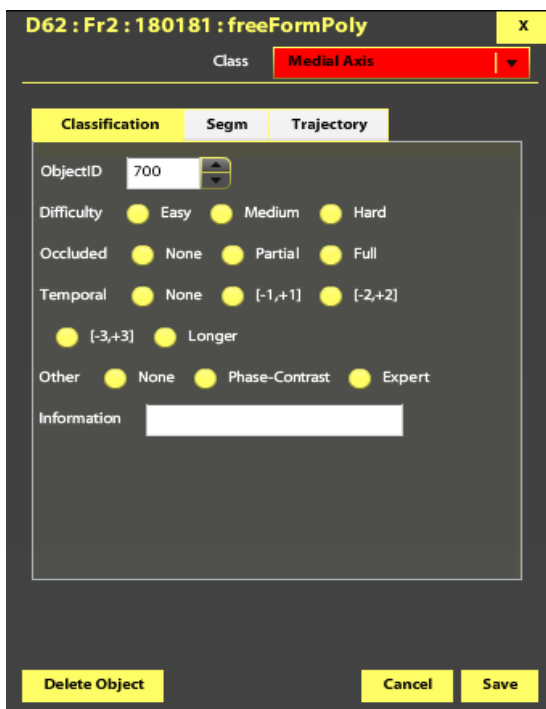
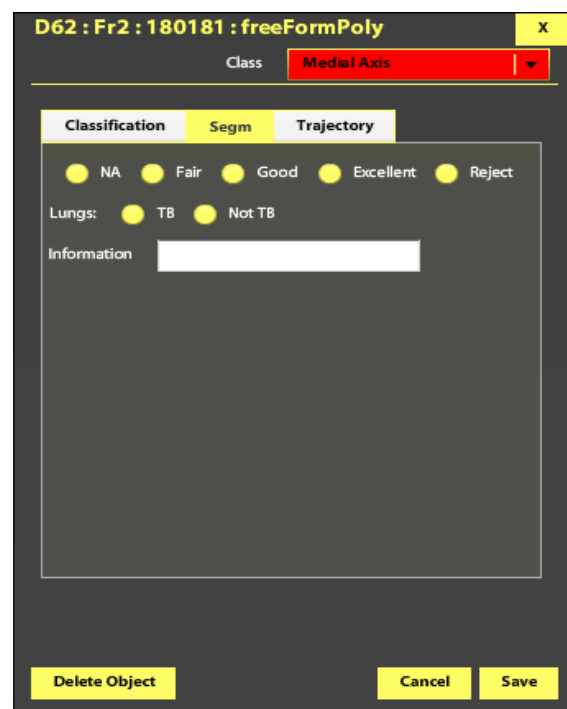


Figure B.10 Help panel



(a) Classification tab



(b) Segmentation tab



(c)Trajectory tab

Figure B.11 Attribute window

## Attribute Window

This contains all the attributes related to the objects, track and segmentation. This panel has been described in more detail in P. Madala's thesis [21]. It contains 3 tabs Classification, Segm, Trajectory. Classification contains all the attributes related to classification of the object. Segm has options related to segmentation. Trajectories tab contains all the attributes related to trajectories of the object. The user can input it manually or change interactively. The class can be change from the drop down menu. It has an option to show the track. The KW18 files can be downloaded from the panel.



Figure B.12 Debug window

## Debug Window

Debug Window shows all the debug messages. Whenever any operation is happening, it prints a message in the Debug window. It can be turned on and off using F1 key or Ctrl+F1 key.

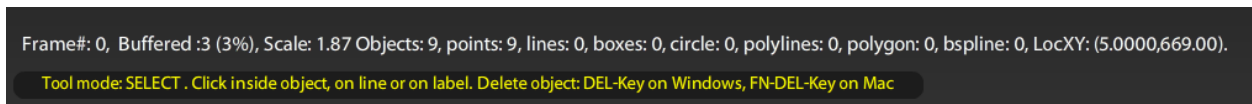


Figure B. 13 Info and message window

The text in white shows the information of the objects, frame, scale, and location. It can be turned on or turn off using F2 or Ctrl + F2. The text in yellow shows different messages to provide user a feedback of the operations happening.

# Bibliography

- [1] A. Haridas, R. Pelapur, J. Fraser, F. Bunyak, K. Palaniappan, “Visualization of automated and manual trajectories in wide-area motion imagery”, 15th Int. Conf. Information Visualization, London, 2011, 288-293
- [2] C. Sommer, C. Strahle, U. Kothe, F. A. Hamprecht, “Ilastik: Interactive Learning and Segmentation Toolkit”, 8th IEEE International Symposium on Biomedical Imaging (ISBI). Proceedings, 2011, 230-233
- [3] S. Jaeger, A. Karargyris, S. Candemir, L. Folio, J. Siegelman, F. Callaghan, X. Zhiyun K. Palaniappan, R. K. Singh, S. Antani, G. Thoma, Y.-X. Wang, P.-X. Lu, C. J. McDonald, "Automatic Tuberculosis Screening Using Chest Radiographs", IEEE Transactions on Medical Imaging, 2014, , Vol. 33, No. 2, 233-245
- [4] A. E. Carpenter, T. R. Jones, Michael R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, P. Golland, D. M. Sabatini: “CellProfiler: image analysis software for identifying and quantifying cell phenotypes”, Genome Biology, 2006, R100
- [5] S. Candemir, S. Jaeger, K. Palaniappan, J. P. Musco, R. Singh, Z. Xue, A. Karargyris, S. Antani, G. Thoma, and C. J. McDonald, “Lung Segmentation in Chest Radiographs Using Anatomical Atlases with Non-rigid Registration,” IEEE Transactions on Medical Imaging, 2014, Vol. 33, No. 2, 577-590
- [6] J.R. Anderson, S. Mohammed, B. Grimm, B. W. Jones, P. Koshevoy, T. Tasdizen, R. Whitaker, R. E. Marc: “The Viking viewer for connectomics: scalable multi-user annotation and summarization of large volume data sets”, Journal of Microscopy, Jan. 2011, 13-28

- [7] Colin Eberhardt, "Flex, Silverlight Or Html5? Time To Decide", White Paper, 2011, <http://www.scottlogic.com/blog/archive/2011/05/Flex-Silverlight-HTML5.pdf>, (Accessed on 25<sup>th</sup> June, 2014)
- [8] A. Kundu, C. Agarwal, A. Chandrababu, M. Kumar, K. Ramanarayanan, R. F. Chong. "Getting started with Adobe Flex", IBM Corporation, May 2010, [http://public.dhe.ibm.com/software/dw/db2/expressc/wiki/Getting\\_Started\\_with\\_Adobe\\_Flex\\_p2.pdf](http://public.dhe.ibm.com/software/dw/db2/expressc/wiki/Getting_Started_with_Adobe_Flex_p2.pdf)
- [9] J. Rose, "Top 10 Changes in Flex 4", May 2009, <http://www.infoq.com/articles/top-10-flex4-changes>, (Accessed on 25<sup>th</sup> June, 2014)
- [10] "Flex Cairngorm Architecture Overview – Part 1", <http://www.flamelab.de/article/flex-cairngorm-architecture-overview-part-1/>, (Accessed on 25<sup>th</sup> June, 2014)
- [11] "Home - Swiz Enterprise Framework.", <https://swizframework.jira.com/wiki/display/SWIZ/Home>, (Accessed on 25<sup>th</sup> June, 2014)
- [12] D. Doermann and D. Mihalcik. "Tools and techniques for video performance evaluation" In int. conf. Pattern Recognition, Volume 4,167-170.
- [13] "Google Charts.", <https://developers.google.com/chart/interactive/docs/index>. (Accessed on 25<sup>th</sup> June, 2014)
- [14] "FLOT Examples.", <http://www.FLOTcharts.org>, (Accessed on 10th December , 2012)
- [15] " Raphael" , <http://www.raphaeljs.com>, (Accessed on 10th December , 2012)
- [16] "D3.js - Data-Driven Documents.", <http://D3JS.org>, (Accessed on 25<sup>th</sup> June, 2012)

- [17] "Flare",  
<http://flare.prefuse.org/>, (Accessed on 10th December , 2012)
- [18] "Livedocs" ,  
[http://livedocs.adobe.com/flex/3/html/Part7\\_DataVis\\_1.html](http://livedocs.adobe.com/flex/3/html/Part7_DataVis_1.html) (Accessed on 18<sup>th</sup>  
December, 2014)
- [19] V. B. S. Prasath, O. Haddad, F. Bunyak, O. Glinskii, V. Glinsky, V. Huxley, K. Palaniappan, "Robust filtering based segmentation and analysis of Dura Mater vessel laminae using epifluorescence microscopy", 35th Annual International Conference EMBS (IEEE EMBS/EMBC 2013), Osaka, Japan, 2013, 6055-6058.
- [20] D. Beard, "FireFly- web based interactive tool for the visualization and validation of Image Processing algorithms", Master's thesis, Department of Computer Science, University of Missouri-Columbia, 2009.
- [21] P. Madala," Interactive Web Based track editing and Management", Master's thesis, Department of Computer Science, University of Missouri-Columbia, 2012.
- [22] F. D. Chaumont, S. Dallongeville. ; J. Olivo-Marin., "Icy: a new open-source community image processing software ", IEEE International Symposium on Biomedical Imaging, 2011, 234 - 237
- [23] F. D. Chaumont, S. Dallongeville, N. Chenouard, N. Herve, S. Pop, T. Provoost ,V. Meas-Yedid ,P. Pankajakshan , T. Lecomte , Y. L. Montagner , T. Lagache, A. Dufour, J.C. Olivo-Marin, " Icy: an open bioimage informatics platform for extended reproducible research", Nature Methods, 2012. 690-696
- [24] A. Haridas, " KOLAM: A Software Platform for Interactive Visualisation & Analysis of High Resolution Microscopy Imagery", figshare, 2014,  
<http://dx.doi.org/10.6084/m9.figshare.704387> (Accessed Aug 01, 2014)

- [25] V. B. S. Prasath, O. Haddad, F. Bunyak, R. K. Singh, O. Glinskii, V. Glinsky, V. Huxley, K. Palaniappan. "Computerized Dura Mater Laminae analysis of fluorescence microscopy images", Figshare: <http://dx.doi.org/10.6084/m9.figshare.661533>.
- [26] "Flex Coordinate System",  
[http://help.adobe.com/en\\_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf69084-7de0.html](http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf69084-7de0.html) (Accessed on Aug 4<sup>th</sup>, 2014)
- [27] R. Pelapur, V. B. S. Prasath, F. Bunyak, O. V. Glinskii, V. V. Glinsky, V. H. Huxley, and K. Palaniappan," Dura Mater Microvasculature Segmentation and Analysis in Epifluorescence Microscopy Images", 36th Annual International Conference EMBS (IEEE EMBS/EMBC 2014), Chicago, USA 2014
- [28] "How to use your CGI-bin",  
<http://www.parkansky.com/tutorials/bdlogcgi.htm> (Accessed on June 4<sup>th</sup>,2014)
- [29] Nouman Naveed, " Using Cairngorm with Adobe Flex-Part 1",  
<http://flexatom.com/?p=166>( Accessed on June 4<sup>th</sup>,2014)
- [30] B. Bryant, H. Sari-Sarraf, M. Wachtel, R. Long, and S. Antani, "Fast GPU-Based Segmentation of H&E Stained Squamous Epithelium from Multi-Gigapixel Tiled Virtual Slides," SPIE Medical Imaging, Lake Buena Vista, FL, February 2013.
- [31] Advanced Virtual Microscope  
[http://benbryan.info/AVM\\_Client\\_Web/](http://benbryan.info/AVM_Client_Web/) (Accessed on Aug 25<sup>th</sup>, 2014)
- [32] Carl Vondrick, Donald Patterson, Deva Ramanan. "Efficiently Scaling Up Crowdsourced Video Annotation" International Journal of Computer Vision (IJCV). June 2012.