MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

CENG 491 – INITIAL DESIGN REPORT

MİLSOFT - PHOTOGRAMMETRY LAB PROJECT [PHOTOLAB]

BAD SECTOR

Hanife ÜNAL

Meryem AYAS

Serap ATILGAN

Serra Sinem TEKİROĞLU

3 December 2007

TABLE OF CONTENTS

1 INTRODUCTION

During the research on requirement analysis report, we had started to gain experience on photogrammetry. With the drawing diagrams and specification of requirements, the general design of PHOTOLAB was started to be constructed. In order to demonstrate the initial design of our product PHOTOLAB as a group 'Bad Sector' ,we prepare this report.As a result of our researches, we become aware of what basic functionalities we are required to implement and what features can we add our product.
While preparing initial design report, requirement analysis report will be a main guide for us.In order to clarify every single detail of the reader's mind ,we explained our design process via various diagrams. We have used Data Flow Diagrams, Class Diagrams,Sequence Diagrams, State-Transition and ER Diagrams etc.According to the feedback of our project assistant,we corrected our ER diagrams in initial design report.

1.1 Project Definition and Scope

Nowadays, in all over the world, new projects have been launched for the Unmanned Air Vehicle (UAV) Systems whose role in defense concepts is becoming more important, and the expenses for UAV systems are increasing above the expectations.

UAVs can do exploration, observation and target detection with the help of a camera that is mounted on them. The images collected by UAVs are just a collection of photos without processing. To make use of these photos, they have to be processed and so become meaningful. This process is called Image Exploitation. Image exploitation, by using some techniques, makes use of image processing algorithms for information extraction.

This project is about Image Exploitation Systems. The data collected from the Unmanned Air Vehicle is not just a collection of 2D pixels, it is a 3D view of the Earth's surface; therefore the image has a depth. In addition, the images may not be taken at the same time or at the same attitude, so there are some difficulties in this project and we need to follow a proper order during this project. The expected four methods and their proper order are as follows:

- **Digital Elevation Model (DEM):** A three-dimensional surface map of an area is usually stored as a grid of elevation points called a Digital Elevation Model (DEM). DEMs are created by collecting elevations and referencing them to corresponding points in the mapped area. The elevations add a Z value to the ground's X and Y horizontal coordinates. So, using a DEM, a user can view an area in three dimensions, giving a clearer understanding of the problem.
- **Orthophoto:** An orthophoto is an aerial photograph that has been geometrically corrected to remove geometric distortion caused by camera tilt and differences in elevation. The digital orthophoto is a photographic image of the terrain - but more importantly, it is true to scale and therefore accurate distances and areas can be measured. These orthophoto images are well suited for detail planning and analysis of what exists on the ground. In order to acquire correct orthophoto, we need to generate a precise and accurate DEM of the Earth's surface.
- **Mosaic:** Many problems require finding the coordinate transformation between two images of the same scene or object. One of them is Image Mosaicing. It is important to have a precise description of the coordinate transformation between a pair of images. Image mosaics are collection of overlapping images together with coordinate transformations that relate the different image coordinate systems. By applying the appropriate transformations and merging the overlapping regions of warped images, it is possible to construct a single image covering the entire visible area of the scene. This merged single image is the motivation for the term ``mosaic''.
- **Super Resolution:** Super-resolution is a term for a set of methods of increasing image or video resolution. All these methods are based on same idea: using information from a set of low-resolution images to create one or a set of high-resolution images. These methods try to extract details from frames to reconstruct other frames. SR works when several low resolution images contain slightly different views of the same objects. In this case total information about the object is much higher than information in one frame.

1.2 Application Areas

The images collected from UAVs' cameras have wide range usage areas. UAVs are currently used in a number of military roles such as gaining military or medical information and also they are also used in growing number of civil applications. Some of the application areas of photogrammetry and separately the above four methods are listed below:

- Problems in road design and similar civil and military engineering projects.
- Statistical analysis and comparison of different kinds of terrain.
- Realistic display of landforms for such diverse areas as pilot training, weapons guidance, and landscape architecture.
- In observational astronomy, universities and NASA centers that have small telescope operations can use the methods to a wide variety of research areas.
- Photogrammetry is mainly used in archaeological surveys, urban planning and monument restoration.
- Searching and rescuing humans trapped in collapsed buildings or adrift at sea.
- In security issues like scenes of murders, burglars or smugglings, or in damage detection issues.
- In computer applications, such as games, atlases, and encyclopedias.
- In industrial applications such as shipbuilding, traffic engineering, mining engineer, automobile construction, structures and buildings.

1.3 System Requirements

We examined the system requirements in two parts: hardware requirements and software requirements.

1.3.1 Hardware Requirements

While determining the hardware requirements, we look after the similar software programs that have been developed till now. The software products that we mentioned at market research part generally had similar hardware requirements. The following requirements for hardware seem to be minimal and enough for our project development:

- P4 or Equivalent Processor
- Minimum 512 MB RAM
- 3D Graphics Card
- Minimum 40 GB HDD

4.1.2 Software Requirements

We will use several tools for different phases during development of the project. The following requirements during documentation and development phases are necessary for project.

Documentation Phase

- Microsoft Office 2007
- Milestone Professional 2008
  - For drawing Gantt Charts for the reports. It is more functional than SmartDraw while drawing Gantt Charts so we preferred Milestone Professional 2008.
- Diagram Studio 5.3
  - For drawing the diagrams and charts for the reports we were using SmartDraw. However, from now on we draw our diagrams in Diagram Studio since it is more flexible.
- Adobe Acrobat Reader
  - For writing and submitting the reports. We write our reports on Microsoft Office 2007 and submit them after we convert it to pdf format.

Development Phase

- Windows XP and Linux
    - PHOTOLAB must be running operating system independent, so we will work on both Windows and Linux.
- Microsoft Visual Studio .NET 2003
    - From our previous experiences, we are used to work on Visual Studio .NET. Because the language we will implement PHOTOLAB is C++, and .NET provides easy usage, we preferred it.
- wxWidgets GUI Toolkit
    - We will use wxWidgets in GUI design. Software is advised by Milsoft and because it is working with C++, we preferred it.
- Microsoft Visual Assist
    - It is an add-in for Visual Studio .NET and provides generally more accurate and complete code suggestions. It also has the syntax highlighting feature.
- OpenGL Library
    - Since it can be run under both Windows and Linux platforms and it is compatible with C++, we preferred OpenGL libraries for 2D and 3D image visualization.
- OSSIM (Open Source Software Image Map)
    - We preferred it since OSSIM is a high performance software system for image processing, geographical information systems and photogrammetry.

2 DESIGN CONSTRAINTS

The main design constraints that may be a potentially thread for our project are as follows:

2.1 Time Constraints

For the successful completion of the project at the end of the this year, time constraints are very sufficient. According to our plan, PHOTOLAB has to be finished at the end of May 2008. As a group we scheduled our project by considering the course syllabus. We have to release a prototype demo at the end of this semester. Since Milsoft gives assignments regularly, we already started onto the prototype of the project so we have no doubt about the deadline of prototype submission. However, since it is a hard and a long project, in order to minimize the potential risks of not completing the project in time, as a group we need to follow a very strict schedule.We update our detailed time schedule regularly, so as long as, the group members follow the schedule on the Gannt Chart, the project could be completed successfully.

2.2 Software Constraints

Our sponsored company Milsoft gives us two options about the programming language.We can choose either Java or C++. When we consider the design of graphical user interface, Java is more flexible. In C++ making GUI is not easy since we are not experienced before however when we make a research about the libraries that we can use in our project,we decided that C++ provides more convenient libraries. In addition, Milsoft's first programming language choice is C++ and some of the group members are not experienced much in Java.After a long consideration period as a group we decided to develop our project in C++. For designing the GUI we are using wxWidgets GUI Toolkit however since there is not enough documentation about wxWidgets in the internet, for the time being its help file is sufficient. We hope that if we encounter some problems Milsoft provides us necessary documentation.

2.3 Performance Constraints

While developing the product, one of our main aim is the product must provide an understandable and easy usage. In addition, the execution time of the program is very

important in our product.So performance is very important constraint for us.While deciding the algorithms we use, we must choose the algorithms that are executed fastly because some of the features we planned to add requires fast execution such as mosaicking the videos. For example, while mosaicking the images, although homography method is much easier to implement, we plan not to use this method because of the performance issues.

2.4 Experiences and Skills of Member Constraints

Since the product is developed by our group members, as developers, our programming and design skills and experiences is also one of the restrictions. Our experience is very limited because we did not have made such software projects before. Our project requires many technical information. Moreover, some group members are not familiar with the diagrams. So in order to solve this drawbacks, we may consult experienced people to get help about solving problems.We have to manage the problems as quick as we can in order to complete the project in time.

3 ARCHITECTURAL and COMPONENT LEVEL DESIGNS

3.1 Photolab Modules

Photolab is designed on modular base for a systematic architecture. There are four modules in Photolab; namely, GUI, Photogrammetry, and File System Module. These modules are connected to each other according to data traffic.

3.1.1 GUI Module

We are planning to use “wxWidget” for the whole Project. Therefore at first phase we will use “wxWidget” in graphical design components. This cross-platform open-source GUI library ensures that Photolab will be a platform-independent software. It simplifies the image related operations. GUI module is central part of the modular architecture. Simply it

connects user , Photogrammetry and File System modules. User sends operation requests and image/video (File) identifier by the help of GUI module. GUI distributes requests and delivers their responce to the user. Besides, the simple works like zoom in/out or rotate image will be done under GUI module architecture.

3.1.2 File System Module

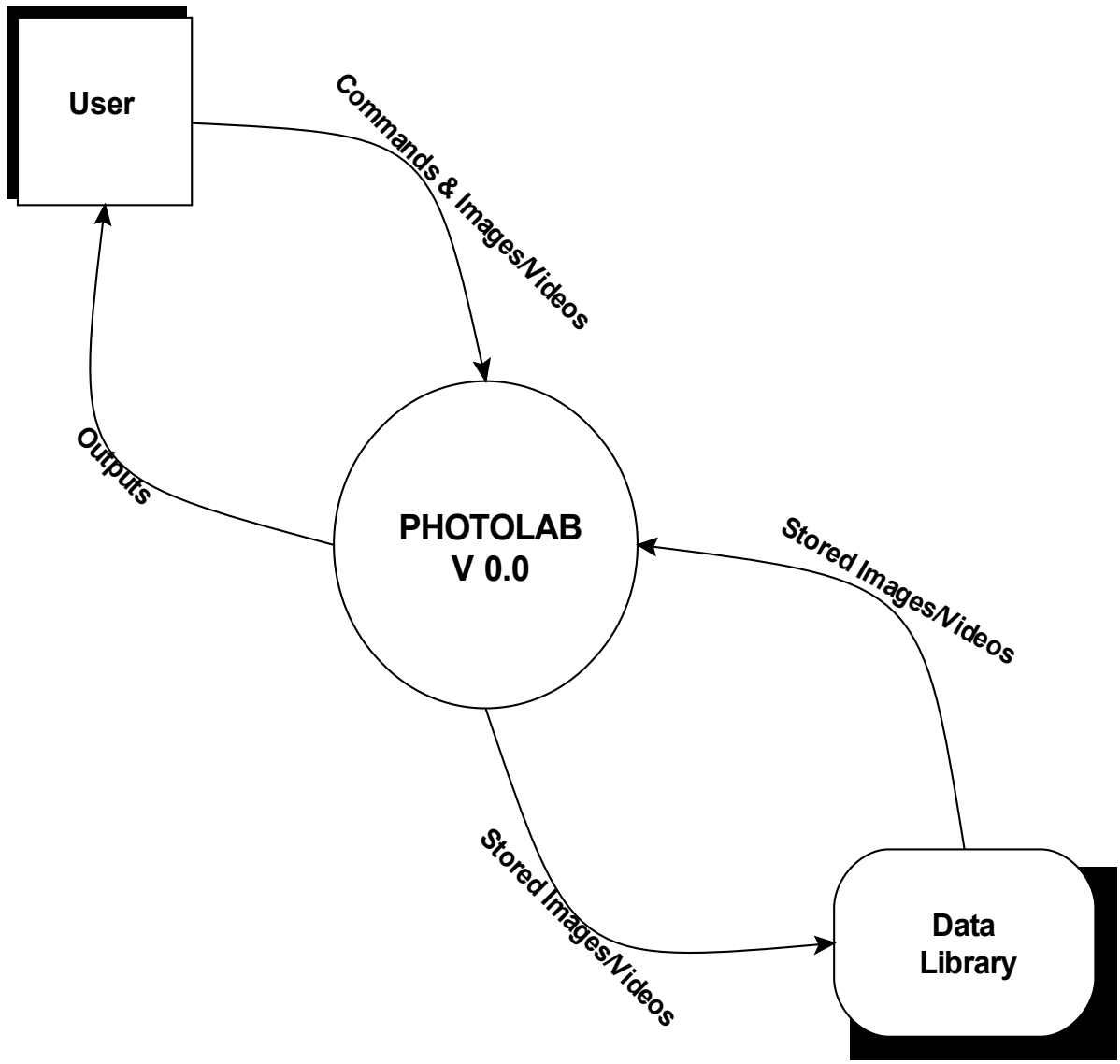
We especially pay attention to File Sytem Module. This module is designed to handle file operations correctly. According to process on multiple and large images, file operations must be managed in most efficient way. File system module has two parts. First one is a basic database-like image library and second one is operator part. Operator part will connect the image library with other modules. Photolab is designed for working on a variety of image formats. Therefore file system module must handle these file formats.

3.1.3 Photogrammetry Module

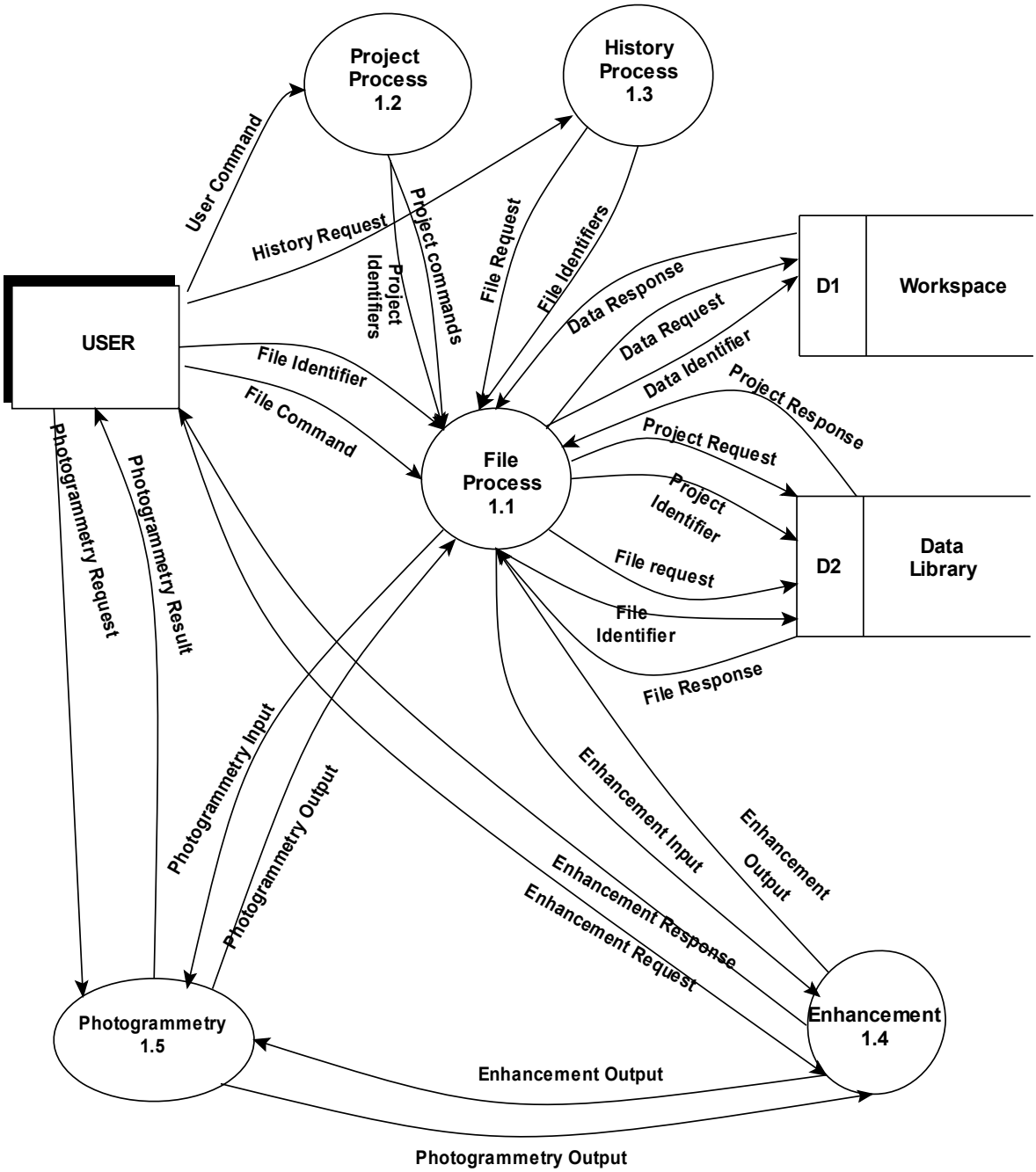
Photogrammetry Module mainly has four parts. Each part will do different jobs and works with different inputs. For simplicity we designed this module as two levels. DEM, Orthophoto, Mosaic, and Super Resolution classes has row algorithms and works on images in bottom level. In order to give expected output, the second level of this module must convert the row output into utilizable by other modules and classes according to user expectations.

3.2 Data Flow Diagram

3.2.1 Level 0 DFD

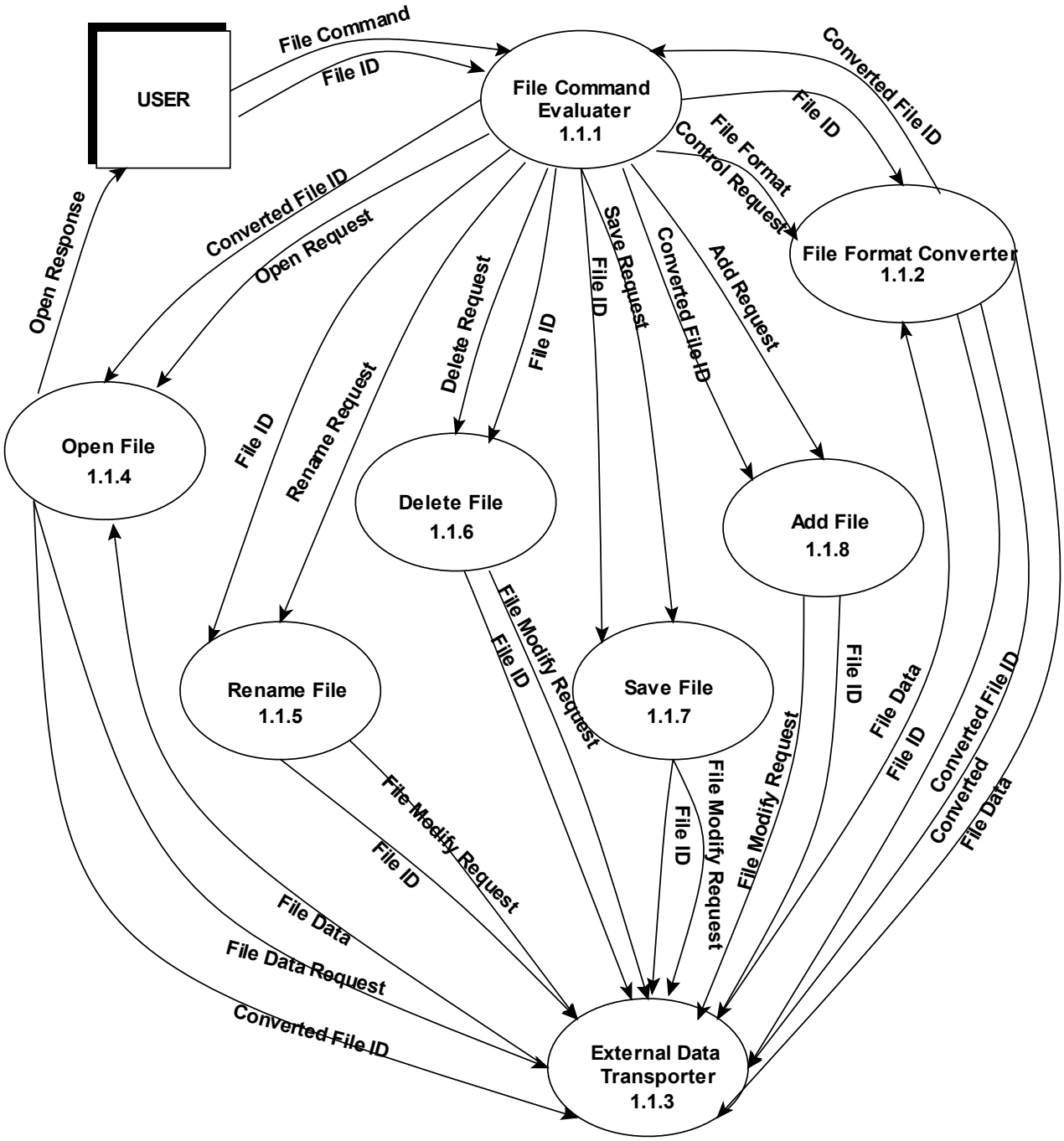


3.2.2 Level 1 DFD



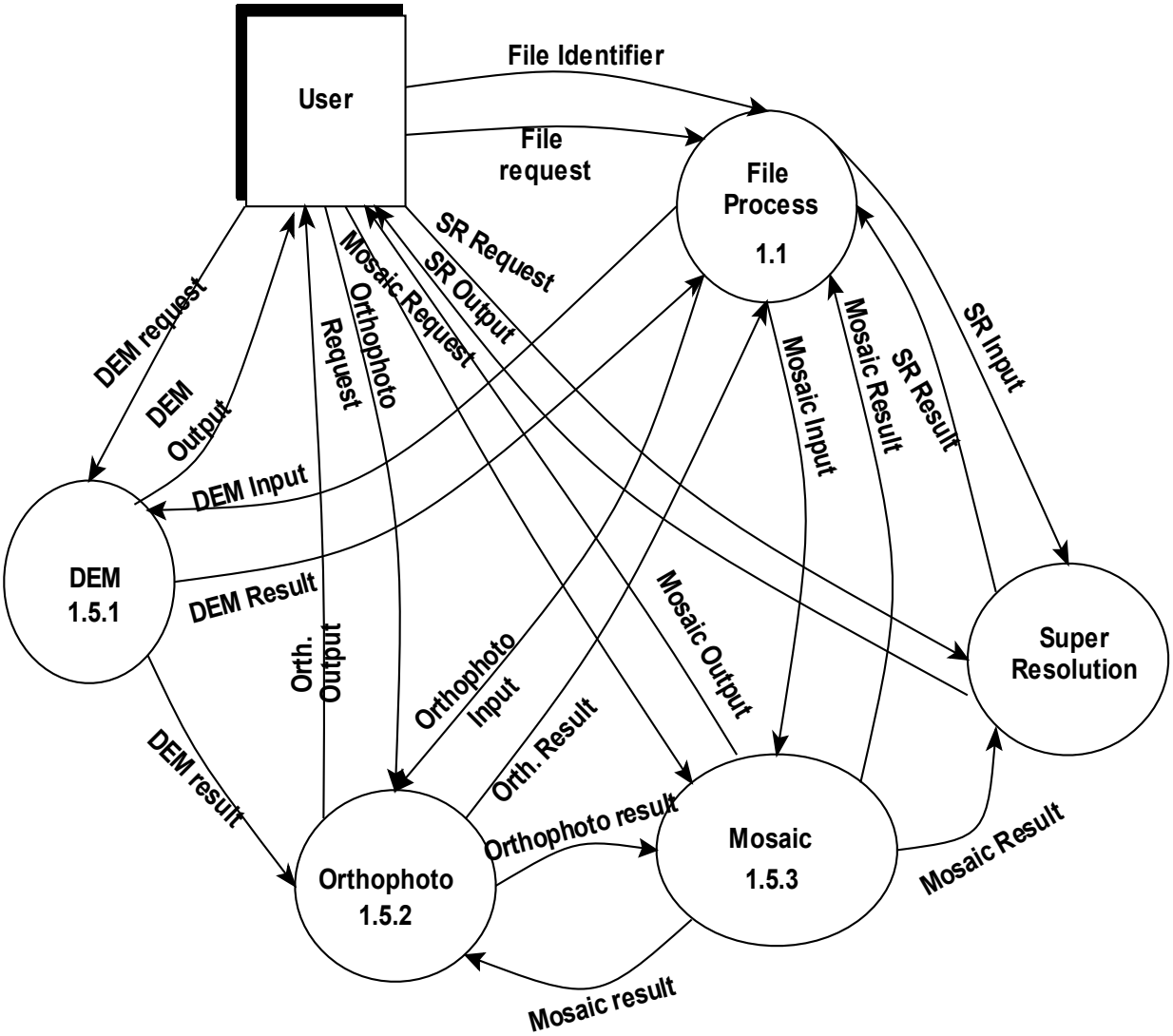
3.2.3 Level 2 DFD

File System Process



3.2.4 Level 2 DFD

Photogrammetry Process



3.3 Data Dictionary

Name	File Identifier
Where & How Used	File Process(1.1) INPUT Data Library(D2) INPUT History Process(1.3) OUTPUT
Description	Identifier for the file consists of name of file and name of Project it belongs to.

Name	File Command
Where & How Used	File Process(1.1) INPUT USER OUTPUT
Description	Commands for file operations of adding, opening, deleting, saving and renaming.

Name	File Request
Where & How Used	File Process(1.1) INPUT Data Library(D2) INPUT History Process(1.3) OUTPUT
Description	Request for the file from the file process with file identifier.

Name	File Response
Where & How Used	File Process(1.1) INPUT Data Library(D2) OUTPUT
Description	Response of the data library to the file process.

Name	Project Identifier
Where & How Used	Data Library(D2) INPUT Project Process(1.2) OUTPUT File Process(1.1) INPUT & OUTPUT
Description	Identifier of the project for the files it includes.

Name	Project Commands
Where & How Used	File Process(1.1) INPUT Project Process(1.2) OUTPUT
Description	Commands for opening, closing, updating, deleting, saving, displaying projects.

Name	Project Request
Where & How Used	Data Library(D2) INPUT

	File Process(1.1) OUTPUT
Description	Request for the project from the file process with project identifier.

Name	Project Response
Where & How Used	File Process(1.1) INPUT Data Library(D2) OUTPUT
Description	Response of the data library to the file process.

Name	User Command
Where & How Used	Project Process(1.3) INPUT USER OUTPUT
Description	Commands of the user about project.

Name	History Request
Where & How Used	History Process(1.2) INPUT USER OUTPUT
Description	User request to undo the processes done on images.

Name	Data Identifier
Where & How Used	Workspace(D1) INPUT File Process(1.1) OUTPUT
Description	Identifier for data that is a combination of file identifier and a flag of file that defines whether it is an image file, video file or world file.

Name	Data Request
Where & How Used	Workspace(D1) INPUT File Process(1.1) OUTPUT
Description	Request for data with data identifier from workspace.

Name	Data Response
Where & How Used	File Process(1.1) INPUT Workspace(D1) OUTPUT
Description	Response of the workspace to the data request of the file process.

Name	Photogrammetry Request
Where & How Used	Photogrammetry(1.5) INPUT USER OUTPUT
Description	Request of the user for a photogrammetry process on images.

Name	Photogrammetry Result
Where & How Used	USER INPUT Photogrammetry (1.5) OUTPUT
Description	Result of the photogrammetry process on images.

Name	Photogrammetry Input
Where & How Used	Photogrammetry(1.5) INPUT File Process(1.1) OUTPUT
Description	Input files for photogrammetry process.

Name	Photogrammetry Output
Where & How Used	File Process(1.1) INPUT Enhancement(1.4) INPUT Photogrammetry(1.5) OUTPUT
Description	Output files of photogrammetry process.

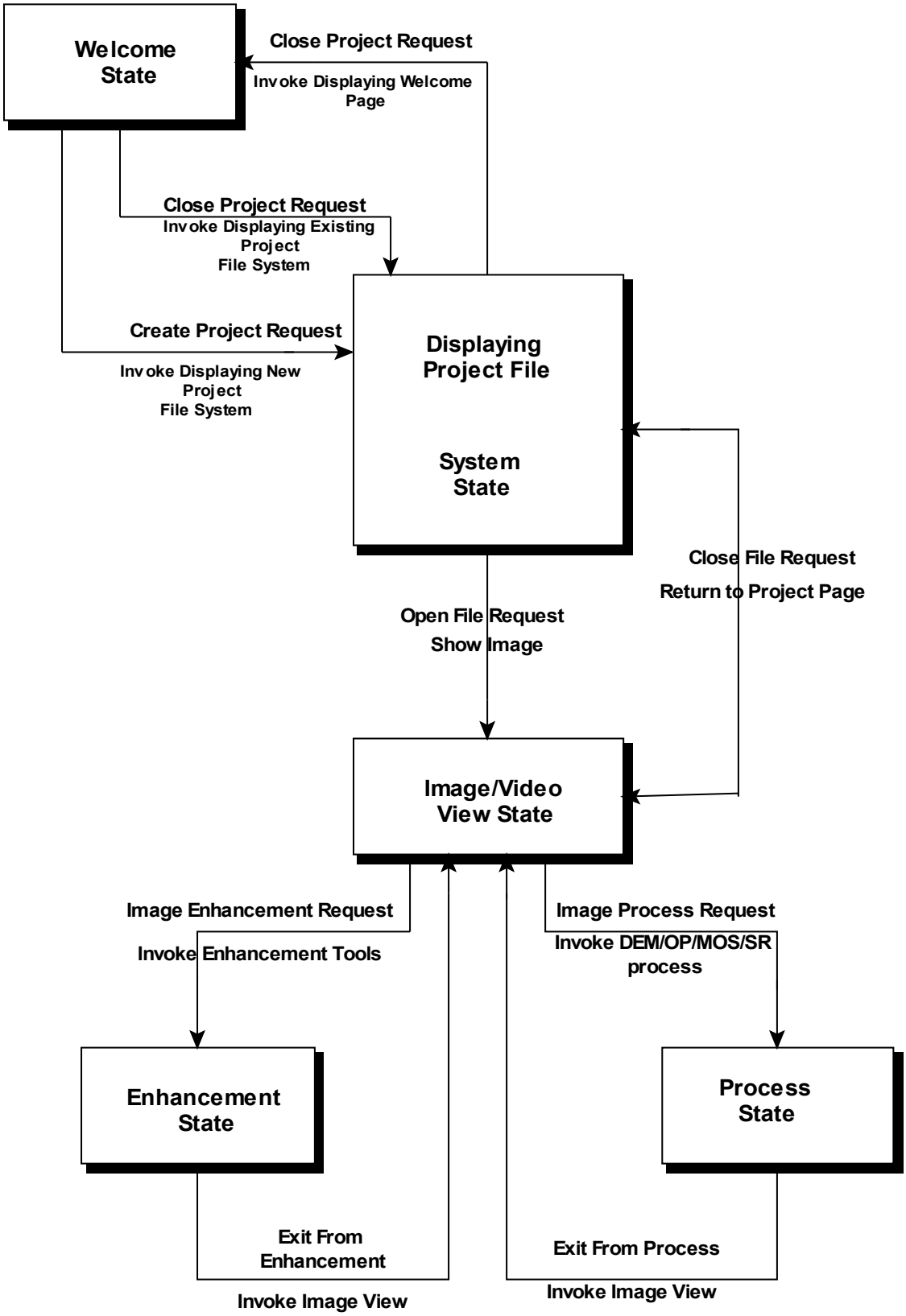
Name	Enhancement Input
Where & How Used	Enhancement(1.4) INPUT File Process(1.1) OUTPUT
Description	Input files for enhancement process.

Name	Enhancement Output
Where & How Used	File Process(1.1) INPUT Photogrammetry(1.5) INPUT Enhancement(1.4) OUTPUT
Description	Output files of enhancement process.

Name	Enhancement Request
Where & How Used	Enhancement(1.4) INPUT USER OUTPUT
Description	Request of the user for an enhancement process on images.

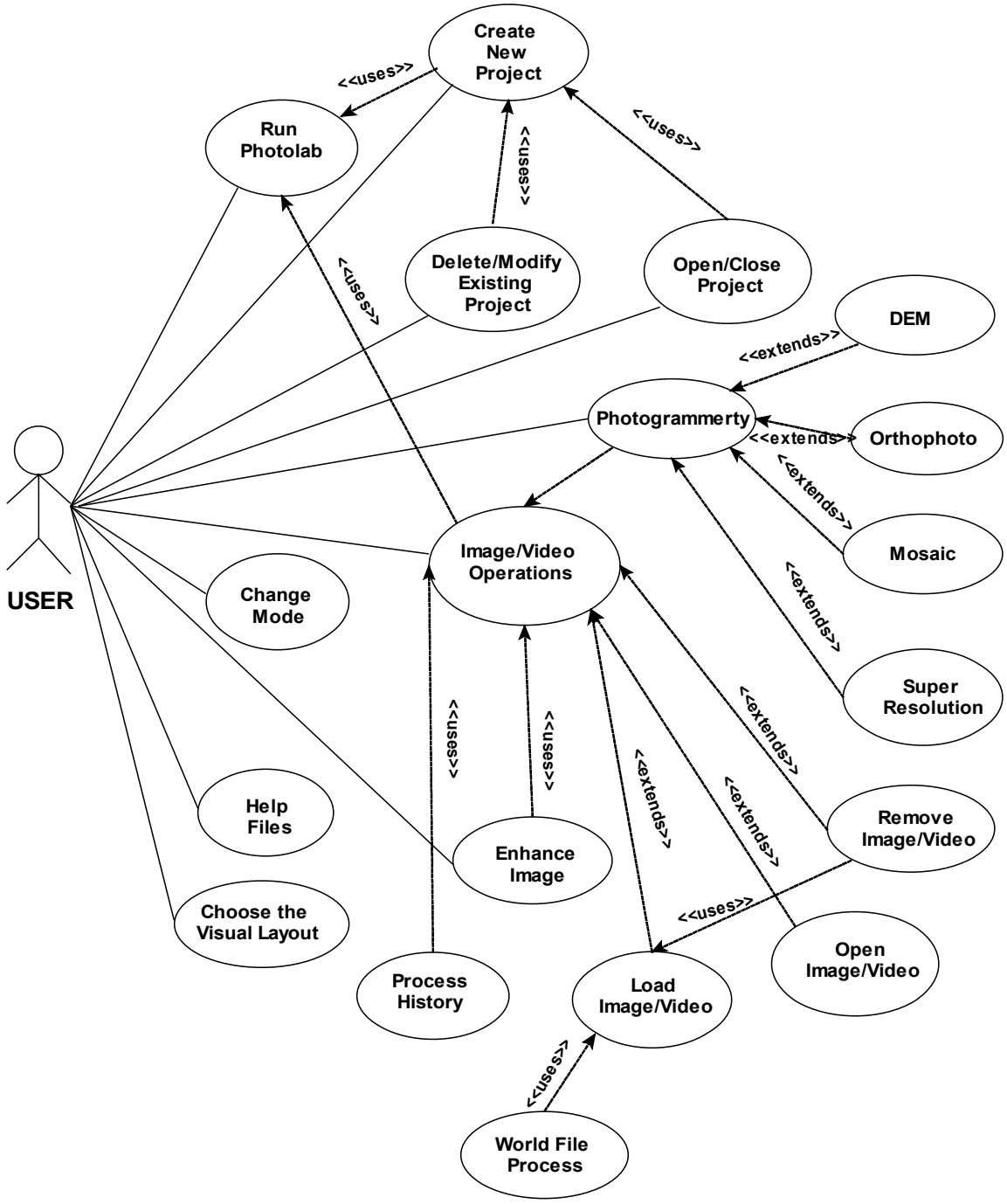
Name	Enhancement Response
Where & How Used	USER INPUT Enhancement(1.4) OUTPUT
Description	Response of the enhancement to the enhancement request.

3.4 State Transition Diagram

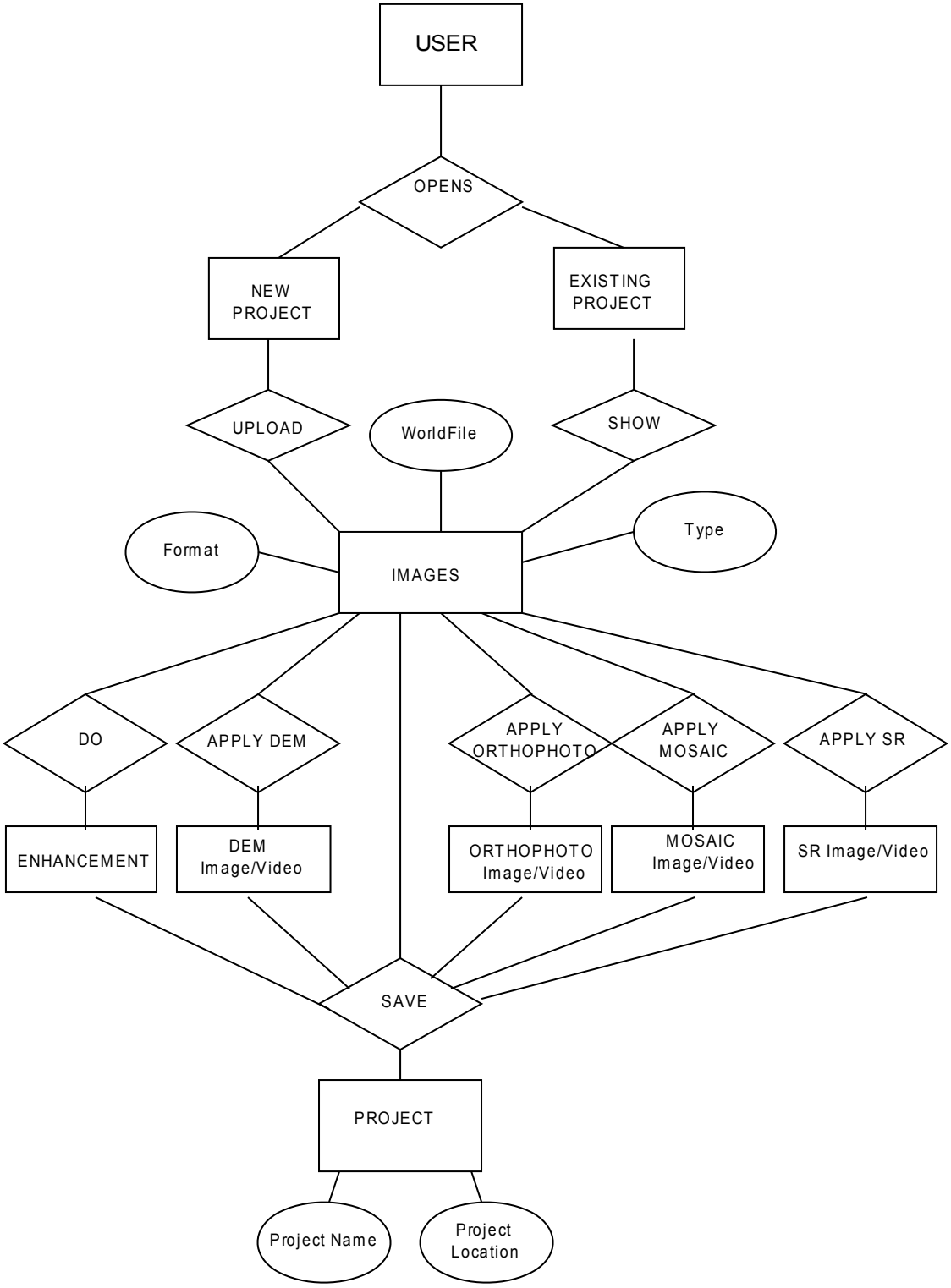


4 OBJECT ORIENTED DIAGRAMS

4.1 Use Case Diagrams

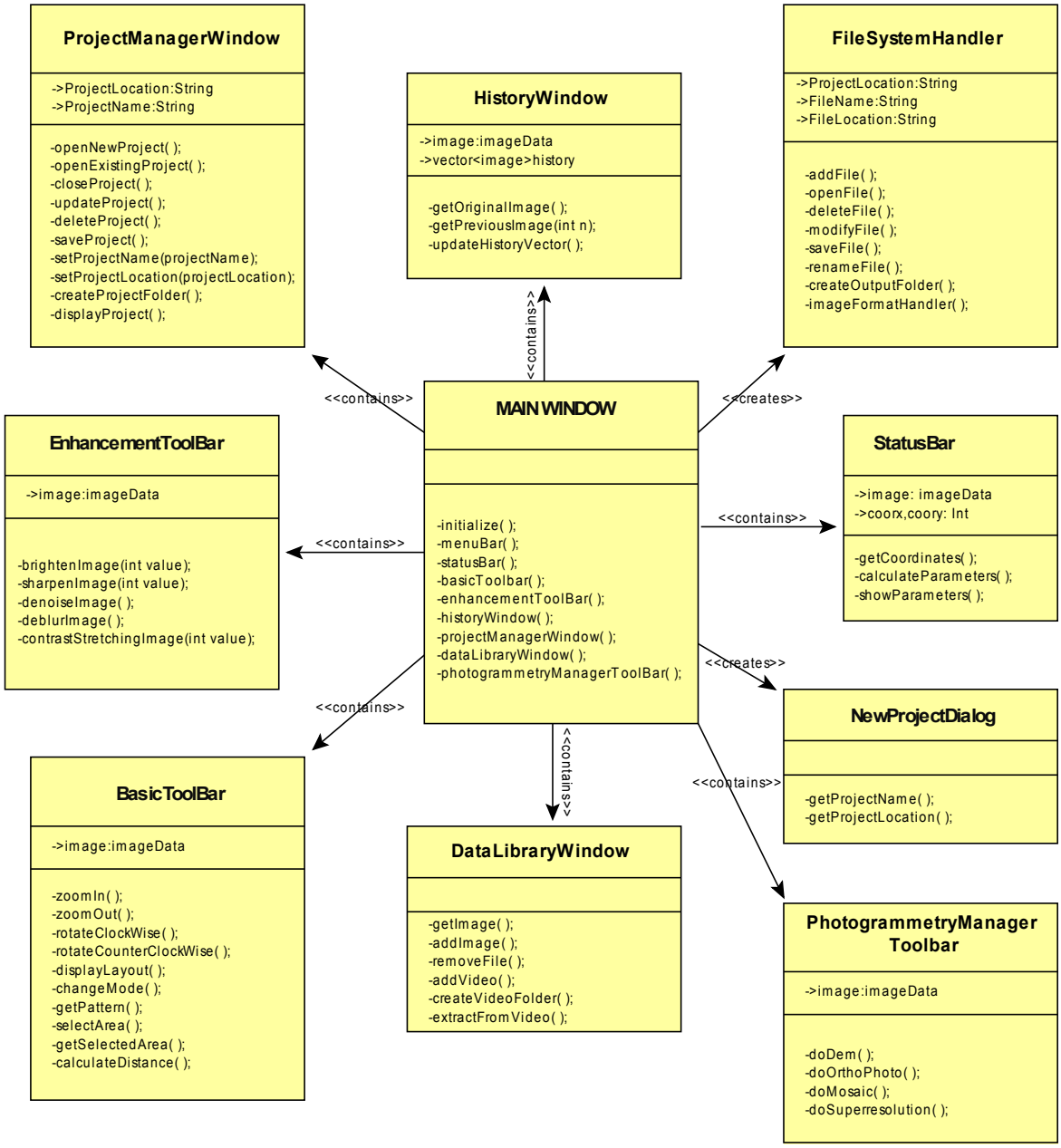


4.2 Entity Relationship Diagram

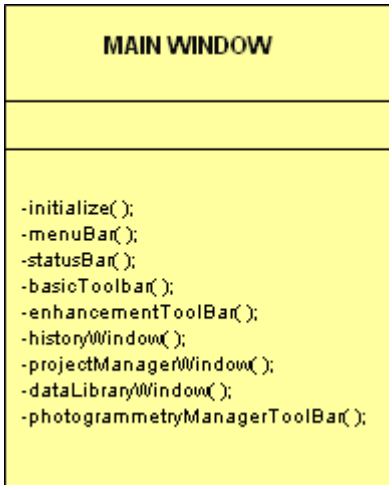


4.3 Class Diagrams

4.3.1 GUI Module and File System Module Class Diagram

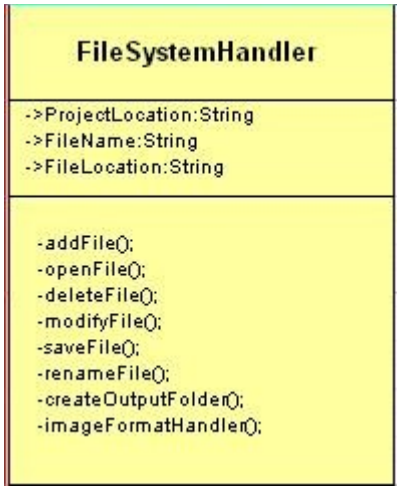


4.3.1.1 Main Window



- **initialize( ):** Function to adjust initial values and properties of main window.
- **menuBar( ):** Function to create an instance of MenuBar class for main window.
- **statusBar( ):** Function to create an instance of StatusBar class for main window.
- **basicToolbar( ):** Function to create an instance of BasicToolBar class for main window.
- **enhancementToolbar( ):** Function to create an instance of EnhancementToolBar class for main window.
- **historyWindow( ):** Function to create an instance of HistoryWindow class for main window.
- **projectManagerWindow( ):** Function to create an instance of ProjectManagerWindow class for main window.
- **dataLibraryWindow( ):** Function to create an instance of DataLibraryWindow class for main window.
- **photogrammetryManagerToolbar( ):** Function to create an instance of PhotogrammetryManagerWindow class for main window.

4.3.1.2 File System Handler



- **ProjectLocation:String :** Stores the location of the project that the current file operations belong to.
- **FileName:String :** Stores the name of the file.
- **FileLocation:String :** Stores the location of the file.
- **addFile( ):** Adds the file with name FileName and location with FileLocation to the project with location ProjectLocation.
- **openFile( ):** Opens the file with name FileName and location with FileLocation.
- **deleteFile( ):** Deletes the file with name FileName and location with FileLocation from the current project
- **saveFile( ):** Saves the file with name FileName and location with FileLocation.
- **renameFile(string NewFileName):** Changes the name of the file from FileName to NewFileName.
- **createOutputFolder( ):** This function is called in case of a process with video files. A new folder is created for video output in data library.
- **imageFormatHandler( ):** This function does the job for converting different supported input file types to bitmap.

4.3.1.3 Project Manager Window

- **ProjectLocation:String :** Stores the location of the project.
- **ProjectName:String :** Stores the name of the project.
- **openNewProject( ):** Creates a new project with the name ProjectName and with the location ProjectLocation.
- **openExistingProject( ):** Opens an existing project with the given project name as ProjectName and project location as ProjectLocation.
- **closeProject( ):** Closes the project that is open in main window without exiting the program.
- **updateProject( ):** Updates the project if there are modifications in it.
- **deleteProject( ):** Deletes the project with the given project name as ProjectName and project location as ProjectLocation.
- **saveProject( ):** Saves the project with the given project name as ProjectName and project location as ProjectLocation.
- **setProjectName(string ProjectName):** Sets the project name as ProjectName. It is triggered by openNewProject( ) method.
- **setProjectLocation(string ProjectLocation):** Sets the location of the project as projectLocation. It is triggered by openNewProject( ) method.

- **createProjectFolder( ):** Creates a folder in specified location set by ProjectLocation, and necessary files are saved under that folder. It is triggered by openNewProject( ) method.
- **displayProject( ):** Displays the project files and other related files in project manager window.

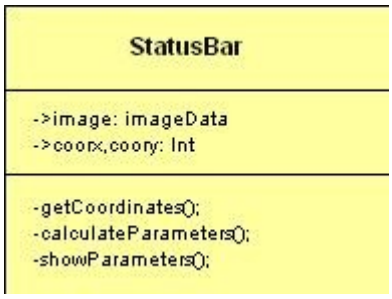
4.3.1.4 History Window

- **image:ImageData :** It is the instance of the ImageData class.
- **vector <ImageData> history :** It is the vector of ImageData class instances that contains the previous 3 versions of the image, to undo the processes that are done on images.
- **getOriginalImage( ):** Resets the processes done on image and loads the original file from data library.
- **getPreviousImage(int n):** It undoes the n processes done on image. The integer number n can not be greater than 3.
- **updateHistoryVector( ):** Updates the history vector if a process is done on image. New version of the image is pushed to stack and the oldest version is deleted if there are more than 3 versions in history vector.

4.3.1.5 Enhancement Toolbar

- **image:ImageData :** It is the instance of the ImageData class.
- **brightenImage(int value):** Do a brightening process with a factor of value on image.
- **sharpenImage(int value):** Do a sharpening process with a factor of value on image.
- **denoiseImage( ):** Do a denoising process on image.
- **deblurImage( ):** Do a debluring process on image.
- **contrastStretchingImage(int value):** Do a contrast stretching process on image with a factor of value.

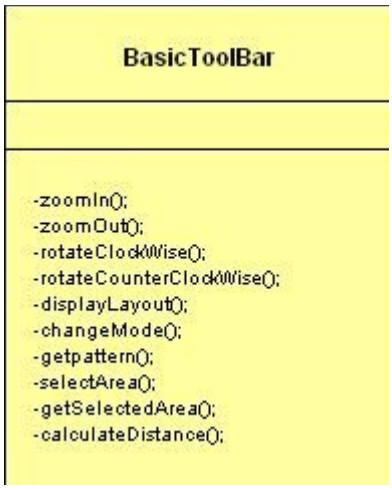
4.3.1.6 Status Bar



- **image:imageData :** It is the instance of the ImageData class.
- **coorx:int :** Stores the x coordinate of the cursor while it is rounding on the image.
- **coory:int :** Stores the x coordinate of the cursor while it is rounding on the image.
- **getCoordinates( ):** Gets the x and y coordinates of the cursor

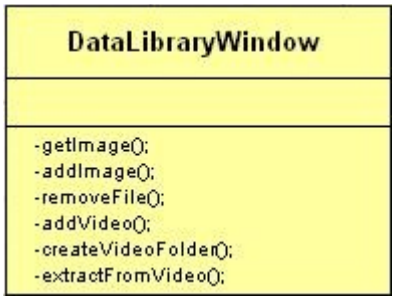
- **calculateParameters( ):** Calculates the UTM coordinates and other parameters according to the x and y values got from getCoordinates function.
- **showParameters( ):** Writes the calculated parameters on status bar.

4.3.1.7 Basic Toolbar



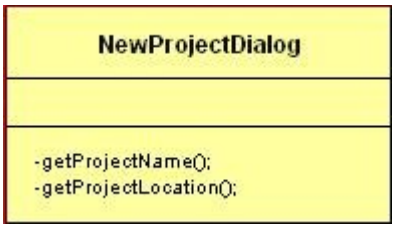
- **image:imageData :** It is the instance of the ImageData class.
- **zoomIn( ):** Does a zoom in process on image.
- **zoomOut( ):** Does a zoom out process on image.
- **rotateClockWise( ):** Rotate the image clockwise with a specified angle.
- **rotateCounterClockWise( ):** The image clockwise with a specified angle.
- **displayLayout(int type):** The layout style is changed according to the type parameter.
- **changeMode( ):** Change the mode from automatic to manual or manual to automatic for detecting tie points.
- **getPattern( ):** Function to enable the user to set a pattern on the image by mouse.
- **calculateDistance( ):** Function that takes the pattern from getPattern method and calculates the distance between two terminals of the pattern.
- **selectArea( ):** Function to enable selecting an area on the image by mouse.
- **getSelectedArea( ):** Function that takes the area selected by selectArea method.

4.3.1.8 Data Library Dialog



- **getImage(imageData image):** Gets an image from data library with properties of image instance.
- **addImage(imageData image):** Adds a new image to data library.
- **removeFile(imageData image):** Removes the image file from the data library.
- **createVideoFolder( ):** The video files are kept under a different folder in data library, so if a video file will be added to library first a video folder should be created. Function creates a video folder.
- **extractFromVideo( ):** The images should be extracted from video stream to be processed. Function extracts images from video stream and saves them in data library.

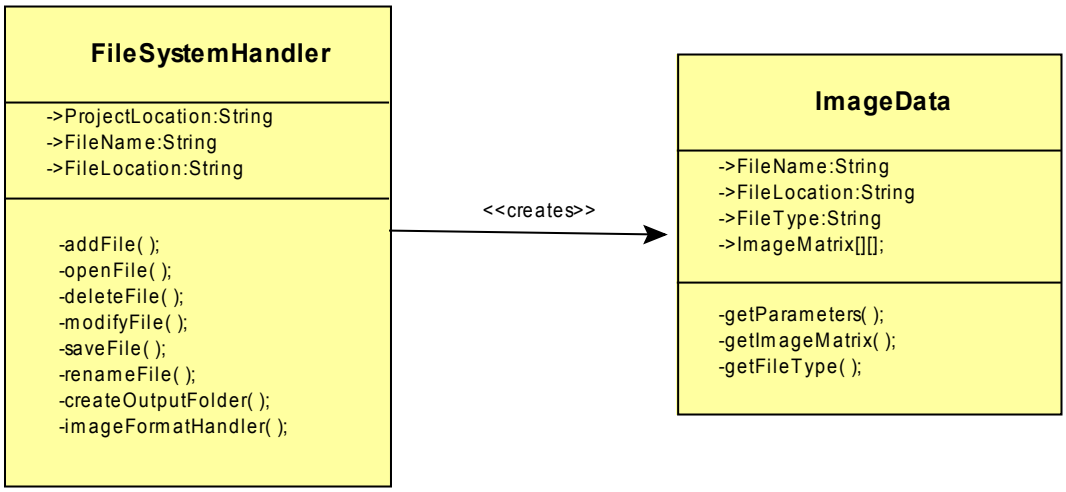
4.3.1.9 New Project Dialog



- **getProjectName( ):** Function gets the name of the project to be created.
- **getProjectLocation( ):** Function gets the location of the project to be created.

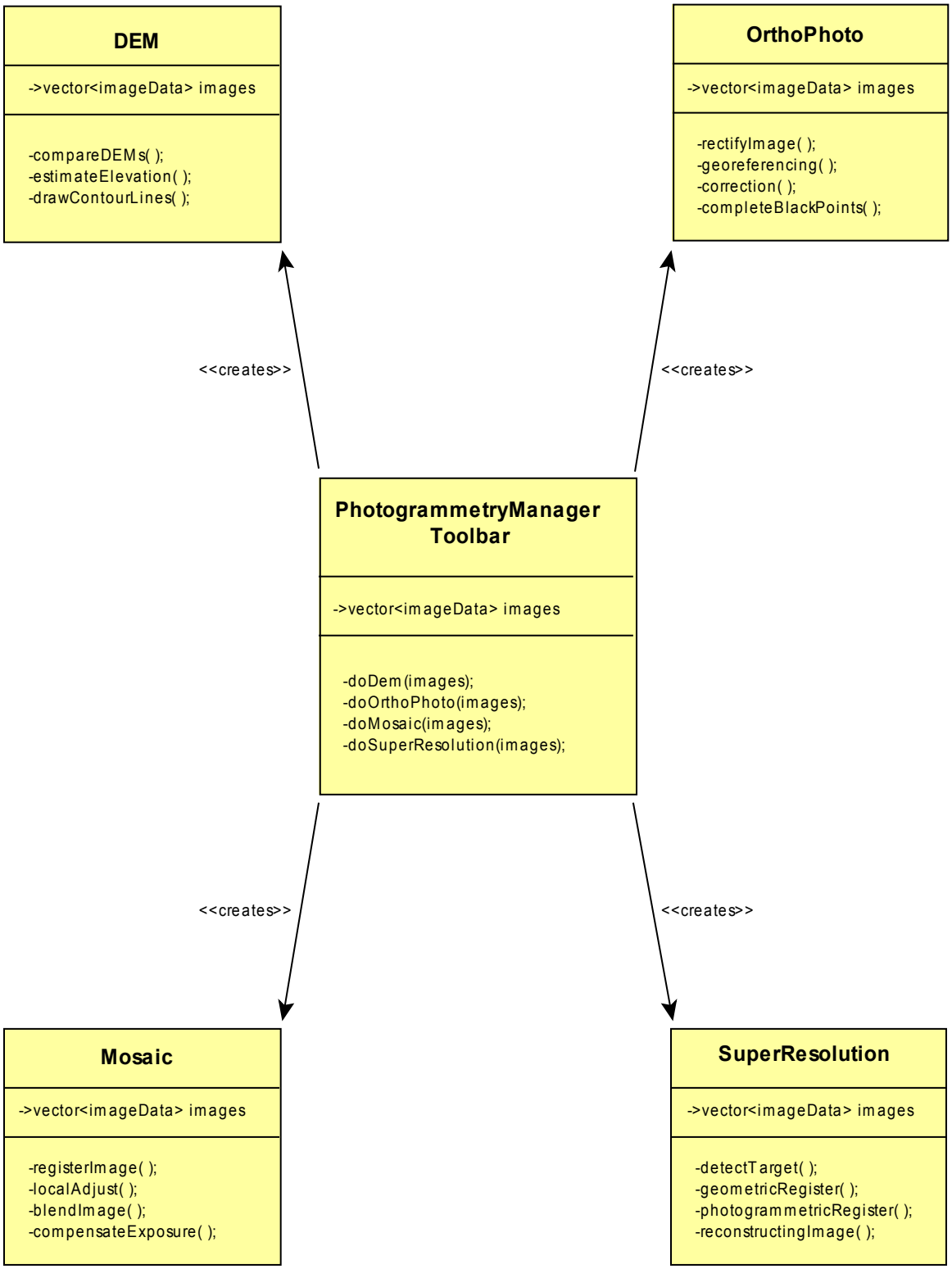
4.3.2 ImageData Class Diagram

.

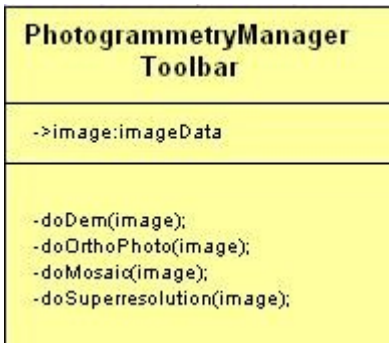


- **FileName:String:** Stores the name of the file.
- **FileLocation:String:** Stores the location of the file.
- **FileType:String:** Stores the type of the file.
- **ImageMatrix[][]:int:** The matrix that holds the image.
- **getParameters( ):** Function gets the parameters from the related world file.
- **getImageMatrix( ):** Function reads the image and stores its bitmap in ImageMatrix.
- **getFileType( ):** Function gets the type of the file and stores it in FileType.

4.3.3 Photogrammetry Class Diagram

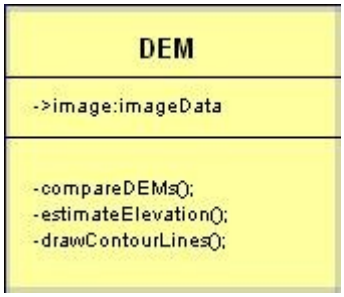


4.3.3.1 Photogrammetry Manager Toolbar



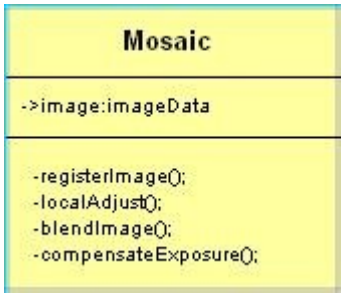
- **image:imageData :** It is the instance of the ImageData class.
- **doDem( ):** Function does the DEM image process on image.
- **doOrthoPhoto( ):** Function does the OrthoPhoto process on image.
- **doMosaic( ):** Function does the mosaic process on image.
- **doSuperResolution( ):** Function does the SuperResolution process on image.

4.3.3.2 DEM



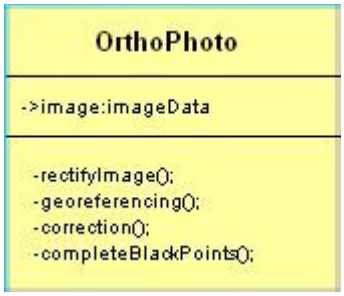
- **image:imageData :** It is the instance of the ImageData class.
- **compareDEMs( ):** Function compares the images for an exact elevation information in image.
- **estimateElevation( ):** Function estimates the elevation of points with the help of collinearity equations and direct linear transformation equations.
- **drawContourLines( ):** Function draws the contour lines from the estimated elevations of points in image by estimateElevation method.

4.3.3.3 Mosaic



- **image:imageData :** It is the instance of the ImageData class.
- **registerImage( ):** Function to find the transform that best maps an image on the other.
- **localAdjust( ):** Function that makes local adjustment to avoid the ghosting and blur in the mosaic.
- **blendImage( ):** Function takes a pixel from a background image and a pixel from the source image and combine them with a specified color assigned to this overlapping pixel.
- **compensateExposure( ):** Function that clears the brightness difference between the overlapping images.

4.3.3.4 Orthophoto



- **image:imageData :** It is the instance of the ImageData class.
- **rectifyImage( ):** Function does rectification on image by using mathematics with world file information.
- **georeferencing( ):** Function does georeferencing on image by using mathematics with world file information.

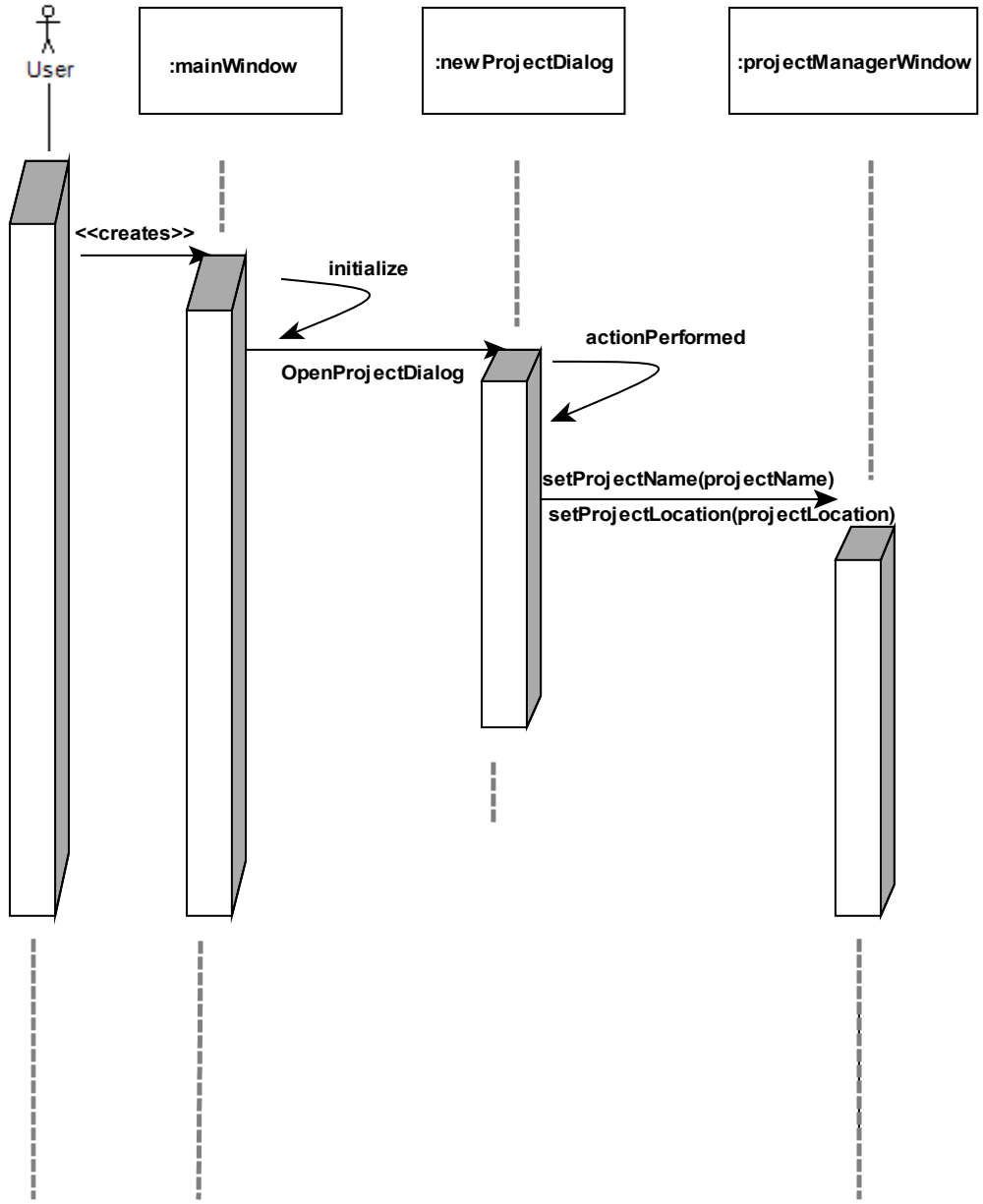
- **completeBlackPoints( ):** Function completes the black points that the orthophoto process causes.

4.3.3.5 Superresolution

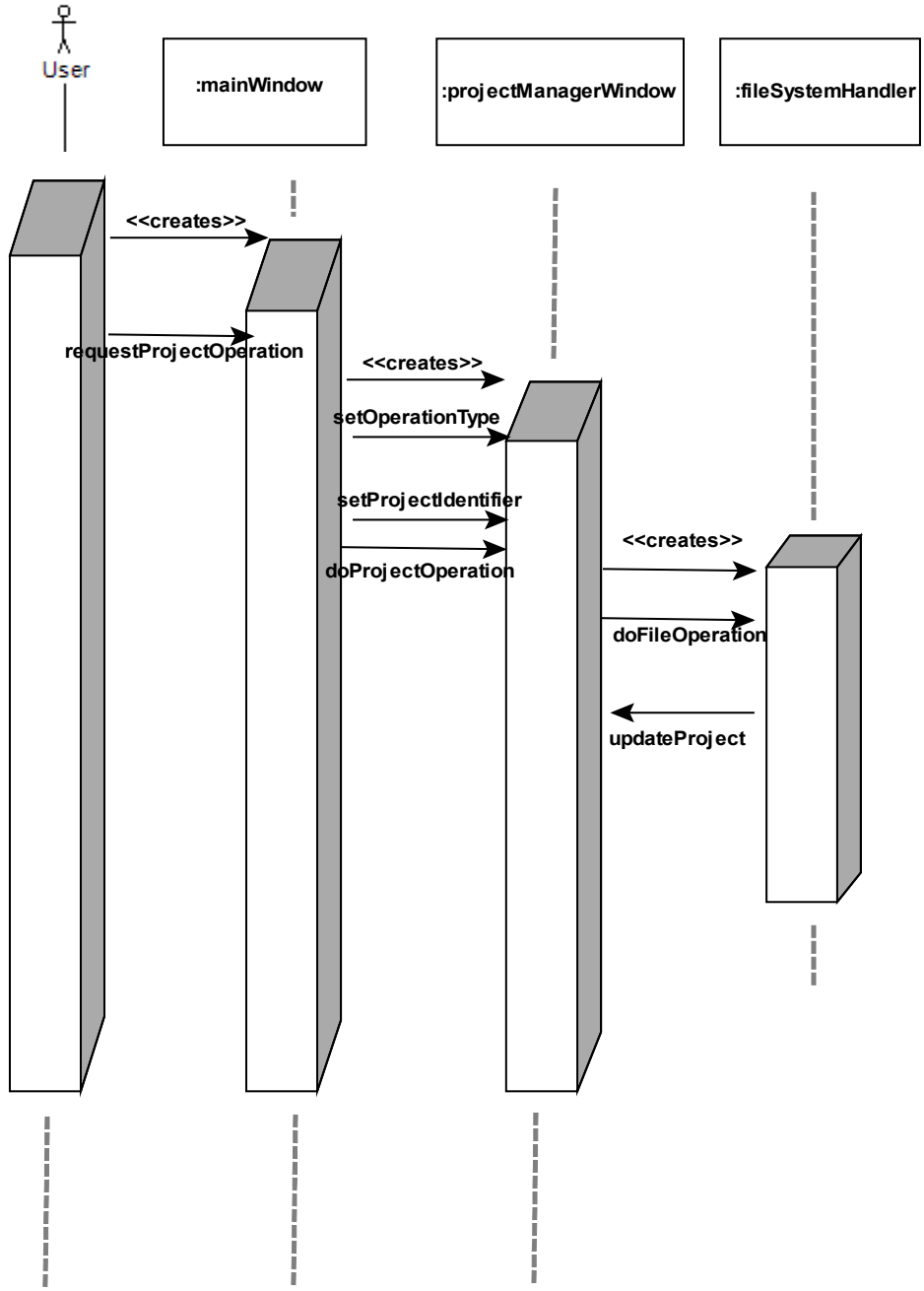
- **image:imageData :** It is the instance of the ImageData class.
- **detectTarget( ):** Function chooses a target image that is going to be used as a reference frame to warp all images.
- **geometricRegister( ):** Function searches matching features among images, finds them and extracts from corresponding images.
- **reconstructingImage( ):** Function warps all images and super resolution image is produced.

4.4 Sequence Diagrams

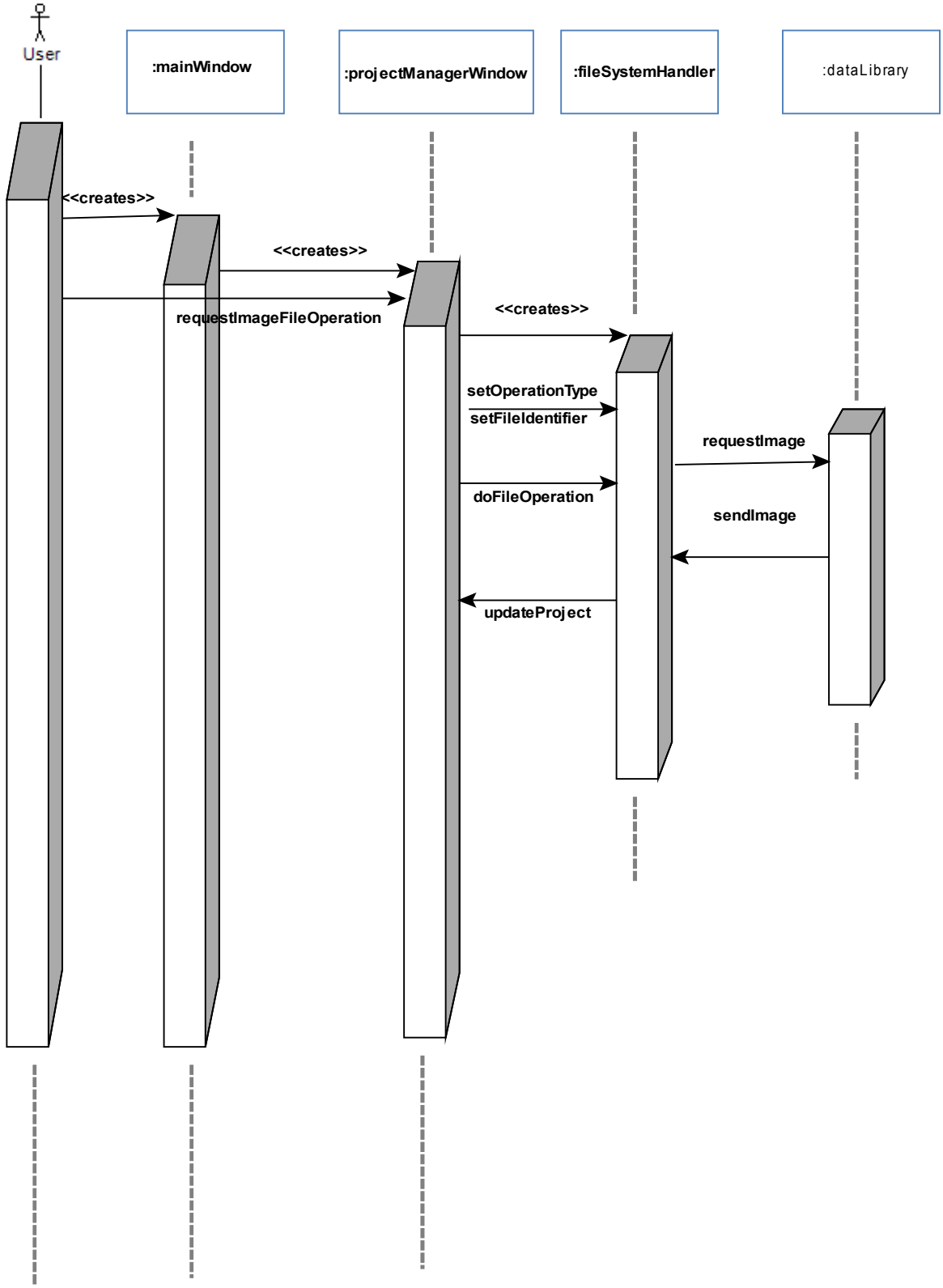
4.4.1 Create New Project



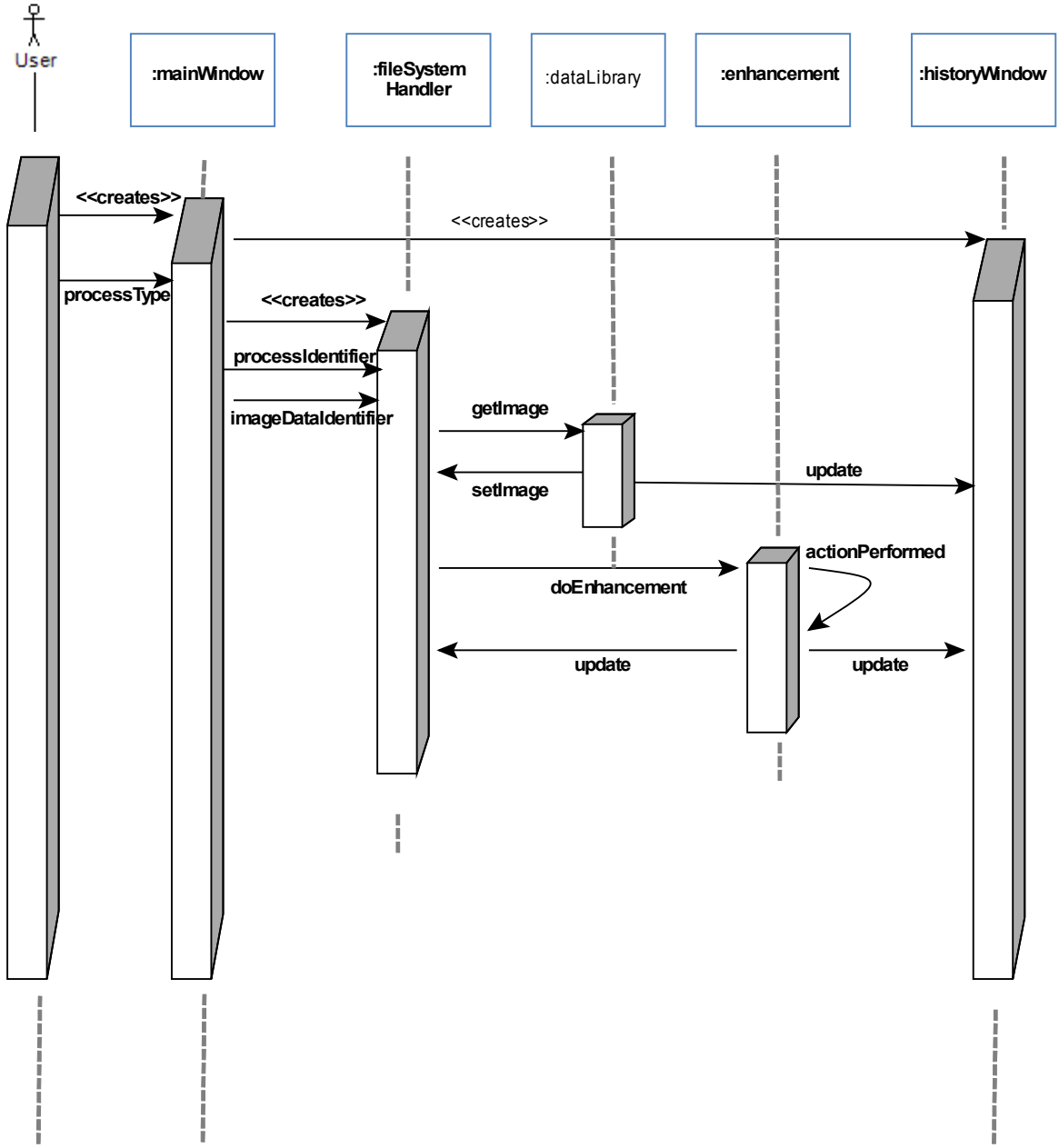
4.4.2 Project Operations



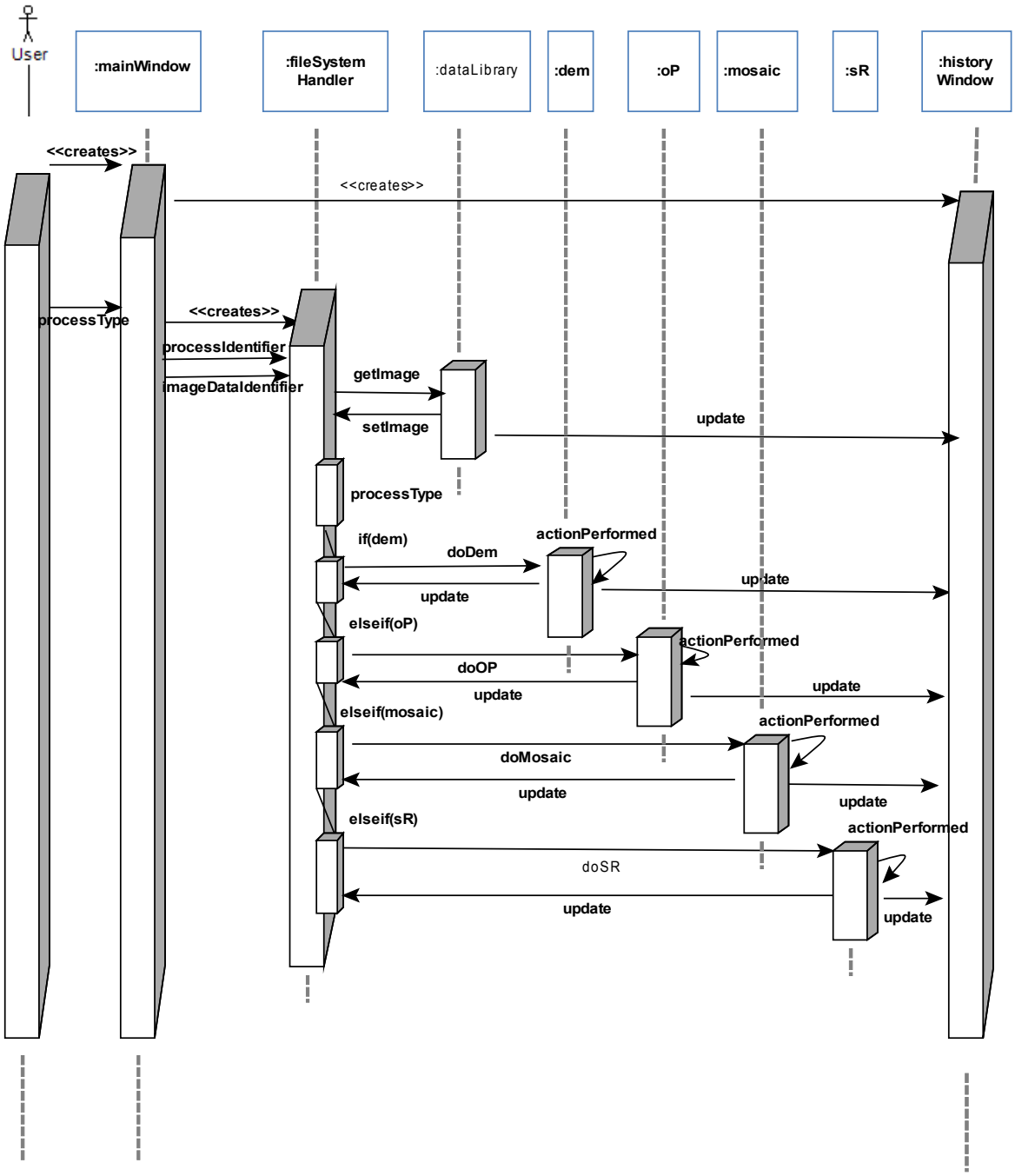
4.4.3 Image File Operations



4.4.4 Enhancement



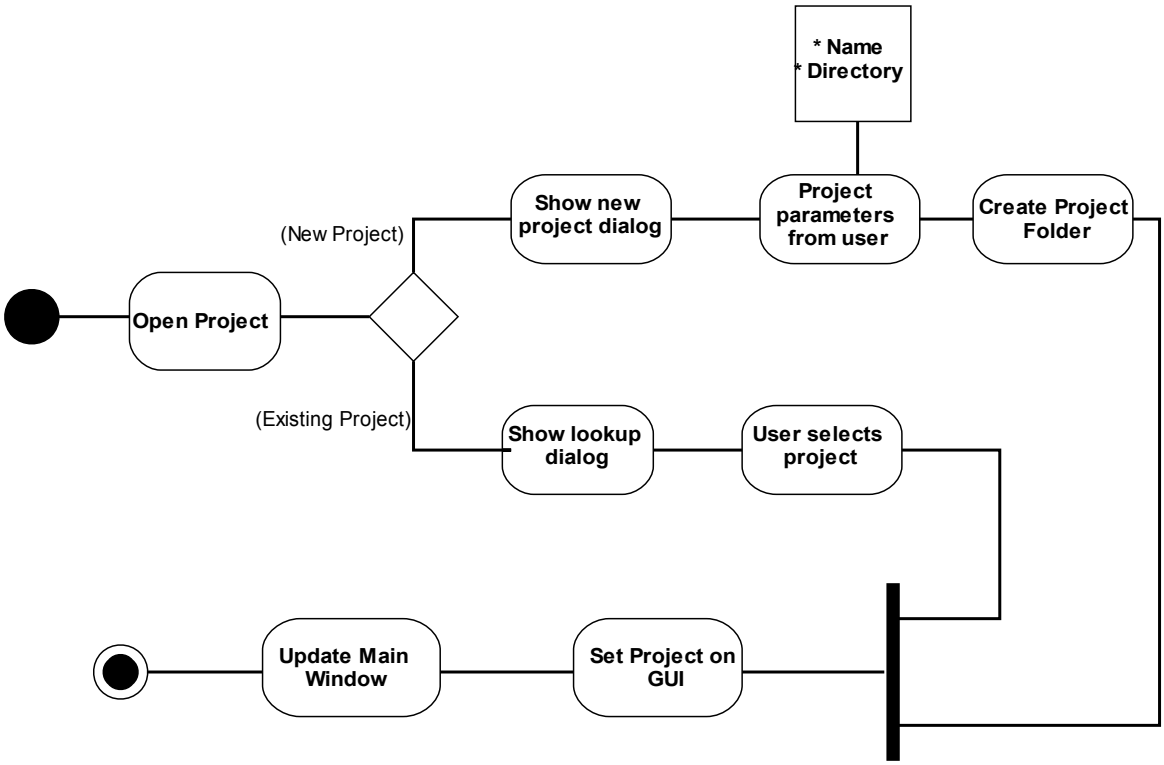
4.4.5 Photogrammetry



4.5 ACTIVITY DIAGRAMS

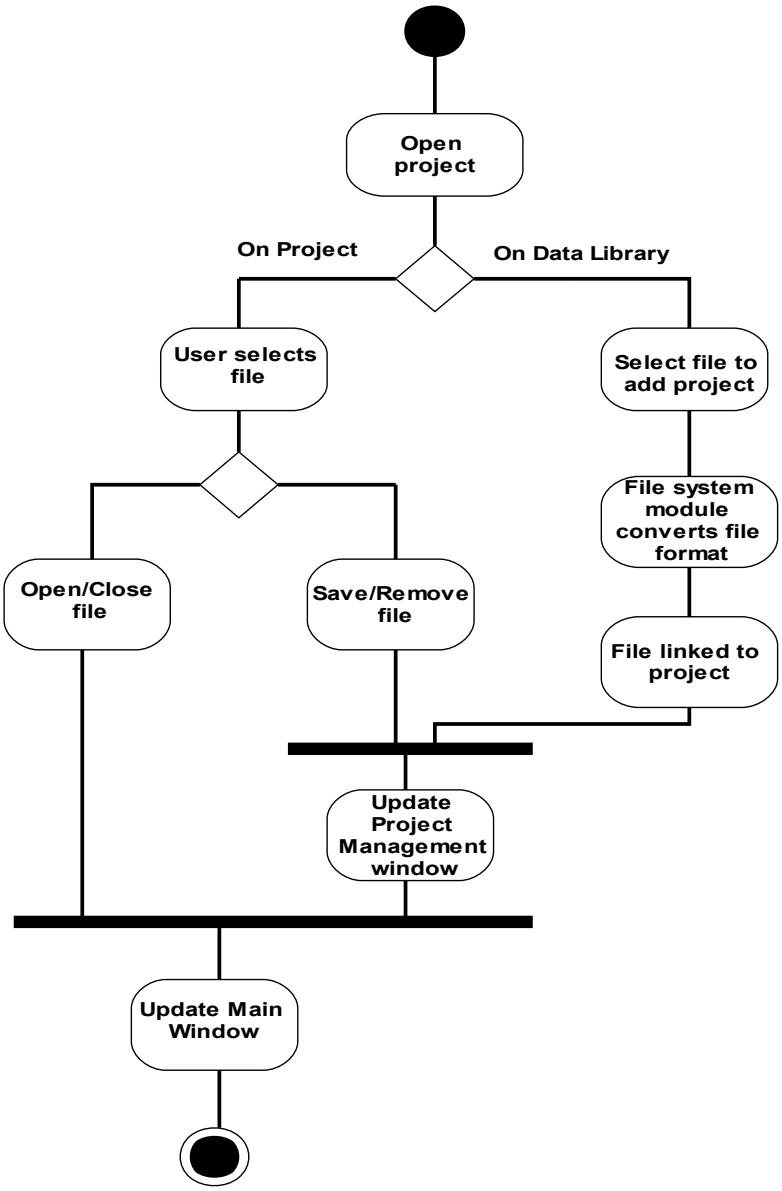
4.5.1 Open Project

When the user initiates the PHOTOLAB program, most likely the user wants to open a project. The user will either open an existing project or create a new project. If the user wants to create a new project, PHOTOLAB shows new project dialog . The user enters a project name and determines the file location to this dialog. After getting the project parameters from the user,a project folder is created by PHOTOLAB. The user may want to open an existing project. In this case, PHOTOLAB shows the user lookup dialog so the user can select the project that he/she wants to open. In both cases, PHOTOLAB opens the project on the main window.



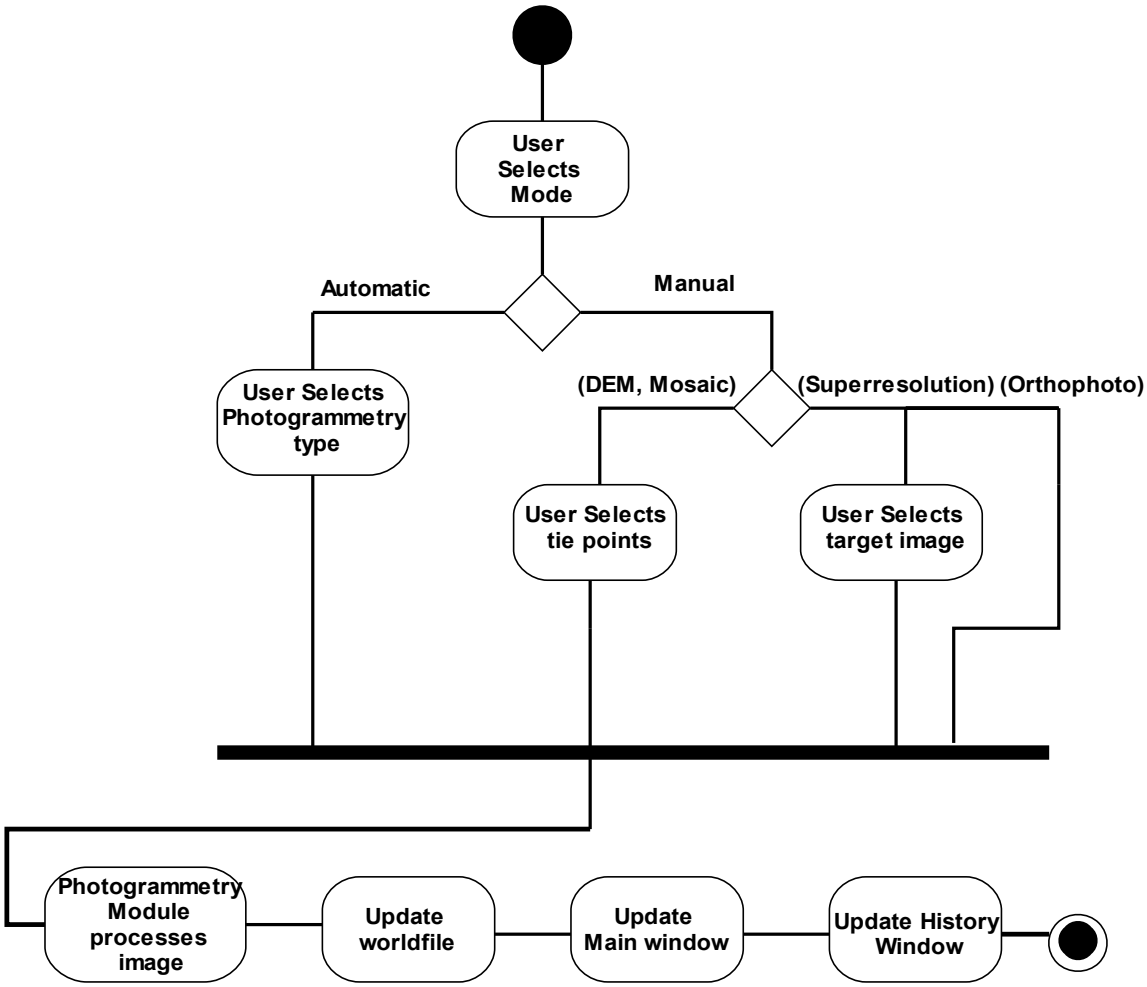
4.5.2 File Operations

File operations start with opening a Project. User may want two different spaces dependin on file operations. On Project, first user selects a file to operate. After selection, user decides on file operation type; open, close, save and remove. since the directory tree structure will be modified , save and remove operations needs updates on Project Management Window. On data library user may want select files to add Project. The file format is controlled by file system module and if it needs a conversion same module converts it into specified format. Then a link is created between file and Project. This operation also needs Project Management Window update. At the end, main window is updated and file operations finish.



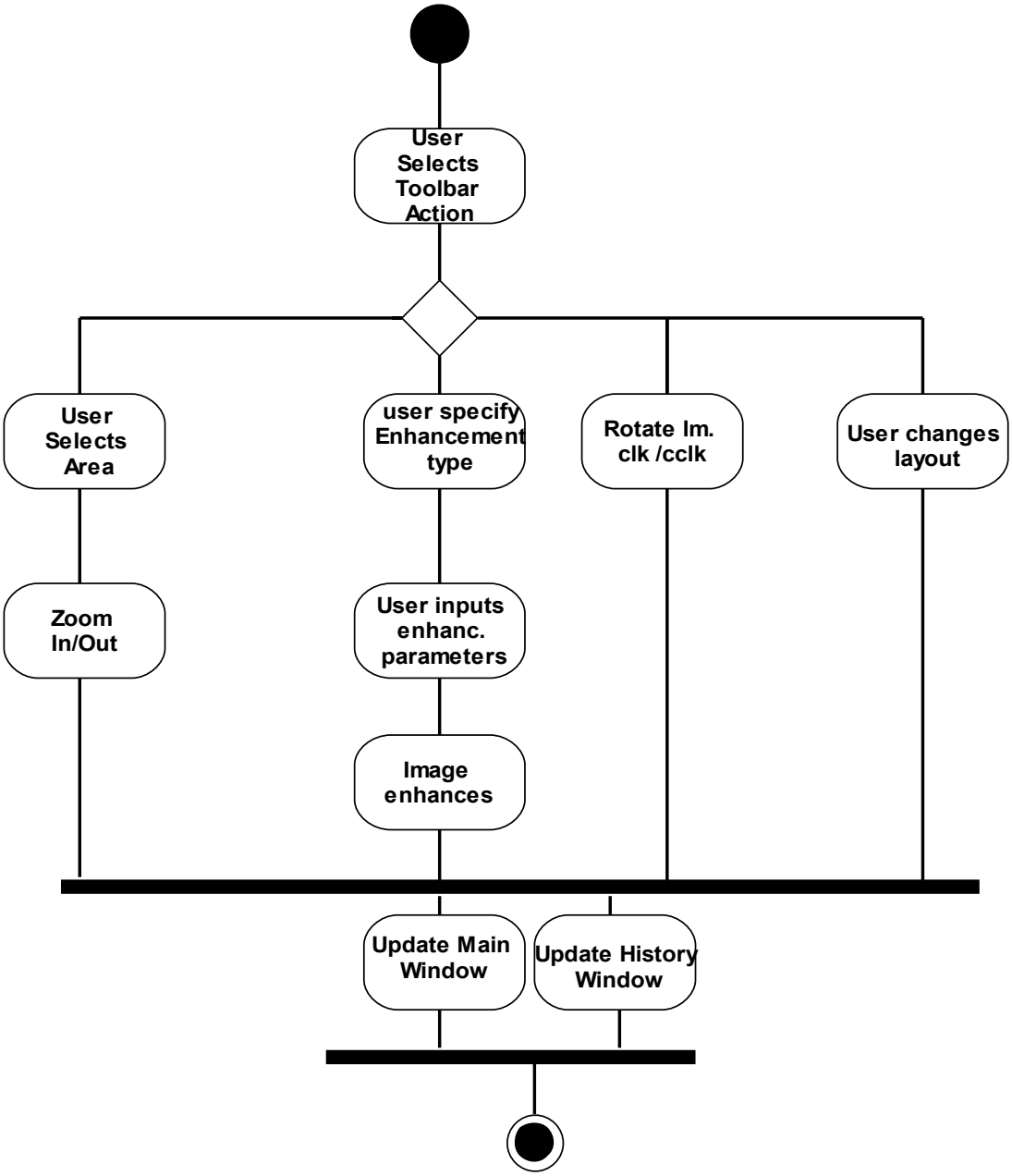
4.5.3 Photogrammetry Operations

The user selects the mode from the menu bar. PHOTOLAB gives two options to the user.User can either select the automatic mode or the manual mode. The default mode provided by PHOTOLAB is automatic mode. If the user selects manual mode, he/she either selects the tie points or the target image according to the photogrammetry process type to be processed. The user selects tie point if the process is DEM or Mosaic and selects target image if the process is SuperResolution and nothing is done if the process is OrthoPhoto. In the automatic mode PHOTOLAB wants user only to choose process type. Then PHOTOLAB processes on the images according to the selected process type.After the photogrammetry process, PHOTOLAB updates the world file,main window and history window.



4.5.4Toolbar Actions

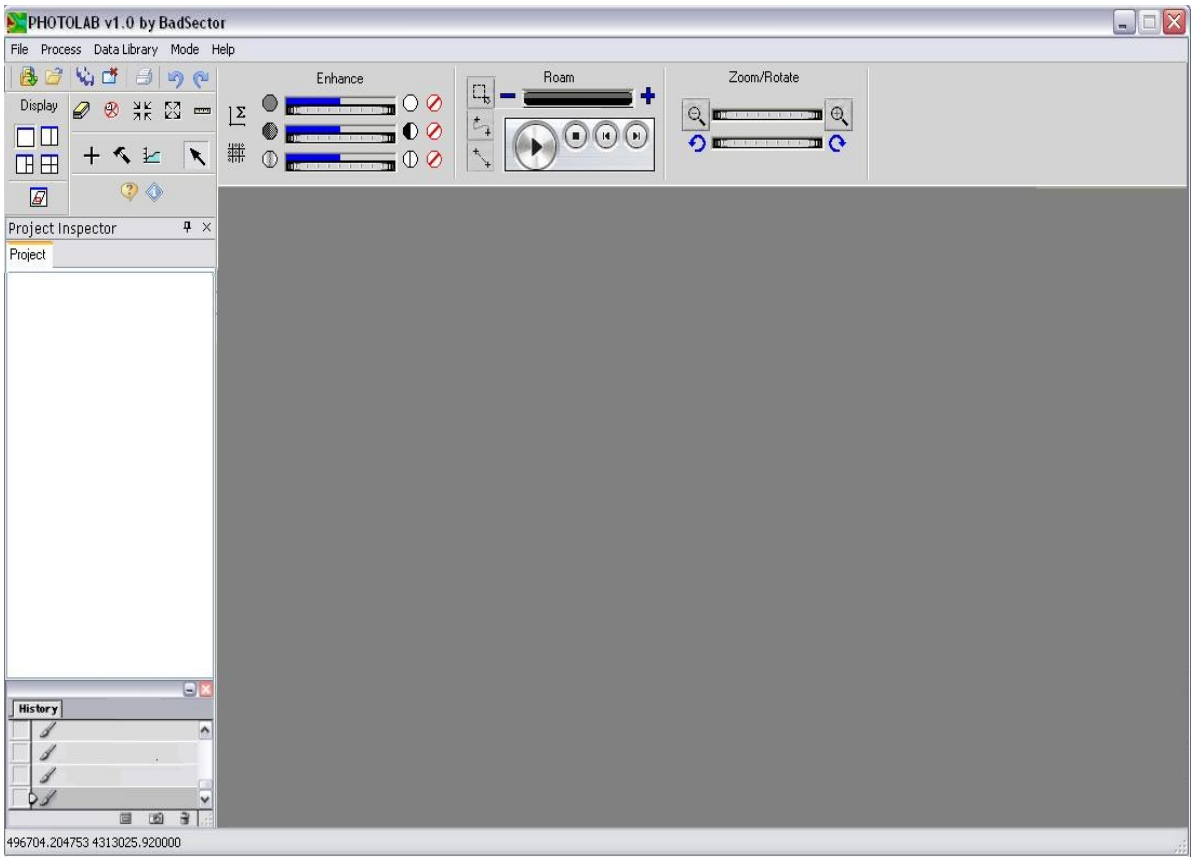
Toolbar actions provide making some operations on images. This toolbar is very crucial part of our IDE. It provides image enhancing, zooming, rotating options. Also, the layout can be changed from toolbar. The user first selects the operation to be done. After operating the process chosen by toolbar, the main window and the history window is updated.



5 GUI – GRAPHICAL USER INTERFACE

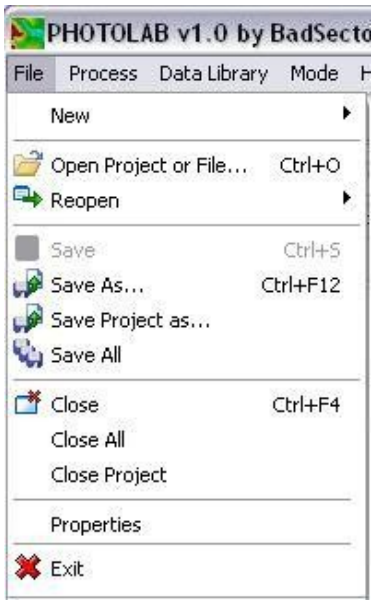
As a group we tried to design our GUI user-friendly to make the interaction with user easy and practical. In our design, all of our modules interact among each other via GUI, so this makes GUI design module, the most important part of our project.

The general graphical user interface of PHOTOLAB looks like as shown below.

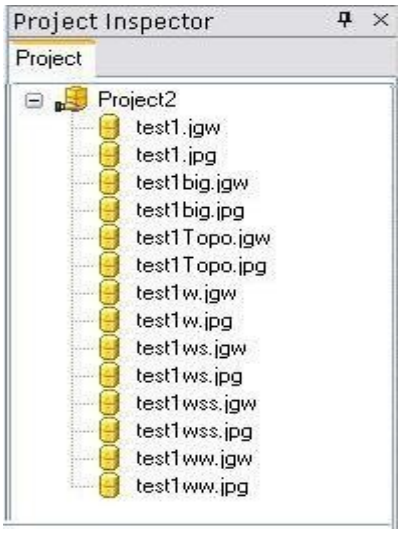


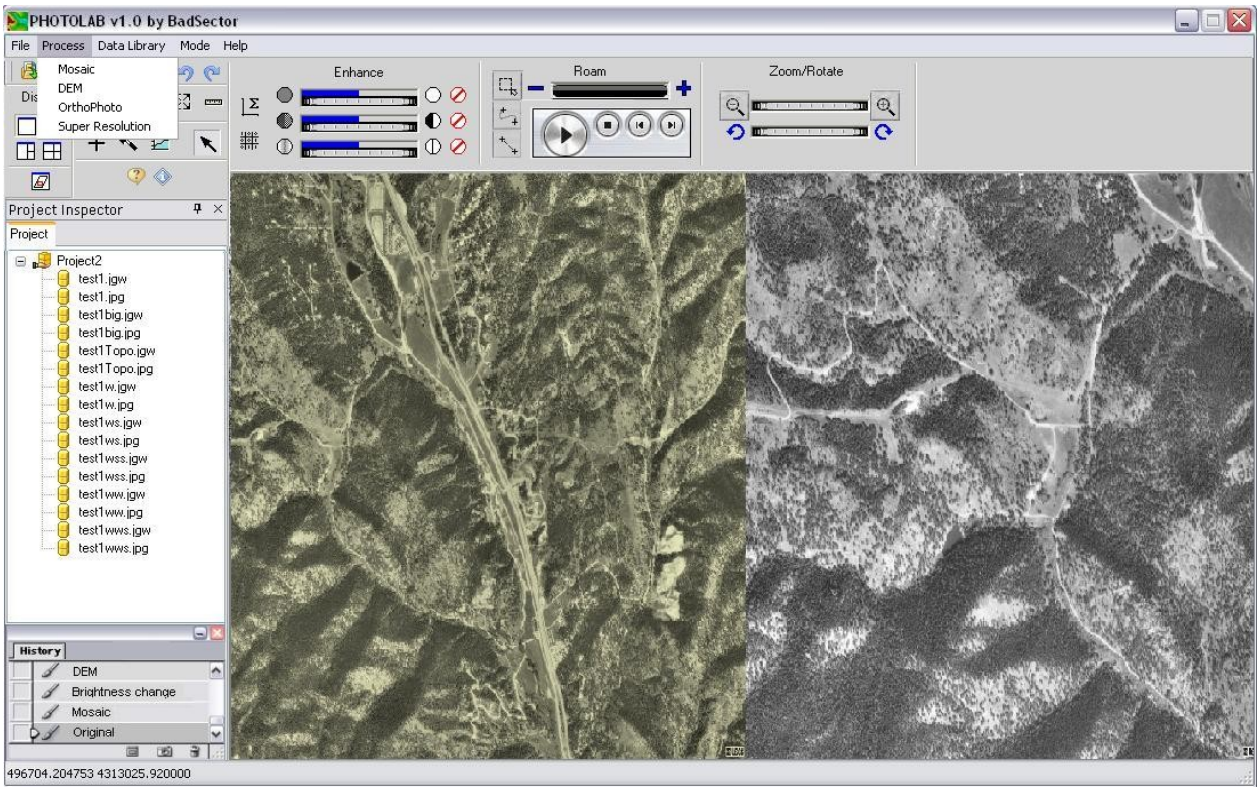
Upon starting, project inspector window is shown at left middle side of the main window. Project inspector window lists the files of the current opened project and it makes file operations easier. Just below it there is the history window. In history window there is a list of last three processes and the original version for the selected image. Besides undo and redo operations, user can turn back to the images before photogrammetry operations. In addition to project inspector and history window there is the main area for image display. At first the display area is not divided to sub display areas.

At the top of the main window, there is a menu bar which contains ‘File’, ‘Process’, ‘Data Library’, ‘Mode’ and ‘Help’ menu items. To start working with PHOTOLAB, you have to open a file or project by using ‘File’ menu. The user can either open a single file or, open or create a project. You save the project and files from this menu item and you also do the closing job from here.



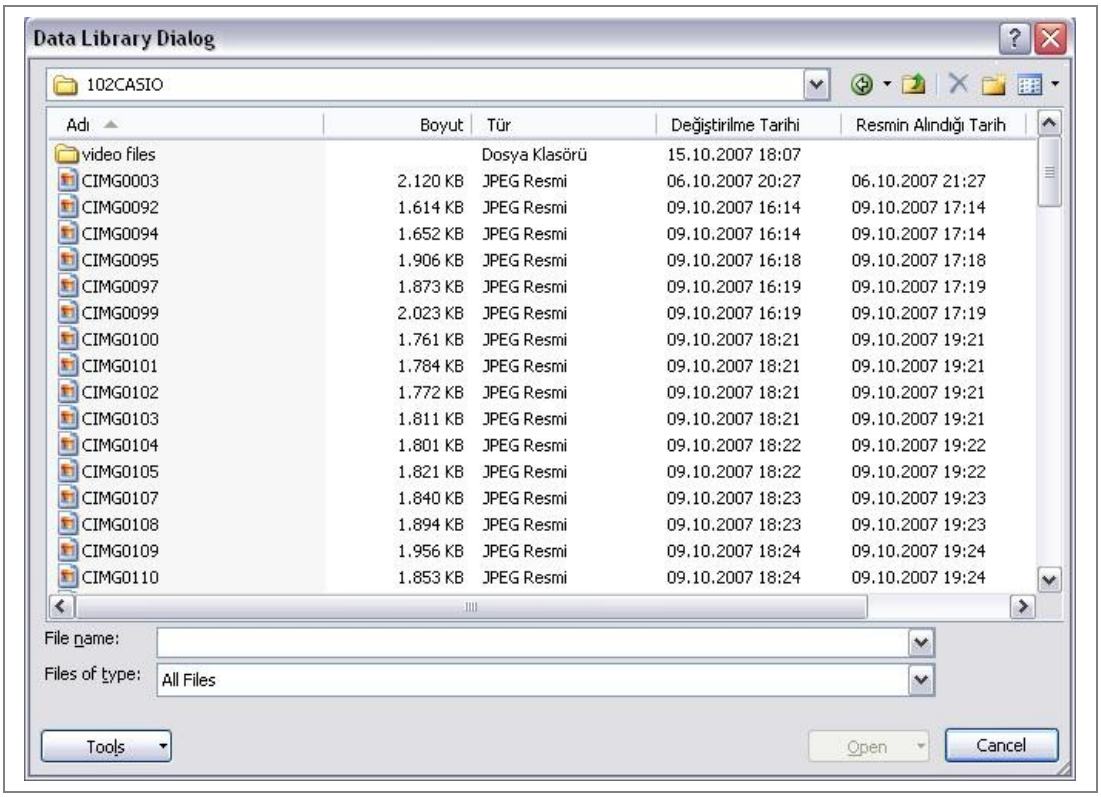
After forming a project or opening an existing one, the components of the project can be seen in project inspector window.





The second menu item is 'Process'. The sub menu items are mosaic, DEM, Orthophoto and Super Resolution. After opening the files that are going to be processed with one of these methods, the user simply click on which process he/she wants the PHOTOLAB to perform. The result of an example mosaic process is below.

On the right of Process menu item, there is Data Library menu item. The only submenu of this menu item is ‘Open Data Library’ submenu. When the user clicks the open data library submenu item, a dialog is opened for user to see the files in data library and operate on them. The data library includes image files, video files and world files of these. The snapshot of the data library dialog is below.



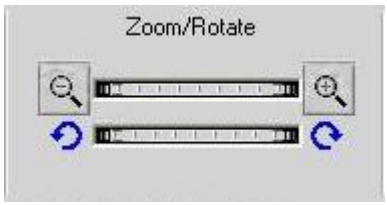
The next menu item after data library menu item is ‘Mode’ menu item. Mode is used for determining the tie point selection method. The user has two options, which are automatic and manual. In automatic mode, the tie points are determined by the program automatically and pointed in images, however in manual mode the user selects the tie points on images.

The last menu item is ‘Help’ menu item. PHOTOLAB provides user help files in order to make easier usage. These help files include the necessary information about the usage of the software in terms of functionalities. Help files also include a search and index box for faster search.

Above is a part of PHOTOLAB user interface toolbar. The display menu on the left provides the user to view the image(s) in five different scenes which are one-two-three-four images on the screen simultaneously or multiple images cascaded. This property of PHOTOLAB GUI is designed with MDI (Multiple Document Interface). Beside this, PHOTOLAB provides a number of other functionalities.

PHOTOLAB gives the user an opportunity to enhance the images by using various enhancement techniques such as setting the brightness values of images, contrast stretching and sharpening.

By the roaming function of PHOTOLAB, the user can roam on a predefined pattern. The user can define a path either as a linear line or as a non-linear path. User can also calculate the distance between the two end points of this predefined pattern.



By the zoom/rotate function of the PHOTOLAB, the user can zoom in and out on the image and he/she can also rotate them.

At the bottom of the main window there is a status bar. The world file of the images give information about the world coordinates of each pixel in the image. PHOTOLAB uses this world file information to calculate the original coordinates of the image. While the cursor is wandering on the image, the world coordinates of the place shown by cursor is written in status bar as UTM coordinates.

At the lower left side of the main window there is history window. The aim of the history window is to provide the user more possibilities to see the whole process step by step. PHOTOLAB records last four changes on the project and the original of the project easily. For this purpose history window shows these recorded changes for selected image.

6 SYNTAX SPECIFICATION

6.1 Classes

Class names start with a capital letter. If it has more than one word, first letter of each word is capital, too. In “x.h” files classes are divided into three parts and their order is private , protected and public members. Inside each part , members are ordered as;

1. Data structures

2. Variables

3. Functions

And there is one empty line to seperate these three. Besides, data structures, variables and functions are grouped according to their usage areas.

Opening curly brace of class is adjacent to closing parenthesis. Closing curly brace is in independent empty line.

A sample class structure is;

```
class Sample{

private:

	vector<Example> sampleVector;



	int Variable1;

	int Variable2;



	void DoSomething();

	void DoAnotherThing();

protected:

public:

}
```

6.2 Functions:

Function names in local areas start with lower case. If there is more than one word other words start with capital letter. Functions are implemented in x.cpp files. All implementation has a tab from the page edge Local variables are grouped at the top of the function implementation. Language concerned implementations like “if”, “while” and “for” have one tab between the nested belongings and curly braces. Opening and closing parenthesis have one space between parameters and each other. A sample function is;

```
void sampleFunction( int parameter )
```

```
{
        int variable;
        int variable2;


        while( isHappening )
        {
                doSomething( );
        }
}
```

6.3 Variables

Gloabal variables started with capital letter. However local variables have lower case first letter. If variable has more than one words other words start with capital letter, too.

6.4 Comments

There will be a comment line above each function which defines the aim of the function. The comment sentences will be formal and clear. Each member of the class ".h" files has definitions as comment. In ".cpp" files , before implementations there will be creation date of file, name of creator, modification date, name of modificator and class definition inside multiple line comment. A sample comment is;

```
/*  # Created 01.12.2007 by Serra Sinem Tekiroğlu

    # Modified 02.12.2007 by Serap Atılgan

   # This sample class comment is written for exemplification of comment specifications. */
```

7 PROCESS MODEL and PROJECT SCHEDULE

7.1 Team Structure

As a group we decided that “Democratic Decentralized (DD)” fits best to our project and project group. Our first priority is cooperation and this model ensures a high rate of cooperation since it forces us to communicate in decision making. We appointed the tasks to the members for short durations at our weekly meetings. Another reason that we have chosen this structure is; we make decisions on problems or weekly tasks by agreement of each member. In case there is a contradiction among the group members, we chose a group leader to say the last word to prevent disagreements.

7.2 Process Model

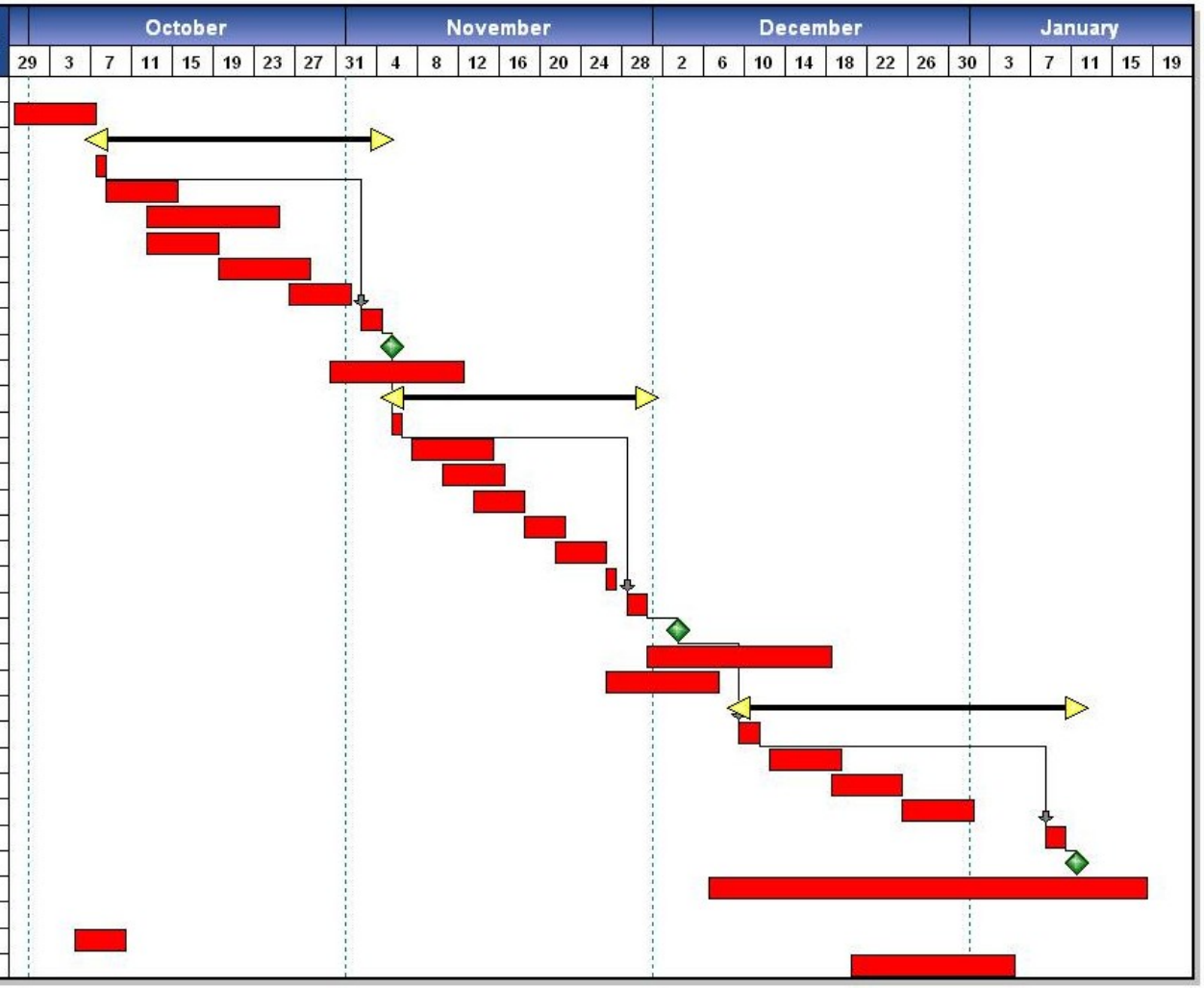
Our project team will iteratively go through planning, modeling, construction and deployment stages. For this reason, linear (waterfall) model of development best fits to our project. After the requirements analysis, we are going to make our initial design. Actually, Milsoft wanted the process model to be spiral during the design and implementation phases so they wanted more than one prototype. By this way, we will have a chance to go back and correct the faults in the design which we found during implementation.

7.3 Gantt Chart

Time Schedule of Photolab Project is shown by Gantt Chart. We have planned for a whole academic year.

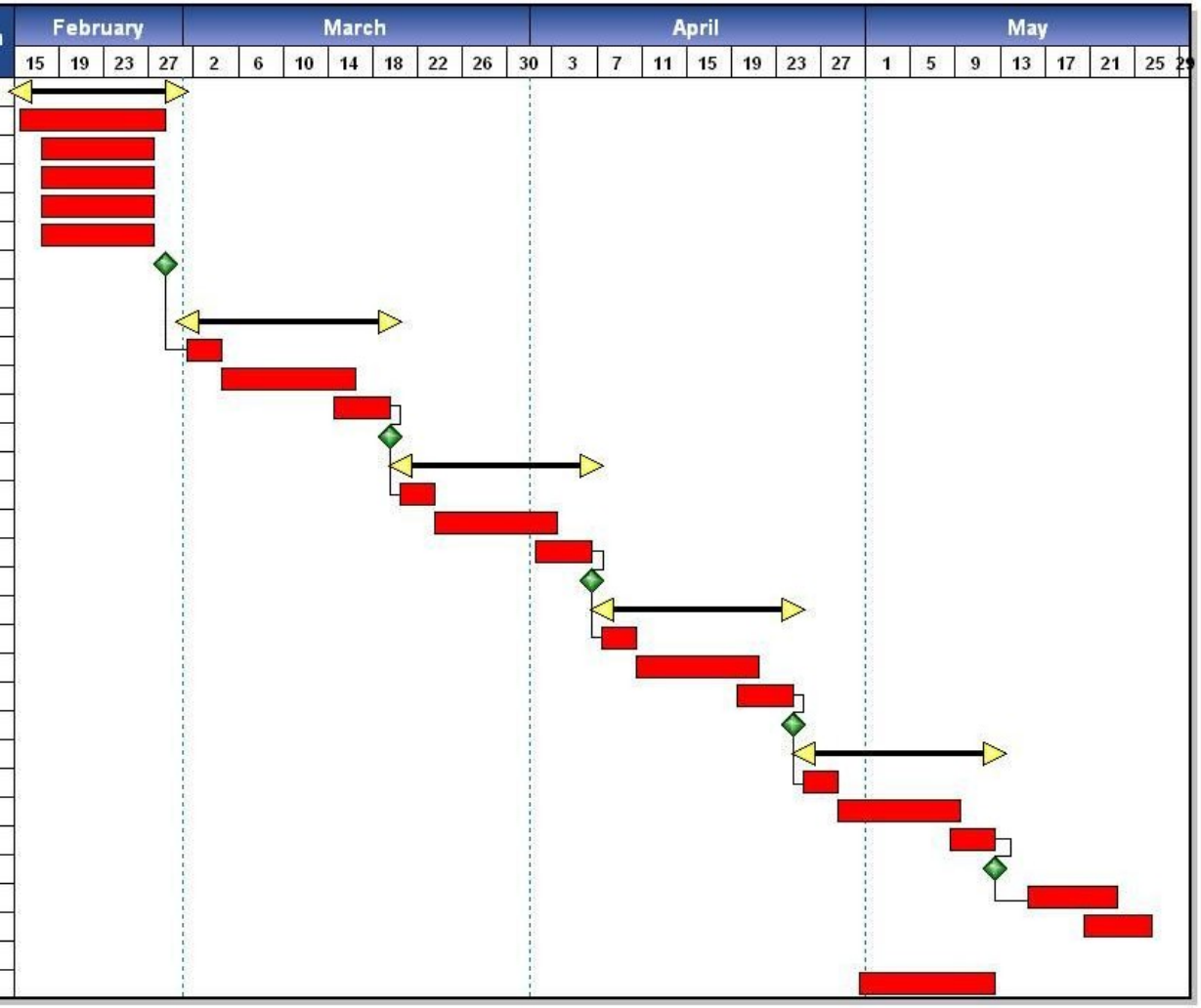
Gannt Chart

ID	Group Memb	Task	Duration
1			
2	1,2,3,4	*Teaming Up&Project Proposal*	7 days
3		Analysis Report	28 days
4	1,2	*Outline of Report*	1 day
5	1,2,3,4	*Market Research*	7 days
6	3,4	*Determining User Requirements*	13 days
7	1,2	*Determining System*	7 days
8	1,2	*Data&Behavioural Model*	9 days
9	3,4	*Functional Model*	7 days
10	1,2	*Last Version of Report*	2 days
11		MILESTONE 1	
12		Milsoft Assignment	2 weeks
13		Initial Design Report	25 days
14	3,4	*Outline of Report*	1 day
15	1,2,3,4	*Initial GUI Design*	8 days
16	3,4	*Defining Constraints*	6 days
17	1,2	*Design of Core Module*	5 days
18	1,2	*Design of Supporting Modules*	4 days
19	1,2,3,4	*Modelling*	5 days
20	3,4	*Update Schedule*	1 day
21	3,4	*Last Version of Report*	2 days
22		MILESTONE 2	
23		Milsoft Assignment	2 weeks
24	1,2,3,4	Initial Prototype Preparation	12 days
25		Final Design Report	33 days
26	1,2	*Outline of Report*	2 days
27	1,2,3,4	*Review of Modules*	7 days
28	1,2,3,4	*Final GUI Design*	7 days
29	1,2,3,4	*Final Design Verification*	7 days
30	3,4	*Last Version of Report*	2 days
31		MILESTONE 3	
32	1,2,3,4	Prototype Preparation	4 weeks
33		Other	
34	1,2	*Web Site Design*	5 days
35	3,4	*Presentation*	2 weeks



Gannt Chart

ID	Group Member	Task	Duration
36		Implementation of Toolbar	
37	1,2,3,4	*Enhancement options*	10
38	1	*Roaming*	10 days
39	2	*Zoom/Rotate*	10 days
40	3	*Display Layout*	10 days
41	4	*Basic Functions*	10 days
42		MILESTONE 4	
43		Implementation of Submodules	
44		Implementation of Mosaic	18 days
45	1,2	*Algorithm Search*	3 days
46	1,2,3,4	*Implementation*	12 days
47	3,4	*Testing*	5 days
48		MILESTONE 5	
49		Implementation of DEM	18 days
50	3,4	*Algorithm Search*	3 day
51	1,2,3,4	*Implementation*	12 days
52	1,2	*Testing*	5 days
53		MILESTONE 6	
54		Implementation of OrthoPhoto	18 days
55	1,2	*Algorithm Search*	3 days
56	1,2,3,4	*Implementation*	12 days
57	3,4	*Testing*	5 days
58		MILESTONE 7	
59		Implementation of SuperResolution	18 days
60	3,4	*Algorithm Search*	3 days
61	1,2,3,4	*Implementation*	12 days
62	1,2	*Testing*	5 days
63		MILESTONE 8	
64	1,2	Whole System Test	8 days
65	3,4	Optimization	7 days
66		Other	
67	1,2	*Presentation*	2 weeks



1-Hanife, 2-Serap, 3-Serra, 4-Meryem

8 TESTING

Testing is to detect the differences between existing and required conditions and to evaluate the overall software by analyzing it with respect to predefined validity and correctness norms. As Bad Sector team, we believe that the testing is a crucial part of a software design procedure since without testing; one can not be sure the overall validity and correctness of the final software package. Upon decisions on testing, Bad Sector team has prepared an initial test plan to follow. In subsequent sections the testing procedure will be covered according to this plan.

8.1 Test Items

Testing is a procedure which must be performed at several predefined points in the life cycle of the software development. Since testing is a very dependent and continuing activity, test plan must be developed according to these predefined levels. In order to specify these points, firstly test items must be identified. Bad Sector team has identified the items as below:

- Software Modules
    - GUI
    - File System
    - Photogrammetry
- Job Control Procedures
    - Production Scheduling and Control(refers to the prepared project schedule (Gantt Chart) and controlling time, cost and efficiency issues.)
    - Calls and Job Sequencing (refers to the job calls and the their sequence in these calls.
    - Job Control Language(“is a scripting language to instruct the system on how to run a batch job or start a Subsystem”[wikipedia].)
- Operator Procedures

  Operator procedures are the items to be tested in order to ensure that the application can run on different machines and environments.

- User Procedures

  User procedures are the items to be tested in order to ensure that user documentation is correct, complete and comprehensive.

8.2 Test Approach

As Bad Sector team we think that the software testing consumes 20 percent to 30 percent of software development resources. Therefore Bad Sector team has decided to specify an approach for all major testing tasks and for the required time estimation to do these tasks, with the minimum time & cost and maximum efficiency & satisfaction.

The team constructed the approaches by identifying the types of testing with the methods and criteria used in the testing.

8.2.1 Component Testing

Component testing is to test particular functions or code modules. It is usually the most micro scale of testing. Bad sector team plans to test the modules of Photolab with the component testing approach.

- GUI

Component Test Check List

Question to be answered:	Passed	Failed
Are the passes between subunits of GUI successful?	Yes, skip to the next question…	No, work on the unit further.
Do the shortcuts work correctly?	Yes, skip to the next question…	No, work on the unit further.
Are the menu items in correct order?	Yes, skip to the next question…	No, correct the order.
Are the appropriate menu choices active?	Yes, skip to the next question…	No, recover the activation.
Are the data interactions between subunits of GUI	Yes, skip to the next unit.	No, focus on the data interactions.

successful?		

- File System

Component Test Check List

Question to be answered:	Passed	Failed
Are all the file operations coherent with the File System?	Yes, skip to the next question…	No, work on the coherence.
Can all the image file types be converted to the common data type correctly?	Yes, skip to the next question…	No, work on the image type conversion.
Can File System and GUI interact successfully?	Yes, skip to the next question…	No, focus on the interaction.
Can File System and Data Library interact successfully?	Yes, skip to the next question…	No, focus on interaction.
Can video files be converted to image files correctly?	Yes, skip to the next question…	No, focus on conversion.
Can images with a relatively large size be handled?	Yes, skip to the next unit…	No, work on handling the large sized images further.

- Photogrammetry

Component Test Check List

Question to be answered:	Passed	Failed
Are the interactions between subunits of Photogrammetry successful?	Yes, skip to the next question…	No, work on the interactions.
Are all the algorithms efficient and fast?	Yes, skip to the next question…	No, work on the algorithms.
Can subunits work successfully?	Yes, skip the next question	No, correct the subunit.
Are the subunits coherent with the Photogrammetry module?	Yes, skip to next question…	No, work on coherence.

8.2.2 Integration Testing

As Bad Sector team, we decided to make an integration test to check the system after combining the parts (i.e. modules or individual applications such as image enhancement or photogrammetry processes) to ensure that they function together correctly. Bad Sector team can not be satisfied with a component, only working correctly on its own area. Therefore, the team will also ensure that the integration is successful while doing a continuous component testing in background.

8.2.3 Interface Testing

Bad Sector team plans to make an interface testing after completing component and integration testing successfully and solving the all critical errors. The aim of the team in doing interface testing is to check the external interfaces with Photolab in order to verify the execution times, data exchange, transmission and control. In order to make the test Bad Sector team needs to find external organizations having interfaces which can be tested with Photolab.

8.2.4 Performance Testing

As Bad Sector team, we also plan making a performance test in order to see how fast a component outputs under a particular workload or what percent of quality (time, cost, efficiency) and validity issues such as reliability or resource usage are satisfied by the system.

8.3 Pass/Fail Criteria

Bad Sector team decided pass/fail criteria for the test cases. The decisions include the suspension of a test in case an occurrence of a more urgent one. After the completion of the urgent test case, the suspended test will be resumed from where it already is. Decisions also include the resumption of a test case until it succeeds as well as approval of a specific test case if it satisfies the criterion for all of the components forming the case.

9 CONCLUSION

During the preparation of Requirement Analysis Report, Bad Sector team members were unaware of how they would embody the design of their software. The whole system was made up of an abstract design. When the Initial Design phase came, the team faced with the reality to move the design of the project from abstract to concrete. Upon this reality, the team decided to make more search on technical issues about the project in order to make a good initial design. As Bad Sector team, we know that the initial design of the project means forming a basis for the implementation phase of the project. Therefore it must be clear, comprehensive, and objective so that it will be a perfect guide to the team through the way of implementation. In the light of these beliefs, the team tried to be specific during the formation of the modules, classes and the hierarchy of the whole system. The team also tried to be objective during the preparation of class, activity and sequence diagrams so that any other developer, inspecting the diagrams, could have understand the design and code it as well as the team.

To conclude, as Bad Sector team, we are aware of how important this initial design document for the upcoming phases of the project. Therefore, the team tried to do its best during the preparation of this Initial Design document. Our team will use this document as a guide for the project presentation at the end of this semester. The document will be upgraded after getting feedback from our project advisor and Milsoft on this document and on upcoming releases during the next phases of the project.

REFERENCES

1. UML Sequence Diagram Tutorial, Effexis Software,LLC,[2005-2007]

   URL: http://www.sequencediagrameditor.com/uml/sequence-diagram.htm

2. IEEE Standard 829-1998, Standard for Software Test Documentation

   URL: http://www.ecs.csun.edu/~rlingard/COMP480/IEEETestPlanTemplate.pdf

3. High Resolution Panaromas using Image Mosaicing,Stanford University EE368 Final Project

   Laurent Meunier and Moritz Borgmann,May 2000

   URL: http://scien.stanford.edu/class/ee368/projects2000/project13/index.html

4. Norvelle, F.R., 1994. Using Iterative Orthophoto Refinements to Generate

   and Correct Digital Elevation Models, Proc. Mapping and

   Remote Sensing Tools for the 21st Century, ASPRS, pp. 134-142.

5. Krzystek, P., 1995. Generation of Digital Elevation Models, Second

   Course in Digital Photogrammetry, Bonn, 6-10 February, Ch. 7.

6. [Davis] Davis, J. Mosaics of scenes with moving objects.(1998).Computer Vision

   and Pattern Recognition

7. Feature Based Image Mosaicing,Satya Prakash Mallick,Department of Electrical and Computer

   Engineering, University of California, San Diego

   URL: http://www-cse.ucsd.edu/classes/fa02/cse252c/smallick.pdf

8. Prof.A.Gruen&Henri Eisenbeiss(2006) UAV Photogrammetry IGP-ETH Zurich, 25/11/2007

   URL: http://www.photogrammetry.ethz.ch/research/heli/index.html