CEng 491 Initial Design Report

Photogrammetry Lab / Milsoft

Fall 2007

pix‘r’us

Ebru Aydin	1394691
Berk Demir	1448588
Ender Erel	1395029
M.Ozan Kabak	1389568

Assistant/Supervisor

Murat Yükselen

1. INTRODUCTION

1.1. Purpose

Purpose of this report is to give detailed information about the design of the Photogrammetry Lab project. Main subjects will be;

- Detailed problem description
- Architectural design
- Data design
- Interface design
- Detailed information about modules, subroutines, functions which are planned to be used
- Hardware, software and environment specifications of the project.

1.2. Design Constraints and Limitations

1.2.1. Time

Pix'r'us Photogrammetry Suit (PPS) is planned to be completed until the end of May – 2008. Phase deadlines will be decided in a rolling structure.

1.2.2. Performance

Since PPS includes time consuming matrix operations and processing over big images, optimization of each function should be satisfied for better performance.

1.2.3. Reliability

PPS will use wxWidgets as an interface library (as a requirement of Milsoft), which dictates both implementation and the functionality of the software. Also libraries used in PPS reduce the time to develop phases; their disadvantages are considered to be limitations.

1.3. Design Considerations

1.3.1. Extensibility

PPS will be designed to be flexible for new capabilities without major changes to the underlying architecture. Super resolution and video mosaic functions can be shown as examples.

1.3.2. Robustness

PPS will be able to operate under stress or tolerate unpredictable or invalid input.

1.3.3. Reliability

The software is able to perform a required function under stated conditions for a specified period of time.

1.3.4. Fault-tolerance

PPS is resistant to and able to recover from component failure.

1.3.5. Modularity

PPS comprises well defined, independent components. That leads to better maintainability. The components could be then implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.

1.3.6. Reuse

The modular components designed should capture the essence of the functionality expected out of them and no more or less. This single-minded purpose render the components reusable wherever there are similar needs in other designs.

2. PROBLEM DEFINITION

2.1. Problem Statement

The mission of pix'r'us is to design Pix'r'us Photogrammetry Suite (PPS) which will host integrated software tools for geospatial imaging developed by pix'r'us. Software is designed to be in co-operation with Unmanned Air Vehicles as input providers commonly using video streams or images. Since target users of PPS are serving on military platforms, accuracy of outputs is the most important fact to focus on. Also simple, user-friendly interface and fast processing are other keys to the design of PPS. According to target user requirements there will be three important functions of PPS; creating Mosaics, creating DEMs, creating Orthophotos.

2.2. Project Goals, Objectives and Scope

Three main goals of PPS can be listed as;

- DEM Creation: Digital Elevation Model (DEM) is digital representation of ground surface topography or terrain.[1] DEMs are the first step to rectification of aerial photos. PPS will generate DEMs by using multiple images.

- Orthophoto Creation: An orthophoto or orthophotograph is an aerial photograph that has been geometrically corrected ("orthorectified") such that the scale of the photograph is uniform, meaning that the photo can be considered equivalent to a map.[2] Unlike an uncorrected aerial photograph, an orthophotograph can be used to measure true distances, because it is an accurate representation of the earth's surface, having been adjusted for topographic relief, lens distortion, and camera tilt.[3] PPS will generate orthorectified photos from input images. PPS will use DEMs (either by importing existing DEMs or generating DEMs from the input images) in this process to increase the accuracy of generated photographs.

- Mosaic Creation: Image mosaicking (or image mosaicing) is the process of combining a set of separate images into a single seamless image (a panorama).[4] PPS will combine input images to generate a unified image if requested.

Among these three main goals there may be additional functionalities such as;

- Vide Mosaic Creation: A video mosaic is a high-resolution image that is created by stitching together the low-resolution frames from a video sequence.[5]

- Super-resolution Image Creation: Super-resolution is a technique to use multiple frames of the same object to achieve a higher resolution image.[6]

3. DEVELOPMENT SCHEDULE

3.1. Current Stage

The foundations of the general software architecture are laid. Relevant class hierarchies and data design hierarchies are constructed. These design considerations are discussed in relevant sections in great detail. A yearlong development schedule has been prepared (see Gantt chart given in Appendix-A). According to the development schedule, the first task is to design a basic image viewer GUI that will be used as a testing & development platform during the subsequent development phases. This first task has already been completed. The image viewer GUI is basically an MDI window containing a number of resizable children windows, each of which is a pane that can be docked or made floating by the user. Children windows display relevant images and perform auto-scrolling if needed. In addition to the children windows, the main MDI window also has a menubar, a toolbar and a statusbar. The

menubar and the statusbar contain menus and buttons that provide various functionalities. The statusbar is divided into two parts, each of which displays two kinds of information:

* Global information: Information about the current project (i.e. project name, opened files etc.) and active child window number.

* Child window specific information: Relevant information about the active child window; such as world coordinates corresponding to the current mouse location, number of feature points found in the image etc.

In addition to its use as a testing & development platform during the development subsequent development phases, this image viewer GUI itself will be developed in time to serve as the GUI of the end product. The screenshot of the first prototype is shown below. In this screenshot, both docking panels and floating panels of the multiple document interface can be seen. The statusbar also shows information about the real world coordinates of the pixel under the mouse cursor. This GUI also has a simple menubar and a simple toolbar, which can be extended for more functionality in the future phases. The docked panels demonstrate the automatic scrollbar adjusting property of the GUI.

Figure 2- 1 : GUI of the prototype

3.2. Future Work

With the development of the basic viewer, the first iteration of the development process is completed. The next iteration mainly involves the design and implementation of the image registration module and its integration to the image viewer GUI. This phase is followed by the phases that comprise of:

* Detailed design and implementation of the image/video mosaic module, its black box testing and its integration to the GUI.

* Detailed design and implementation of the DEM module, its black box testing and its integration to the GUI.

* Detailed design and implementation of the orthophoto module, its black box testing and its integration to the GUI.

* Testing of the interoperability of the modules.

* Testing of the overall software.

These steps are illustrated in greater detail in the Gantt chart in appendix A.

4. ARCHITECTURAL DESIGN

During the architectural design of pix'r'us photogrammetry suite, the main consideration was modularity. The decision of organizing the three main capabilities as separate modules was made at this point. These are considered as functionalities of the main project. These utilities are going to have well defined inputs and outputs to achieve the tasks, which improves coherence of modules. The common functionality such as image registration, that the three modules need, will be another module, by this way common module and any one of the three can work independently. These three modules will not keep their own data, they will only operate on data provided by the project manager. On the other hand in our project the user will have all of the functionalities in one project, so for our project there will be also a manager module. Achieving this will increase usability of the program, such that user can use all the modules which are DEM generation, orthorectification and mosaic generation without changing the project.

Another module of the PPS is the graphical user interface, which basically consists of an MDI image viewer, which will be an improved version of the first prototype. The main consideration over the design phase of this module is to separate the GUI from the other modules. This module interacts only with the project module to get or set the required data, and this provides a high level of abstraction from the other parts of the project. This abstraction gives the project the ability to change the GUI without changing any other class.

Data module is a unique module assigned to a project. It keeps all the data used by the project. Images, videos etc. are all kept in this module. This module interacts only with the main project module. When a module needs a data, it requests the data from the project module, and the project requests this data from the data module. Then the project module takes the data coming from the data module, and passes it to the requesting module. This abstraction seems to cause a redundant amount of data traffic between modules, but this abstraction comes with a degree of freedom. This freedom gives the project the ability to change the data module without changing any other module. In any future phase of the project, a more flexible and efficient database module can replace the current module.

A particular example that elaborates on the data traffic mentioned above is presented next. This example will illustrate the overall architecture in a concrete manner.

- The user issues a DEM generation call.
- The GUI sends a message to the project module, relaying the DEM generation request. Relevant data (image ID's etc.) is passed along with the message.

- The project manager module checks its current context to decide if the requested computation (DEM generation in this case) is applicable. If not, the request is stalled. For example, if the input images are not registered to each other prior to the DEM call, the project manager will stall the DEM generation process. It will send a message to the GUI module describing the exception and will ask for user authorization to proceed with the possible recovery alternatives. In the DEM case a possible recovery alternative is to register the images. If the user approves a recovery method, the project manager makes the required calls for the selected method and continues processing the request.
- The project module calls the DEM module to perform the requested computation. Pointers to the actual data (pointers to the image matrices in this case) are passed to the DEM module in this call. To obtain these pointers, the project manager consults the data manager module.
- The DEM module performs the requested computation and issues a request to the project manager module to save the result it generates.
- The project manager saves the DEM result by a call to its data manager module.

Note that although this traffic back and forth between the project module, the data manager module and the DEM module seems redundant, it decouples the data manager from the DEM module. The DEM module does not need to know how data management is handled by the project module.

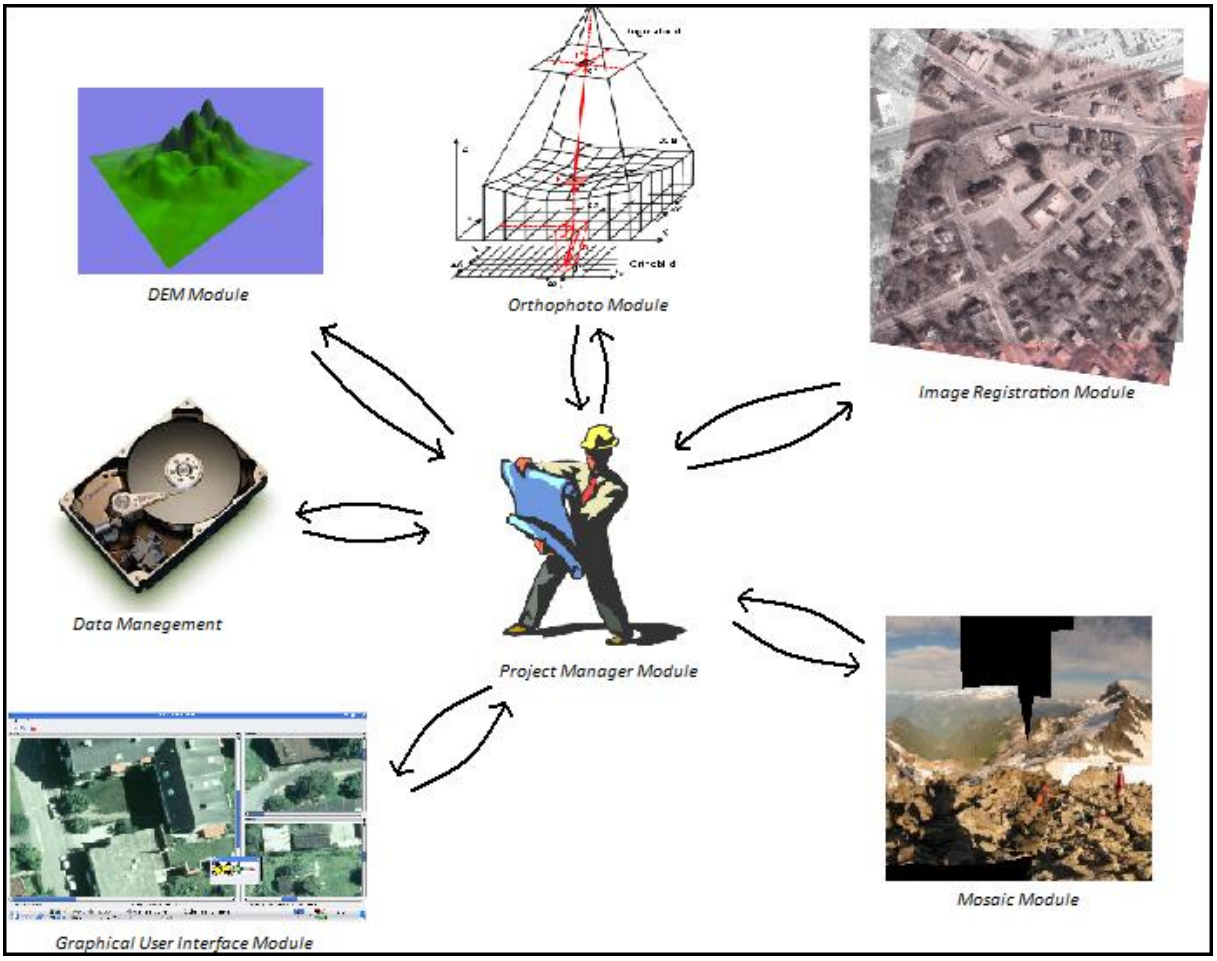

Figure 4- 1 : Diagram showing interactions between modules

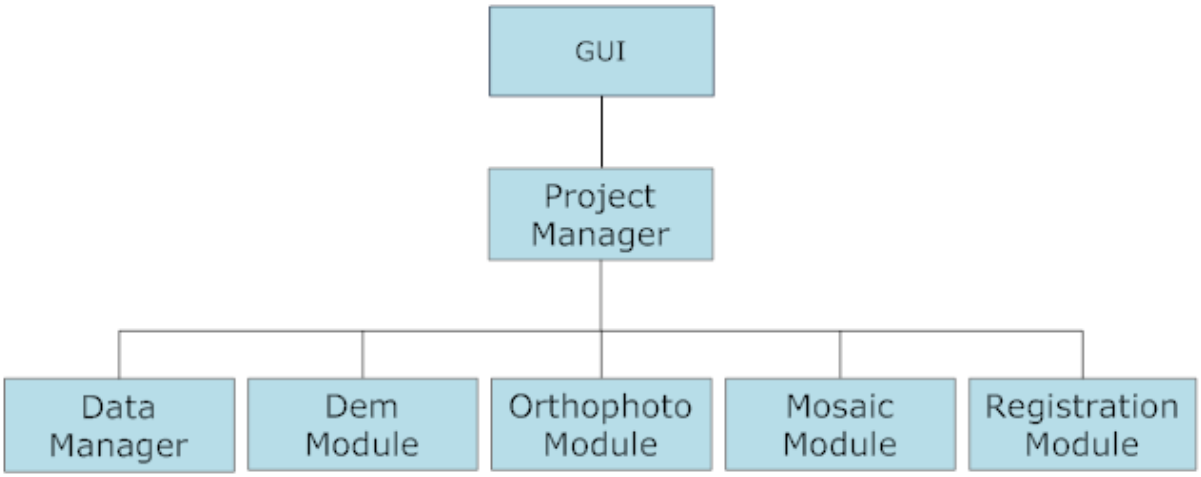

Figure 4- 2 : Module hierarchy

5. DATA DESIGN

5.1. Data Organization

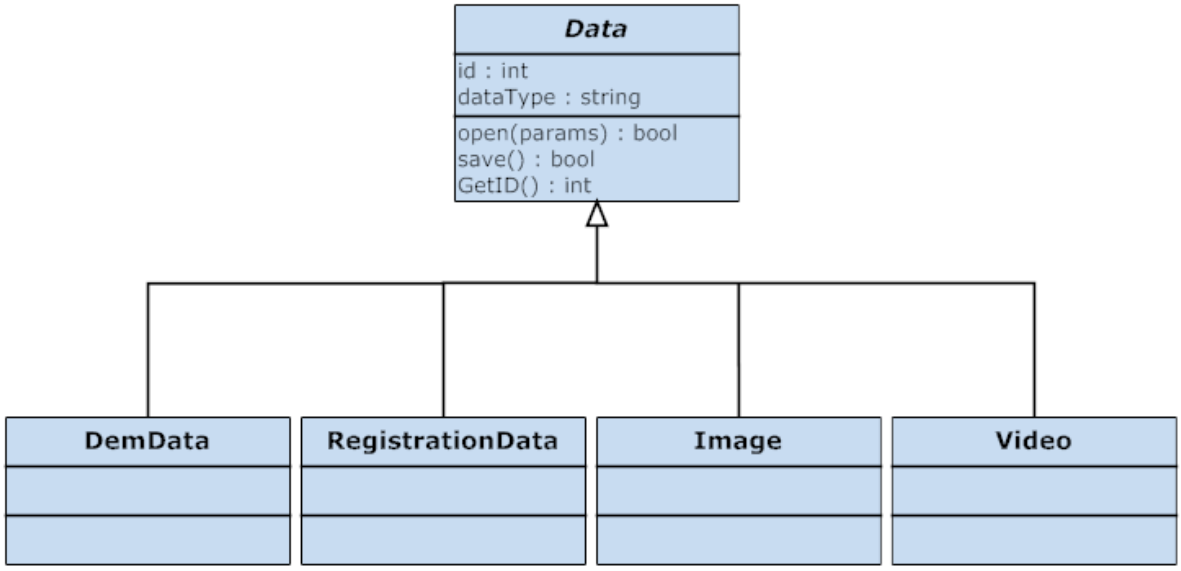

Figure 5- 1 : Inheritance Diagram of the data structure

Data used by the project manager can be classified into 4 elementary kinds: Images, DEMs, videos and image registration data. Each is represented by a class in the data hierarchy. All classes are extended from an abstract data class, which presents common functionalities of the four kinds as an interface. These common functionalities are opening and saving of files that store the serialized data of the corresponding class. The functions that make up this interface will be overridden in each class to provide the relevant semantics. This class hierarchy makes it possible to simplify the design of the project manager. Each of the four elementary classes is explained below in greater detail.

5.1.1. Image Data

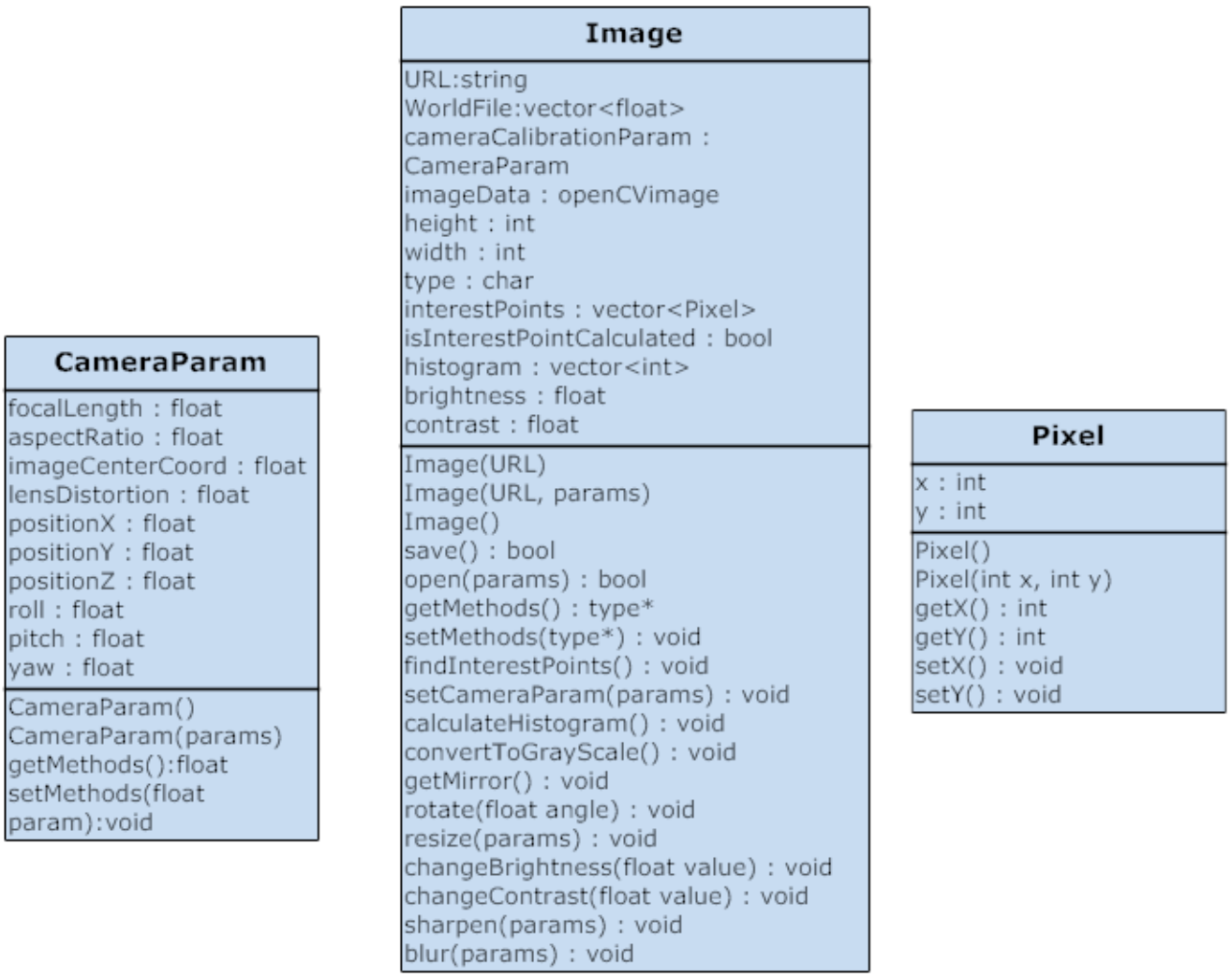

Figure 5- 2 : Class Diagram of the Image data class

Image data structure consists of the information about the image according to requirements of the process modules. Images which are included to the project will be member of the manager project module. Image class will be capable of calculating required statistics and information about the image itself. By this way image data will be compact. Storing all relevant information in the image class allows different modules reach these data.

OpenCV library is going to be used to store and process images. Images are going to have unique ids, which will be taken from the data manager of the project. Using these ids, modules can reach desired image and can store relative image information such as homography matrix. Camera calibration parameters which are needed by various modules for processes are also stored in this class. In basic image view mode these are not needed. So images can be constructed and showed without these parameters. In order to avoid any conflict, data validity of these will be checked before usage. Storing camera calibration parameters in a different class will allow calculations be done in this class and make project more understandable.

This class will also open the world file associated with the image and then read the data from it. It will store this data and reuse it for image to earth registration. The file information will also help to show pointed pixel's earth registration information on the status bar which is located at the bottom of main window.

Image input/output will be done by this class using abilities provided by GDAL library. This gives the project the ability to support a broad range of image file formats, where some of them being specially designed for geospatial images, and some of them being very common.

This class has some methods for processing the contained image data, such as brightening or sharpening the image. To provide this additional ability to the end user, the abilities of openCV library will be utilized in this project.

Another important data stored by this class is the interest points of the contained image. These points are used in the calculations to register two images and are of much importance. The importance of these points increases the importance of the algorithm used to calculate them; therefore Harris Corner Detection algorithm will be used.[8] The decision to use this algorithm was affected by the fact that it is widely used and its success is approved.

One of the main considerations during the process of designing this class was performance. In order to achieve this purpose, image processing will be done on the Fourier domain, which shows greater speeds than processing in the spatial domain.[7]

5.1.2. Video Data

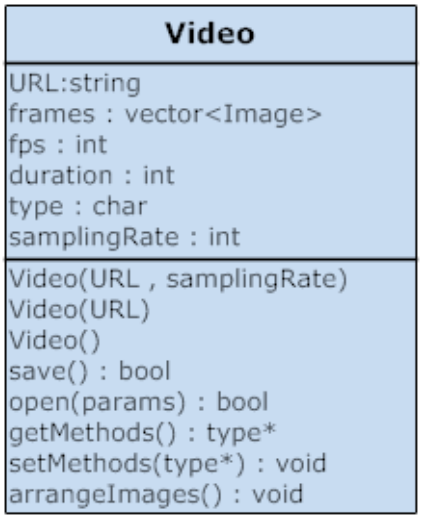

Figure 5- 3 : Class Diagram of the Video data class

Video Data class is designed to support video mosaic feature. It is basically an array of images taken from the specified video file. It captures a number of images from the video file using the given sampling rate. If a sampling rate is not specified, it uses the default value provided. Using all the frames contained in the video file requires a lot of time and is redundant; therefore a sampling functionality will be provided. The images taken from the video according to the specified sampling rate will be kept in a vector of images.

5.1.3. Registration Data

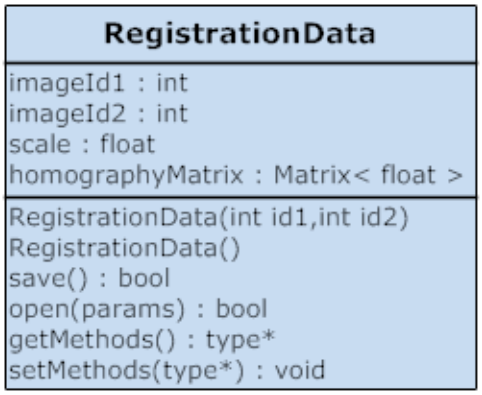

Figure 5- 4 : Class Diagram of the RegistrationData data class.

Registration data is the class that contains the data calculated by the registration module. It can contain image-to-image or image-to-earth registration data. It stores the unique image ids for image-to-image registration or a unique image id and a special id reserved for earth registration. Homography matrix is used only in image-to-image registration; for image-to-earth registration only the worldfile and scaling data will be used.

5.1.4. DEM Data

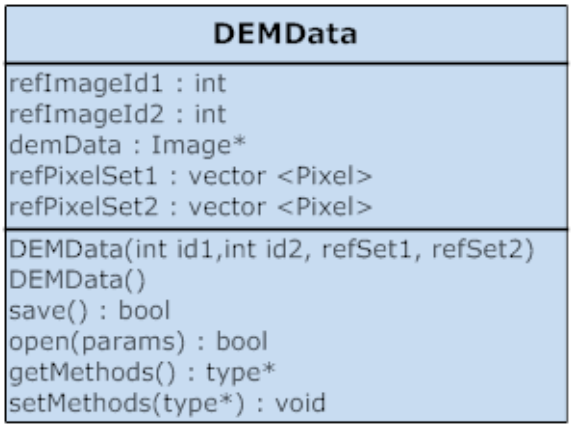

Figure 5- 5 : Class Diagram of the DEMData data class.

DEMData class is designed to be a container for the actual DEM data represented as an ordinary image. This data can either be the data coming from a DEM file, or the output coming from the DEM module. This class keeps the IDs of the images it is created from, or these fields keep a special id reserved for representing NULL. Reference pixel sets keep 4 pixels representing the overlapping area of the respective images.

5.2. Overall Data Design

Data flow between user and program and inside program is described using data flow diagrams. Data flow diagrams for levels up to there were drawn. Since in this project modules are designed as algorithmic process modules, all of them are not exploded. These are described in more detail in the modules/subroutines parts of the project

5.2.1. Level-0 Data Flow Diagram

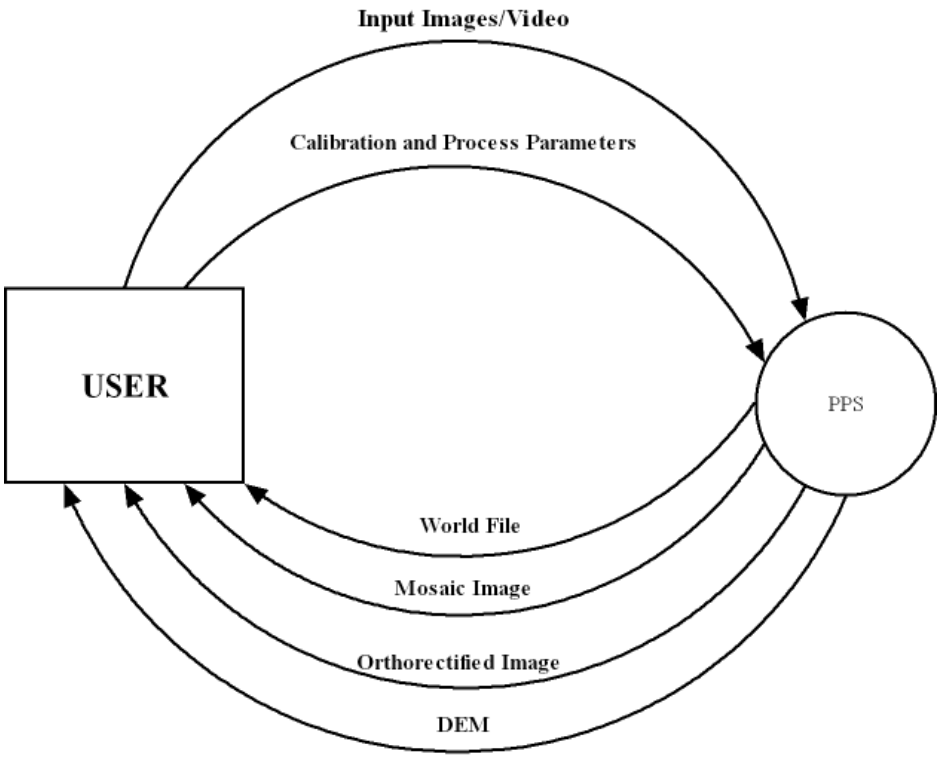

Figure 5- 6 : Level-0 DFD

Level-0 DFD is shown in the figure above; this is an overview of the system in general. Interaction between user and system is described. PPS is capable of different functionalities, and most of these have parameters decided by the user, these are shown as process parameters, and given in detail in the preceding parts of the report.

5.2.2. Level-1 Data Flow Diagram

INPUT:
- Input Images/Video
- Calibration and Process Parameters

OUTPUT:
- World File
- Mosaic Image
- Ortohorectified Imaga
- DEM

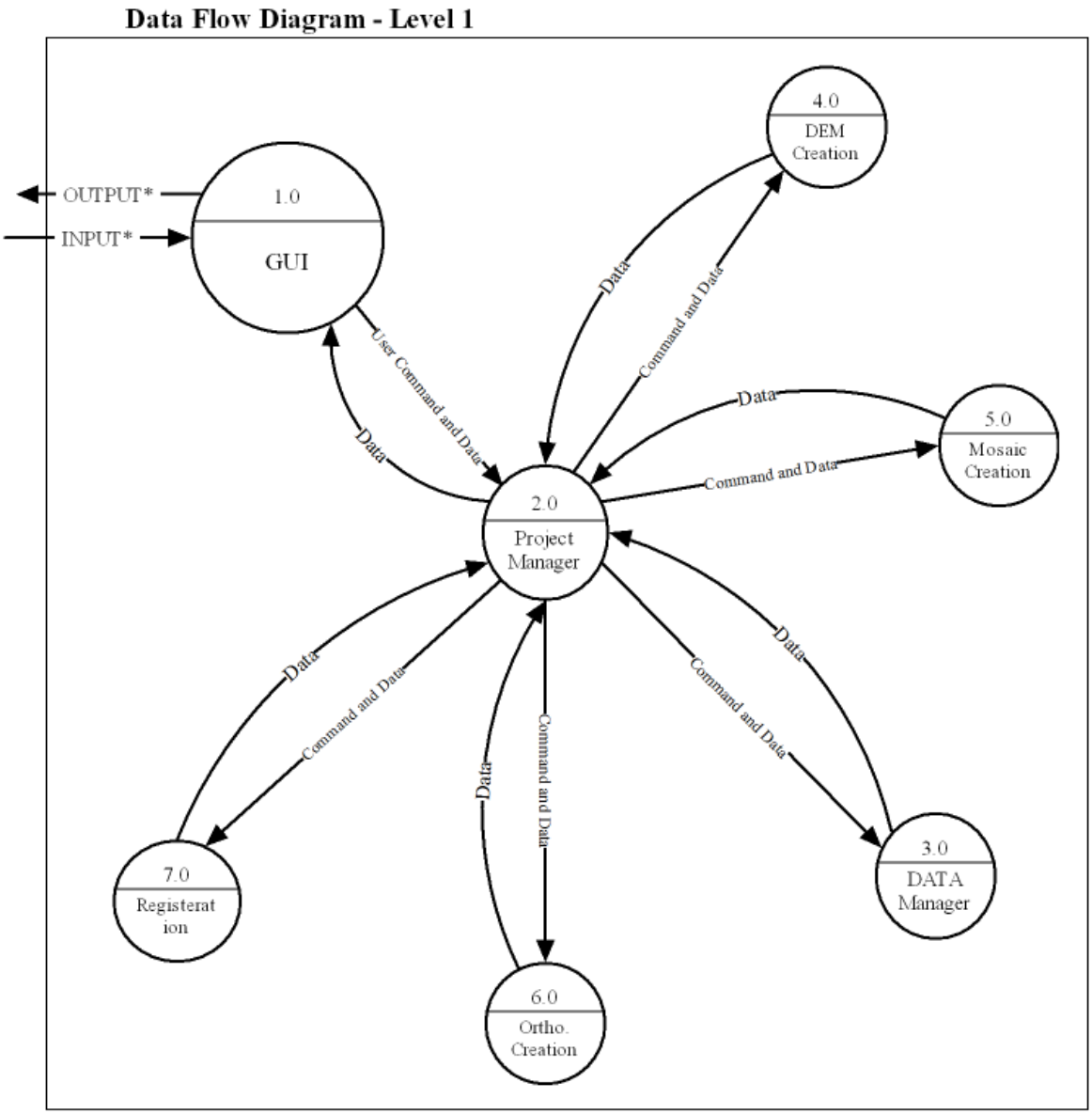

Figure 5- 7 : Level1- 1 DFD

At the heart of the Data Flow in PPS stays Project Manager, which provides required data to modules such as Mosaic Creation, Registration, etc. whom are responsible for image processing. Project Manager gathers input information from GUI, and organizes data with the help of DATA Manager. Information needed for image processing is requested to Project Manager by Photogrammetry modules. Project Manager provides this information from DATA Manager, also results of Photogrammetry modules are send back to DATA Manager with supervision of Project Manager.

Explanations;

- GUI collects user provided Input data, and performs requested operation data flow to the user.
- Between GUI (1.0) and Project Manager (2.0) data flows shown in figure Level2 – 1,
- Between Project Manager (2.0) and Registration (7.0) data flows shown in fig. Level2-7,
- Between Project Manager (2.0) and Orthocreation (6.0) data flows shown in fig. Level2-6,
- Between Project Manager (2.0) and DATA Manager (3.0) data flows shown in fig. Level2-3,
- Between Project Manager (2.0) and Mosaic Creation (5.0) data flows shown in fig. Level2-5,
- Between Project Manager (2.0) and DEM Creation (4.0) data flows shown in fig. Level2-4.

5.2.3. Level-2 Data Flow Diagrams

5.2.3.1. Project Manager DFD

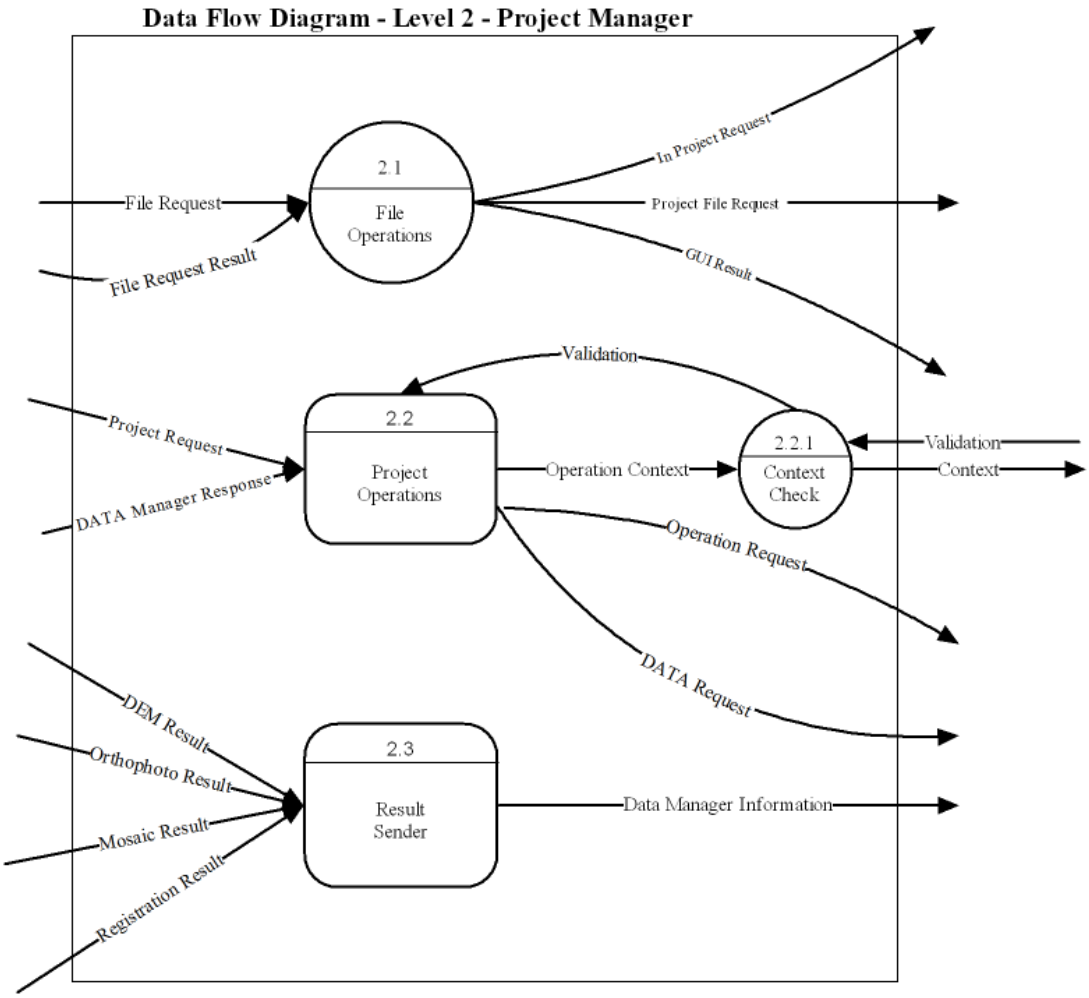

Figure 5- 8 : Level 2 DFD

Details of the data flow in the project manager are given above. Capabilities of the project manager are grouped into three main set of processes. File operations includes creating a new project, saving or loading an existing project and closing the project. Creating and loading project includes constructing process modules and data manager. To save a project, project manager saves the current status and data manager saves all information since other modules do not save any data about the status of the project they do not hold a place in this procedure. Saving details are described in the file structure. Project operations respond user requests about sub processes. According to the request related sub process invoked. Since no other module interact with each other results of the sub processes are also sent to the project manager, these data are sent to the data manager and through GUI sent to the user.

5.2.3.2. GUI DFD

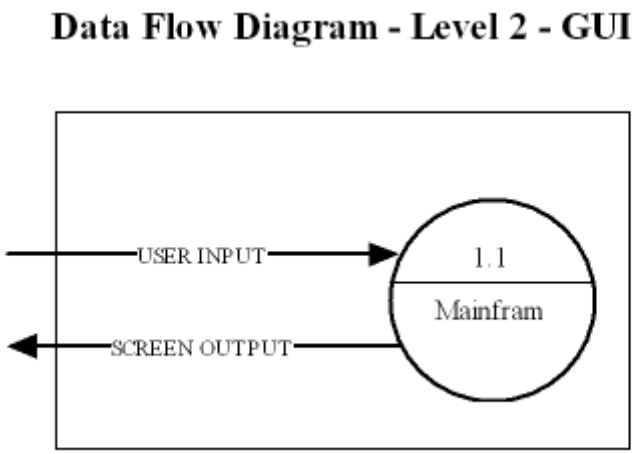

Figure 5- 9 : Level 2 DFD

User requests, such as Photogrammetry functions, image property edits, project operations are transformed to internal messaging with the help of GUI. Data flow between GUI and Project Manager provided with Mainframe.

5.2.3.3. DATA Manager DFD

Project Manager and DATA Manager Modules, sends each other data which is required for image processing. File manager keeps these data organized under four sections.

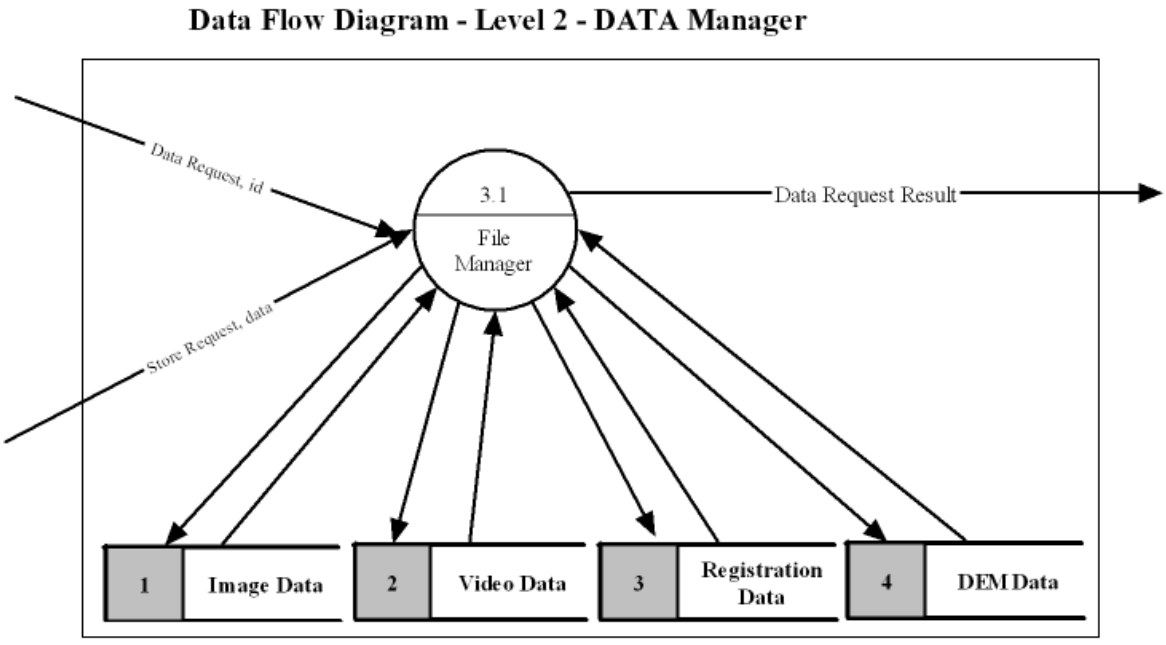

Figure 5- 10 : Level 2 DFD

5.2.3.4. DEM Creation DFD

DEM Generator receives data for requested process. Incoming data are sent by Project manager and outgoing data are collected by Project manager to be sent to DATA Manager.

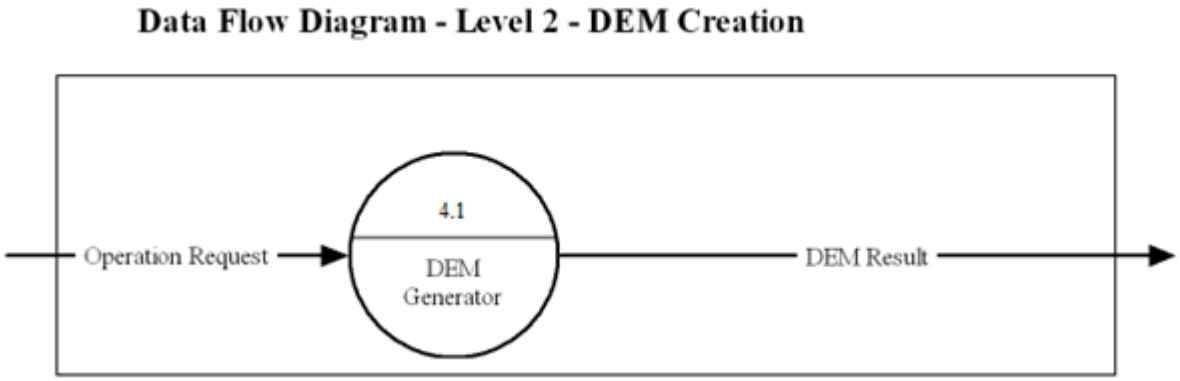

Figure 5- 11 : Level 2 DFD

5.2.3.5. Mosaic Creation DFD

Mosaic creation has a similar data flow as other Photogrammetry modules. Data manager sends input data for requested operation, and collects the outcomes from Mosaic Generator process. Mosaic result is kept by Data Manager with the help of Project Manager.

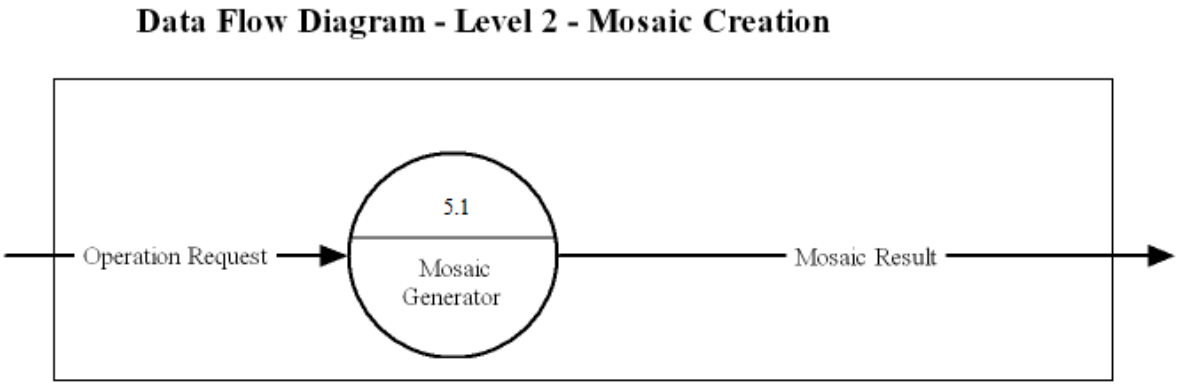

Figure 5- 12 : Level 2 DFD

5.2.3.6. Orthocreation DFD

Orthocreation receives input data, Operation Request as called in the diagram, from Project Manager. Results of Orthophoto Generator process are transported to Data Manager by Project Manager.

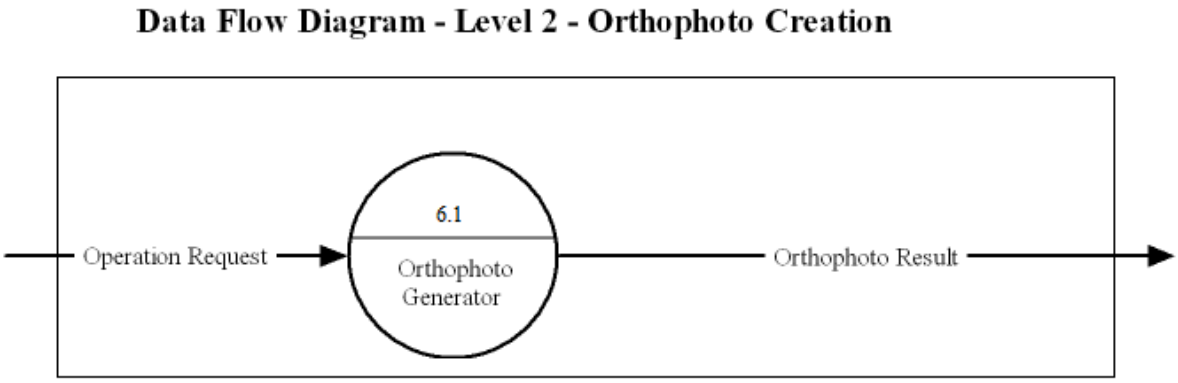

Figure 5- 13 : Level 2 DFD

5.2.3.7. Registration DFD

In registration data flow, incoming data, called Operation Request, is brought by Project Manager to the Register Generator. As the same procedure in other modules outputs from the register generator are carried by the Project Manager to the Data Manager to be kept in organization with other files.

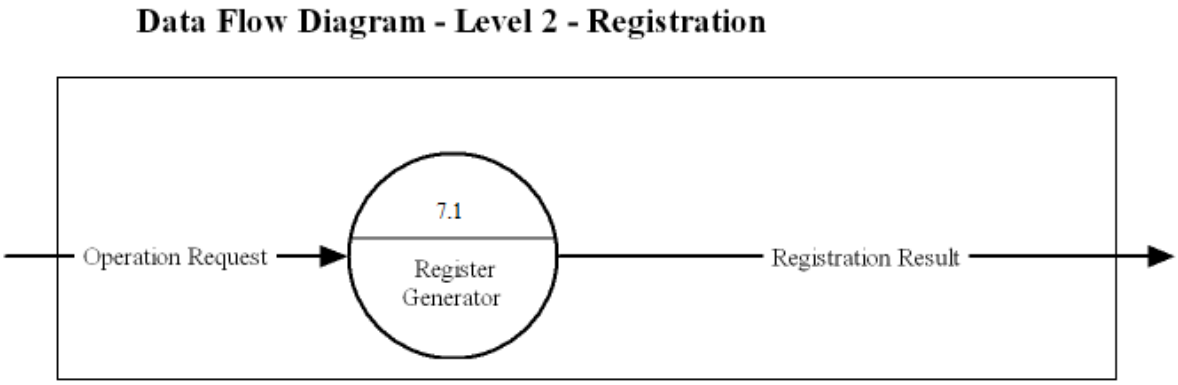

Figure 5- 14 : Level 2 DFD

5.2.4. Level-3 Data Flow Diagrams

5.2.4.1. GUI – Mainframe Data Flow Diagram

User requests over Main Toolbar, Image Toolbar are collected to be processed by Mainframe's sub processes. Some of these process' outputs are sent to panel to be shown by the output device, and some other outputs are carried between sub processes for further work as shown in GUI – Mainframe DFD.

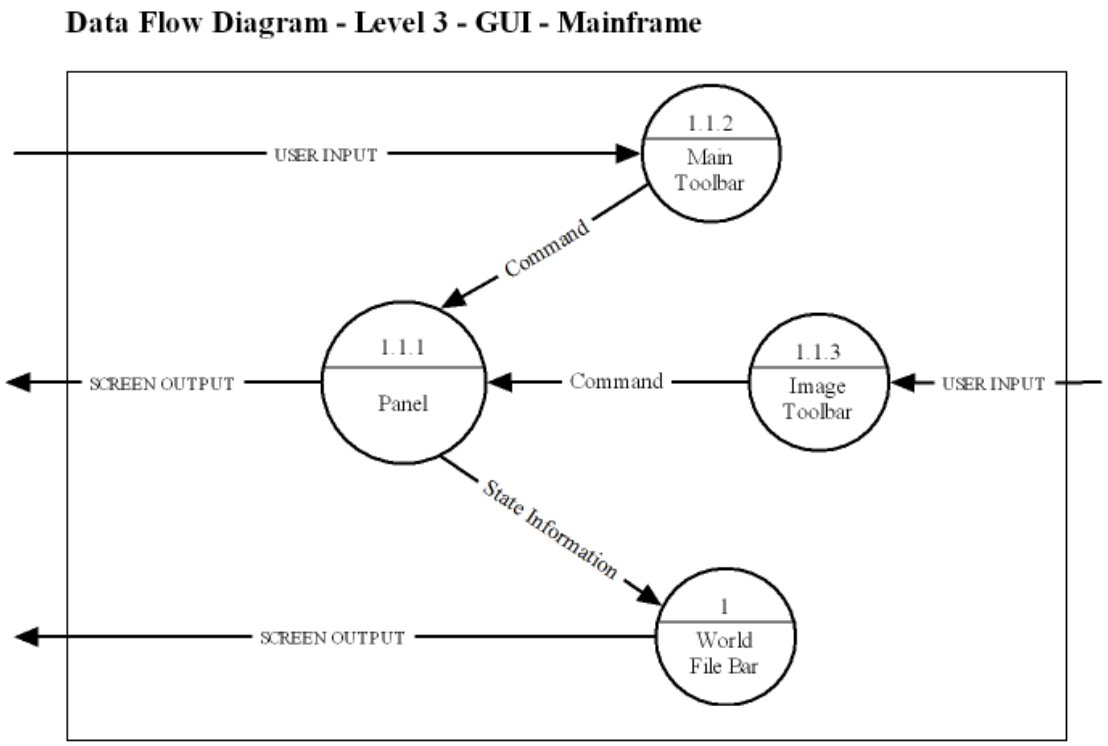

Figure 5- 15 : Level 3 DFD

5.3. File Structure

PPS will have saving and loading capabilities of the project. To support this, PPS needs to save calculated features, matrices and all relevant data. Every data class needs a save and load format. Saving all relevant data for images will be done using image ids.

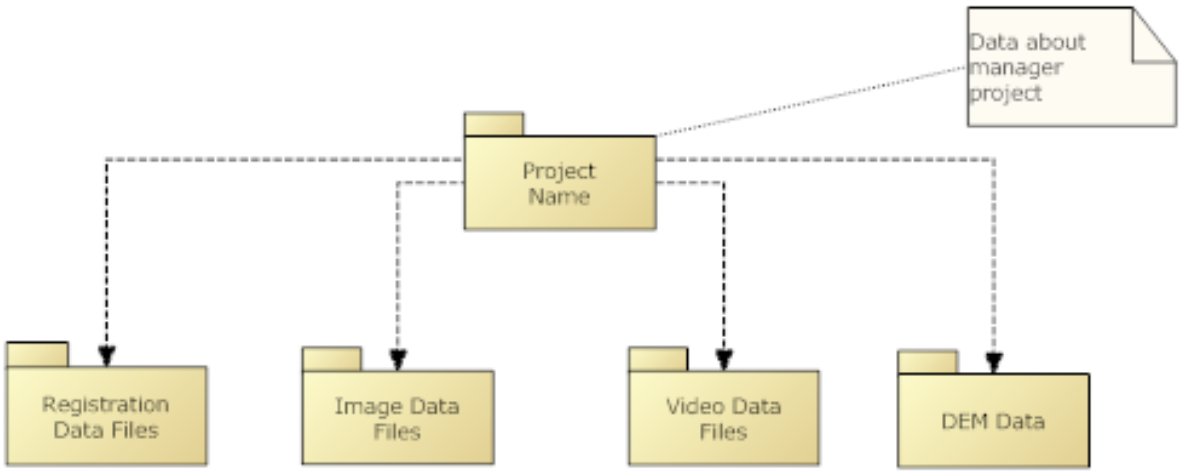

Figure 5- 16 : File Structure

For avoiding recalculation of phases processed, PPS will have save and load features for different file types and workspaces. There will be four different data types, for better organization and faster use these four data types will be collected under four different folders. For every different project there will be a file which holds information about processed images. This information consists of; processed images' unique id's, results of calculations done on these images, last status of every process, etc. Additional saving

options will be provided according to requests of target users' prototype testing phases. With the help of these files; handicap of recalculation for phases will be exceeded which will provide less time consuming software.

Under a project, user will be able to save the result of every function in a desired format. For Mosaic and Orthophoto modules user will be able to save the results in the same format of processed image; or from a variety of choices. For DEM module the user will be able to select the output format from commonly used DEM formats list.

According to USGS DEM technical details, a DEM file format will be as the following;

- Type A record: header information, one per file,
- Type B record: made up of data from one-dimensional arrays called profiles, one per line of elevation data,
- Type C record: contains statistics on the accuracy of the data in the file,
- Logical record size of 1,024 bytes,
- More than one record is usually required to store a single record type B.

6. INTERFACE DESIGN

6.1. Functionality / User analysis

Before presenting the detailed design of the user interface, some key requirements are presented. PPS must provide the necessary interfaces for the following vital functionalities;

File operations,
- Opening/Closing images
- Saving/Loading images
- Opening/Creating/Saving/Closing projects

Photogrammetry operations,
- Creating DEM,
- Creating Orthophoto,
- Creating Mosaic,

Fundamental image processing operations,
- Image Blurring and Sharpening
- Converting to Grayscale
- Histogram Generation
- FFT and IFFT support
- Elementary Filtering
- Image Mirroring
- Image Rotation
- Upsampling / Downsampling
- Brightness Adjustment
- Contrast Adjustment

6.2. Interface

A simple sample of PPS interface is shown below. Main toolbar, image window (including children windows), image toolbar and world file bar are basic components of the main interface. Detailed information is given in subsections. As a rule of thumb for this project all contents of contents will be kept simple and clean, for fast and user-friendly use.

Figure 6 - 1 : GUI outline

6.2.1. Main Toolbar

This bar includes basic file operations over the images and projects such as; opening, closing, saving etc. Help menu, software preferences (theme, window sizes, fonts) are available in main toolbar. Additional content may be added according to flow of project.

Main functionalities of PPS (Orthophoto, DEM, Mosaic creation) are performed under main toolbar.

6.2.2. Image Window

This area is reserved for currently being used image, and derived images from it. Image window can be thought as a workplace for functionalities of PPS. Created Mosaics, DEMs, Orthophotos and the image set of the current project will be displayed in image window area.

6.2.3. Image Toolbar

Image toolbar contains shortcuts to the most frequently used image processing operations. Clicking on these shortcuts will perform the selected image processing task on

the image displayed by the active window. The ordering of the shortcuts (i.e. image processing functions) will be arranged in such a way that the most commonly used tasks will be placed first. This ordering will be determined according to user analysis surveys.

6.2.4. World File Bar

World file bar is a status bar which shows relevant information about the current project as well as specific information about the current active window's contents. Most commonly, the active window will be displaying an image and the status bar will be showing the world coordinates of the location pointed by the mouse pointer on the image.

6.3. Interface Class Design

6.3.1. Overall Discussion of the GUI Class Hierarchy

The GUI is designed as a collection of classes with a certain hierarchy. The main frame of the GUI is an instance of the MainFrame class, which is derived from the wxWidgets' wxMDIParentFrame class. A MainFrame object contains the statusbar, menubar, toolbar and the image processing toolbox as its fields. A children window vector is also stored in the MainFrame object. Children windows themselves are dockable panels. A children window may be a DEM panel, image panel, video panel, DEM creation dialog panel, registration dialog panel, mosaic dialog panel or an orthophoto panel. Each of these panels is represented by a class. These classes are derived from a base abstract Panel class, which serves as an interface for the common functionalities that are offered by each of these panels. These common functionalities include common event handling methods and the display method (called show() in the diagrams). Instances of these classes communicate with each other by sending event objects to each other. These event objects are instances of the Event class, which is an extension of the wxWidgets' wxCustomEvent class. The overall hierarchy is depicted in the figure below.

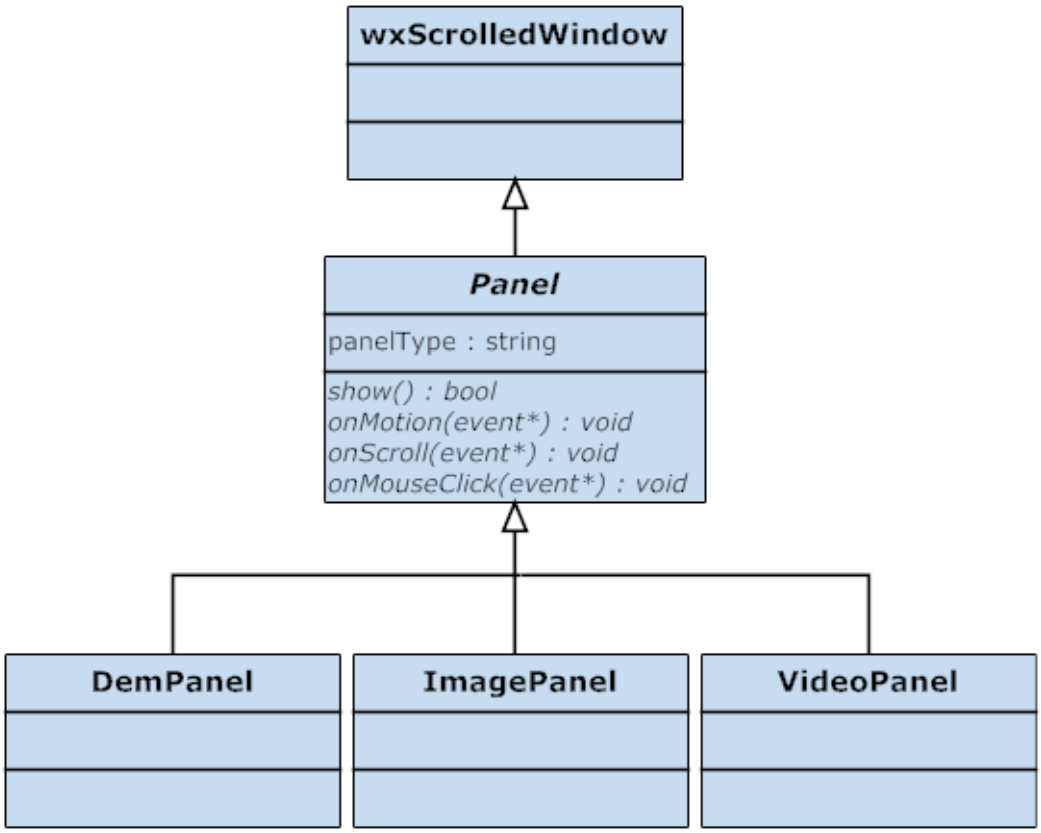

Figure 6 - 2 : Inheritance Diagram showing extensions of Panel class.

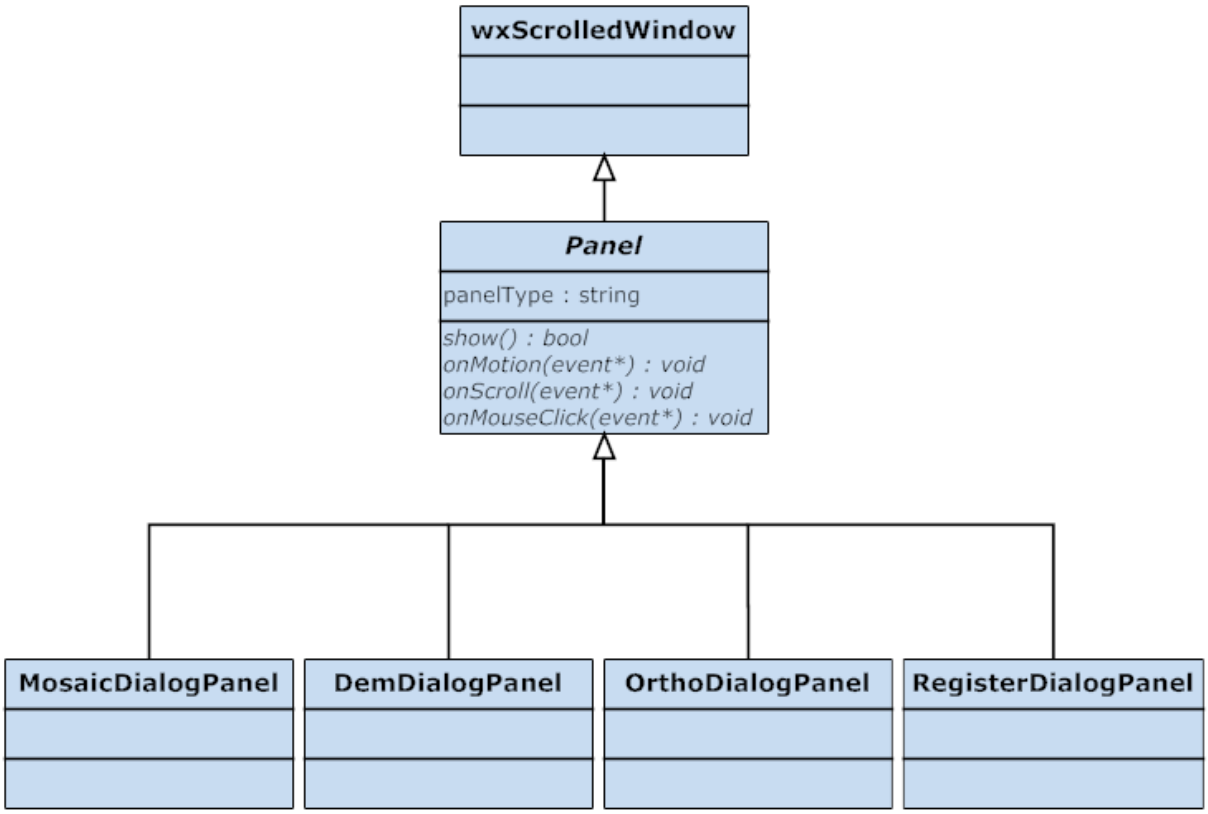

Figure 6 - 3 : Inheritance Diagram showing extensions of panel. The top two figures are essentially the same diagram but are shown separately for the sake of simplicity and understandability.

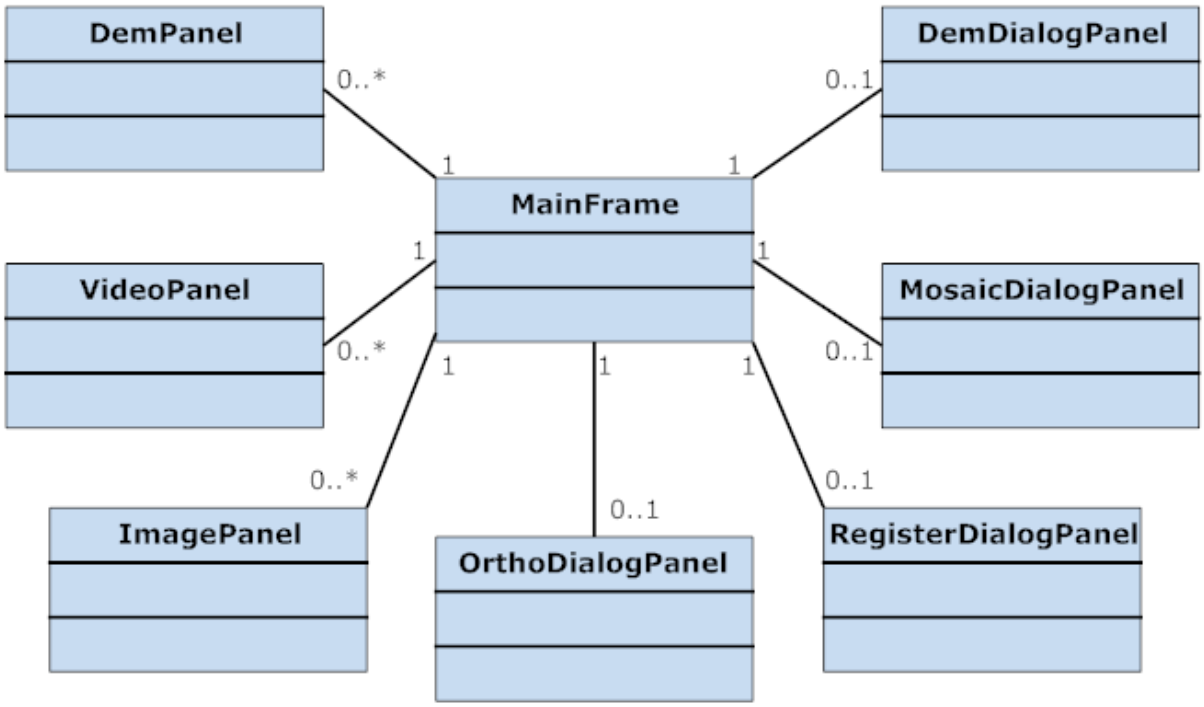

Figure 6 - 4 : Class Diagram showing the cardinalities of panels belonging to the MainFrame class.

6.3.2. ImagePanel Class

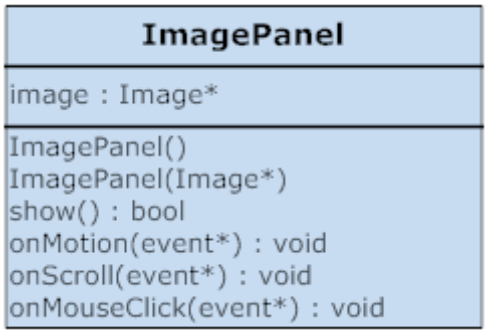

Figure 6 - 5 : Class Diagram of ImagePanel GUI class.

Instances of this class are dockable panels that display images belonging to the current project. This class encapsulates the necessary methods that convert the images into a suitable format that can be displayed on a device context (DC).

6.3.3. VideoPanel Class

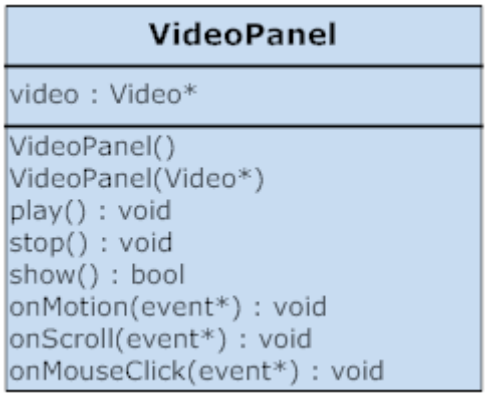

Figure 6 - 6 : Class Diagram of VideoPanel GUI class.

Instances of this class are dockable panels that display videos belonging to the current project. Like the ImagePanel class, this class encapsulates the necessary methods that convert videos into a suitable format that can be displayed on a device context (DC). This class also includes methods that implement the common video playing functionalities (pausing, rewinding etc.)

6.3.4. DemPanel Class

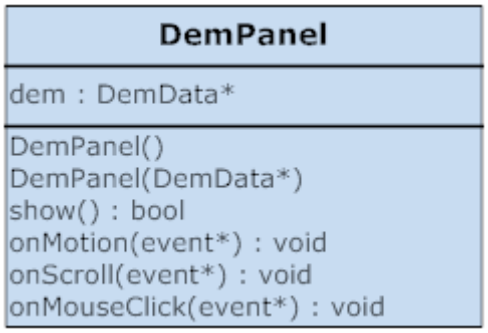

Figure 6 - 7 : Class Diagram of DemPanel GUI class.

Objects of this class are specialized panels that are able to render DEM's as 3-D surfaces. This class uses the OpenGL library to accomplish the rendering task. The overridden event handlers of this class relay relevant user inputs (such as rotation, zooming etc.) to OpenGL.

6.3.5. RegisterDialogPanel Class

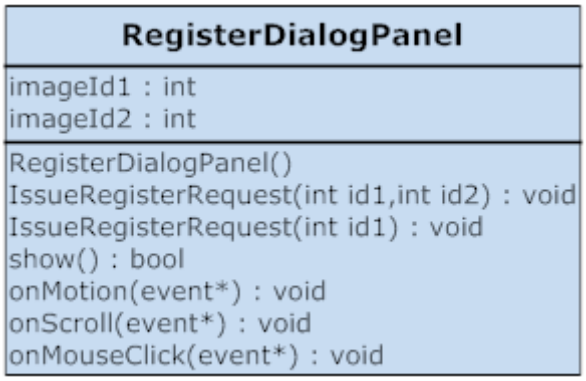

Figure 6 - 8 : Class Diagram of RegisterDialogPanel GUI class.

Objects of this class present the user the necessary interface to issue an image registration task. The user may choose to register an image using a worldfile, or register two images with respect to each other. These options are provided by the overloaded method IssueRegisterRequest(). The standard event handlers are overridden to capture the user input in the desired manner.

6.3.6. DemDialogPanel Class

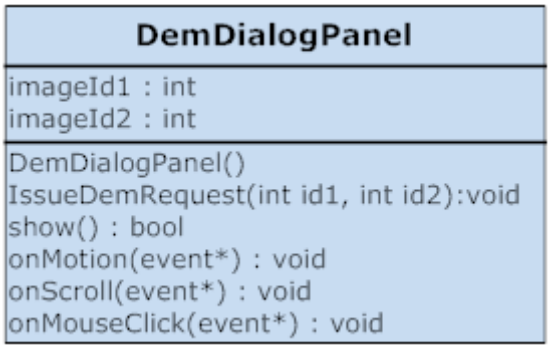

Figure 6 - 9 : Class Diagram of DemDialogPanel GUI class.

Objects of this class are dialog panels that present the user the interface to issue orthorectification tasks. Standard event handlers are overridden in the same manner.

6.3.7. OrthoDialogPanel Class

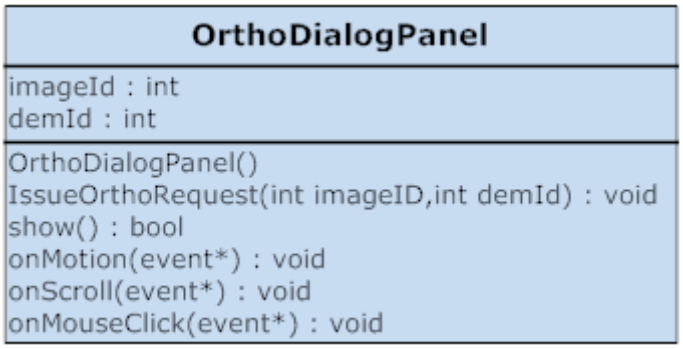

Figure 6 - 10 : Class Diagram of OrthoDialogPanel GUI class.

Instances of this class are dialog panels that present the user the interface to issue DEM creation tasks. Standard event handlers are overridden in the same manner.

6.3.8. MosaicDialogPanel Class

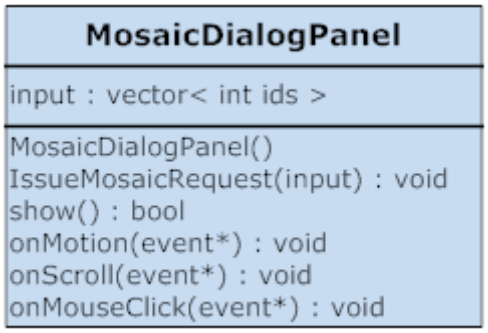

Figure 6 - 11 : Class Diagram of MosaicDialogPanel GUI class.

Objects of this class are dialog panels that present the user the interface to issue image mosaic creation tasks. Standard event handlers are overridden in the same manner.

6.4. Usability Testing

Before the final release of PPS, several prototypes will be created and tested by actual users with a talk aloud protocol, where thoughts of actual users about PPS interface will be gathered and processed in a positive manner.

7. MODULES AND PROCESS DETAIL

7.1. Overall Discussion of the Module Hierarchy

PPS performs user requested operations over user supplied inputs, which puts PPS's GUI module over all other modules in the hierarchy and workflow. Project Manager which works with requests of GUI Module is the task distributor. From a top view, PPS can be seen as a

circle with Project Manager in the center, and GUI as a messenger to the Project Manager Module. On the circle there are helper modules which are directly connected to Project Manager to operate requested tasks; Data Manager Module, DEM Module, Orthophoto Module, Mosaic Module are the elements of helper module set. According to the distributions made by Project Manager, helper module receives messages for performing requested operation and sends back meaningful answers to be stored in Data Manager Module over the Project Manager Module.

7.2. Project Manager Module

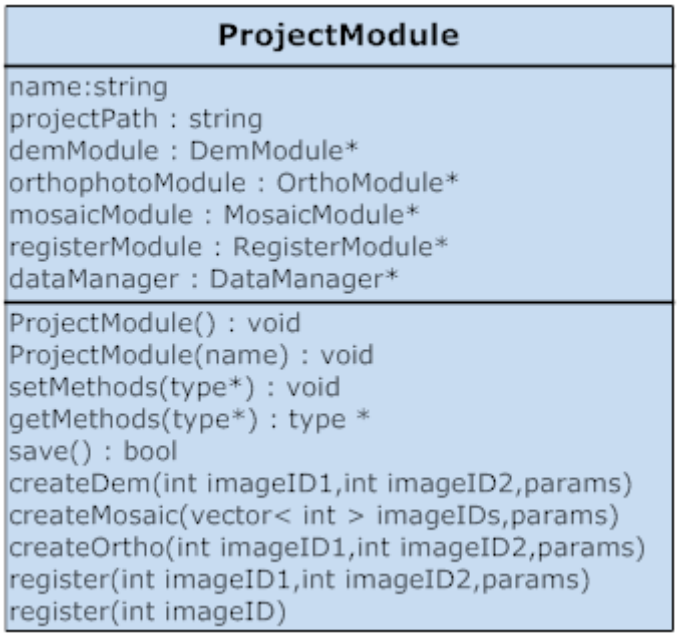

Figure 7 - 1 : Class Diagram of ProjectModule class.

Project Manager is the main module of the PPS. It handles all the data traffic and process flow between modules. User has to open or create a project to be able to use the functionalities of the PPS. After creating a new project, user can open multiple files to work with. The project does not handle the data requests, instead it issues commands to the data manager to open or save the data.

Project Manager keeps pointers to different modules. Each of these modules are specialized to accomplish one functionality. Details of these modules will be explained below. The createDem(), createMosaic(), createOrtho(), register() and save() methods are called by the GUI, and then this class issues the required commands to corresponding modules.

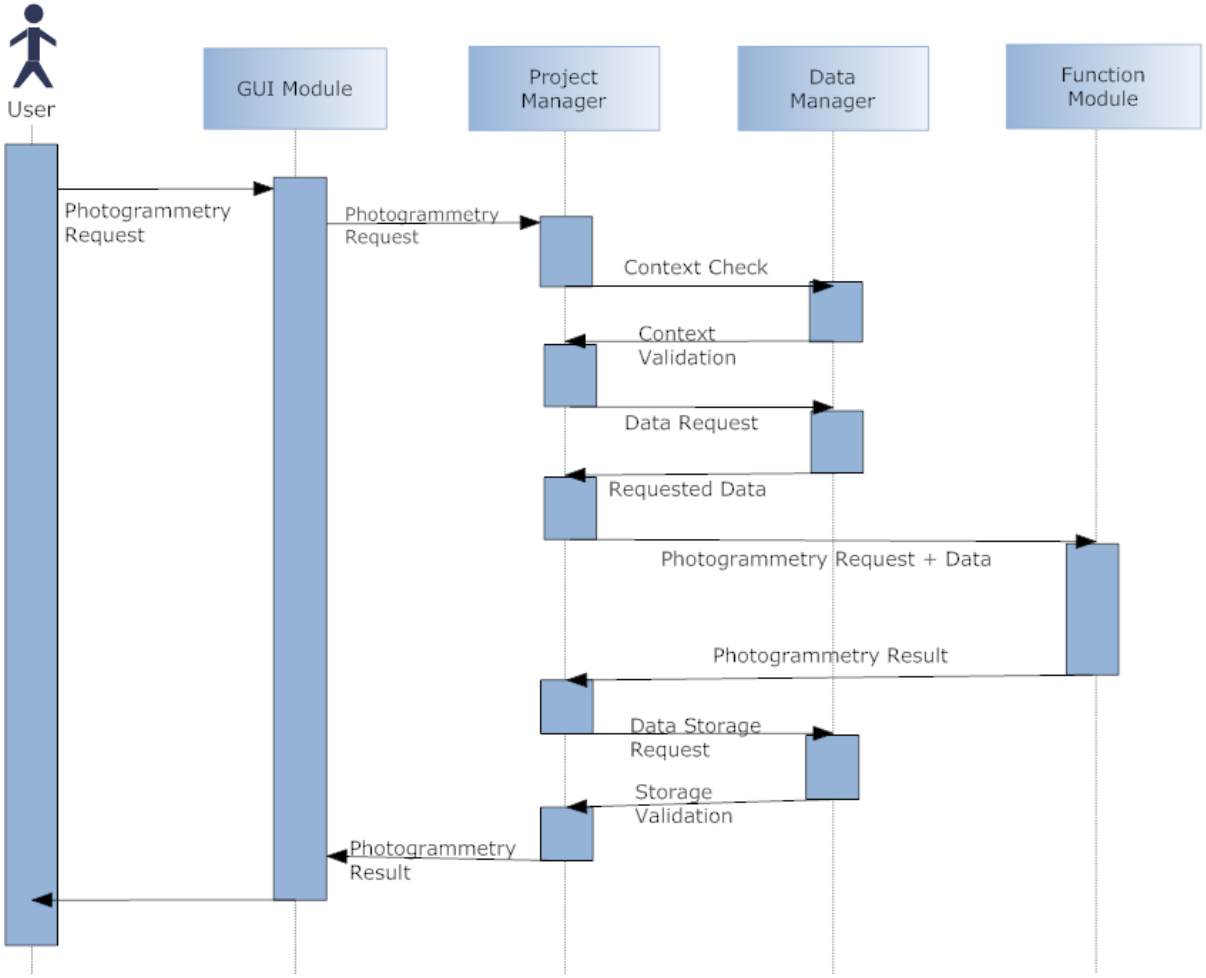

Figure 7 - 2 : Sequence Diagram of the process flow.

The GUI module issues any command to the project manager for any request coming from the user. Then the project module checks the context if the request can be fulfilled or not. If the request is valid, then the project manager requests the needed data from the data module. When the data module returns the required data, the project manager issues the appropriate command to the corresponding module. When the called module finishes its job, it returns the result to the project module. The project module then issues a command to the data manager to add the result to the data managed. When the data is stored in the data manager, then the project module returns the result to the GUI, which displays the result to the user.

7.3. Data Manager Module

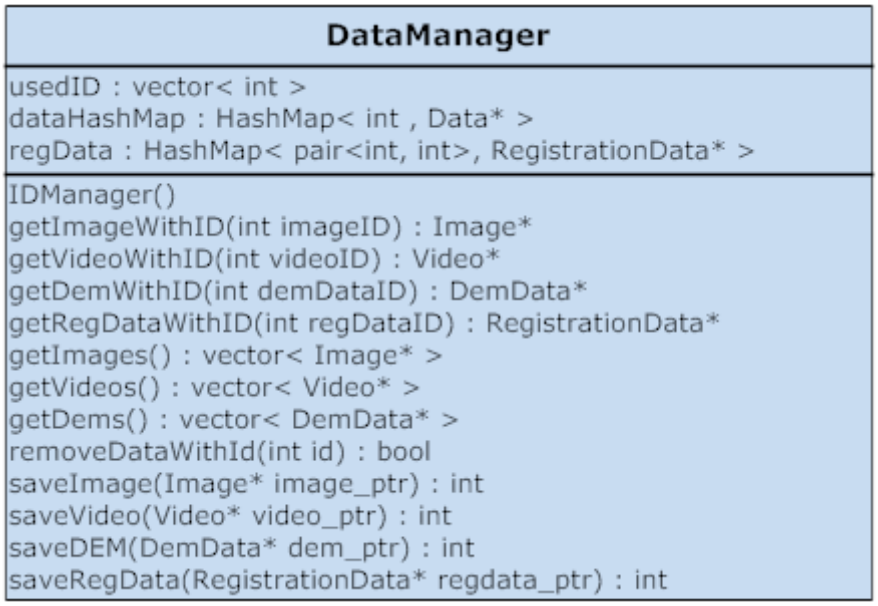

Figure 7 - 3 : Class Diagram of DataManager module class.

All data and file operations of the PPS are handled by this module. According to requests of the project manager, data manager responds with the appropriate data. This module also works as a unique id generator for stored data. When a storage request comes, a new id is generated and assigned to the corresponding data. The id is used as the key to the HashMap, and the data pointer is stored in the HashMap. When this process is complete, the unique id for the data is returned to the project manager.

A similar process is followed when the user requests a data removal through the GUI and the project manager. All the fields related with the corresponding data such as registration data belonging to this id are removed along with the data itself. When this process is complete, the unique id of the removed data is no longer needed, and it is recycled for future needs.

7.4. DEM Module

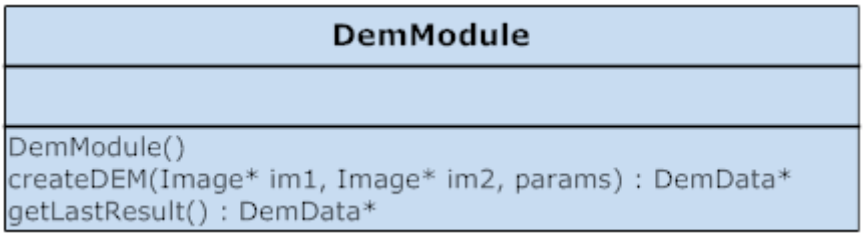

Figure 7 - 4 : Class Diagram of DemModule class.

This class acts like a wrapper around the functionality. It does not store any specific data.

Common DEM algorithms work on stereo images. However, this class is designed to work on non-stereo images. If a stereo input is given, their output can also be calculated by assuming them as non-stereo.

This class only works on overlapping areas of the images, therefore images need to be registered before coming to this module. Camera calibration parameters for both images are needed to process the images. For each pixel in the overlapping area, using information taken from two images, position of the point in third dimension can be calculated.[9]

7.5. Orthophoto Module

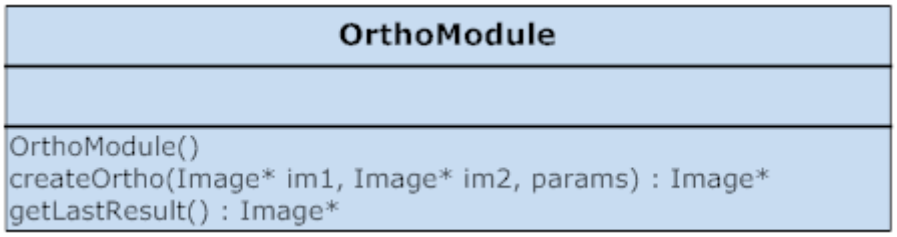

Figure 7 - 5 : Class Diagram of OrthoModule class.

This class acts like a wrapper around the functionality. It does not store any specific data. It takes an image and a DEM as input to generate the output. First, input image is projected onto the input DEM using the camera calibration parameters of the image. If no DEM is specified as input, planar surface assumption can be used to continue the process. After this step, the projected image is captured at a right angle. The output image can be considered equivalent to a map. This process is called Orthorectification.

7.6. Mosaic Module

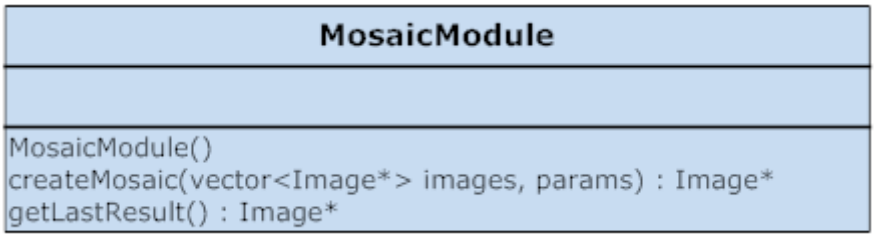

Figure 7 - 6 : Class Diagram of MosaicModule class.

Like the DEM and orthophoto modules, this module consists of a class that wraps the mosaic creation subroutine. The class contains a createMosaic method which implements the mosaic creation algorithms. The method takes a vector of images and a number of other parameters as its arguments and returns a pointer to the constructed mosaic image. Some of these parameters are internal in the sense that they are used by the project manager module to create an effective communication channel between the modules. In the mosaic module's case, one of these parameters will be reserved for the homography matrix, which is a data structure managed by the registration module by default. This matrix is required by the mosaic module to perform its computations; hence the matrix is communicated to the mosaic module by the project manager as a parameter to the createMosaic method. Rests of the parameters of the createMosaic method are optional in the sense that they are used to carry user preferences and other relevant optional data.

7.7. Registration Module

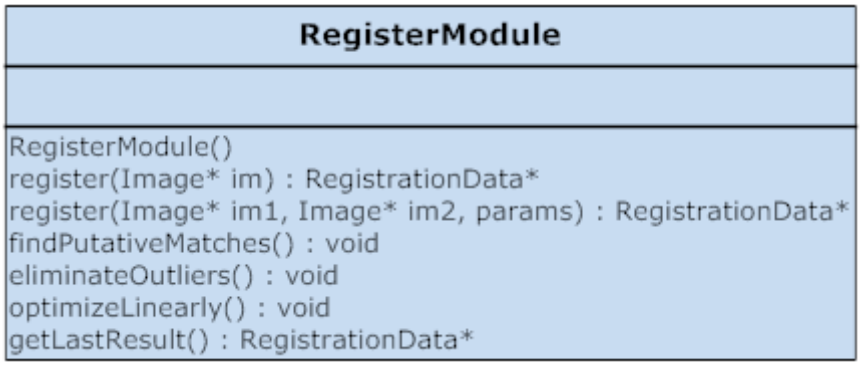

Figure 7 - 7 : Class Diagram of the RegisterModule class.

This class has two functions: first one is to register an image to world coordinates, and the second is to register two images with respect to each other. Registration data is needed for most of the other functionalities, therefore most data will pass through this class at least once.

Registration between two images is done in this module via following these steps below:

First interest points of both images are calculated. For this task Harris corner detection algorithm is utilized. Second, putative matches among these interest points are calculated using the cross correlation formula. The pairs found in this step are not reliable enough, therefore the next step is eliminating outliers. In this step, the main purpose is to eliminate pairs that are not following a general pattern of motion. Using the putative pairs found in the last step, multiple trial homography matrices between two images can be calculated. Then the most successful of these matrices are taken, which is the one that most number of putative pairs validate. Using this homography matrix, the outliers (unsuccessful pairs) are eliminated. This process is called registration of two images among each other and it gives the spatial relation between images.

8. SPECIFICATIONS AND DEPENDENCIES

8.1. Software and Environment Specifications and Dependencies

The primary targeted software platform is GNU/Linux (kernel 2.6). Although the design is made to ensure that the software will be cross-platform in source format, other development phases (i.e. testing) will concentrate on the GNU/Linux port of the Pix'r'us Photogrammetry Suite. A Windows (XP/Vista) port will also be available; however its support is of less priority.

Pix'r'us Photogrammetry Suite utilizes various external libraries to perform a variety of image processing tasks. Thus, the software requires appropriate ports of these libraries to be available in the software platform. GNU/Linux and Windows ports of these libraries are available and supported actively.

8.2. Hardware Specifications and Dependencies

The end product is required to run on contemporary x86/x64 platforms and will support only these hardware platforms by default. However, as C++ is used as the development language, the software will retain its portability (in the source code format) among the platforms where ISO C++ is supported. Thus, as long as external libraries are available on a platform, the software can be recompiled to obtain a compatible executable that runs in the desired platform.

By default, the software launches a GUI and performs the required computations in an interactive manner: i.e. it asks the user to supply various parameters for fine tuning purposes. Thus, the user also needs an adequate display equipment to be able to run the software in its default interactive mode.

8.3. Other 3rd Party Software Involved

Apart from three external libraries, Pix'r'us Photogrammetry Suite does not incorporate any 3rd party software. The mentioned external libraries are all free software (i.e. open source) and are licensed with free software licenses that are GNU General Public License (GPL) compatible. These libraries are:

* Open Computer Vision Library (OpenCV)
* wxWidgets GUI Toolkit (wxWidgets)
* Geospatial Data Abstraction Library (GDAL)

OpenCV contains robust implementations of various image processing algorithms (feature detection, homography matrix calculation etc.) used by the software. wxWidgets is used to construct the GUI. GDAL is used in the input/output modules when reading/writing image files.

9. REFERENCES

1. Wilson, J.P.; Gallant, J.C. (2000). "Chapter 1", in Wilson, J.P., and Gallant, J.C. (Eds.): Terrain Analysis: Principles and Applications, 1–27. ISBN 0471321885.
2. Bolstad, P., (2005), GIS Fundamentals: A First Text on Geographic Information Systems, Eider Press, White Bear Lake, MN, 2nd ed.
3. Petrie, G., (1977), Transactions of the Institute of British Geographers: Orthophotomaps New Series, vol. 2, no.1, Contemporary Cartography. , pg. 49-70
4. New methods for dynamic mosaicking, Nicolas, H. *Image Processing, IEEE Transactionson*
   On page(s): 1239-1251, Volume: 10, Issue: 8, Aug 200Fernandez, E., Garfinkel, R. & Roman Arbiol, (May-June, 1998) Operations Research, Vol. 46
5. Real-time scene stabilization and mosaic construction (2002), Hansen, M. Anandan, P. Dana, K. van der Wal, G. Burt, P. Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop
6. G.T. Clement, J. Huttunen, and K. Hynynen, "Superresolution ultrasound imaging using back-projected reconstruction" Journal of the Acoustical Society of America, Volume 118, Issue 6, pp. 3953-3960, 2005.
7. Brigham, E. Oran (1988). The fast Fourier transform and its applications. Englewood Cliffs, N.J.: Prentice Hall.
8. C. Harris and M. Stephens (1988). "A combined corner and edge detector". Proceedings of the 4th Alvey Vision Conference: pages 147—151
9. Trucco T. , Verri A. (1998). Introductory Techniques for 3D Computer Vision. New York: Prentice Hall.