SENIOR PROJECT
FALL 2007
FINAL DESIGN REPORT

MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Okan AKALIN e1448273

Görkem KURT e1395250

Kazım Işık e1448760

Yunus Olgun e1448992

Table of Contents

1. Introduction

Till the design phase, group's interest was focused on determining the requirements. In determining the requirements, a literature survey had been performed, market research had been done and the constraints are fixated. In determination of the needs of the project, the team had a better understanding of the necessities, phases, and the operational aspects of the project. This information helped the team to start the designing process at an initial level. Revising the initial design of the project, and working on the prototype leaded the way to make the final design.

Until now, before the initial design phase of the project, the interest was focused on determining the requirements. To specify the requirements, a literature survey had been performed, market research had been done and constraints were fixated. By determining the needs of the project, the team had a better understanding the necessities, phases, and the operational aspects of the project. In the light of the information obtained from requirements, the initial design of the project has been completed.

The initial ideas on the design have developed into mature visions about the project.

1.1 Project Title

The title of the project is Photokan. The title is a derivation from the company name.

1.2 Problem Definition

The advent of the computing technology and the research in the computer vision and image processing area opened up many possibilities for softcopy photogrammetric tools in which geometrical properties of objects are examined. These tools attract a well deserved attention since without the software assist, gathering information on geographical properties, such as terrain analysis, land survey becomes a heavy chore. Provided with online data, these tools can supply crucial information efficiently in case of emergencies. A disaster such as an earthquake is a good example of this. A quick analysis of the disaster's effects can speed up the life saving operations. Another area of application is the military tactical operations in which terrain topology forms the basis of the rationale. A before hand knowledge of the surface directly affects the decisions, such as the determination of necessary vehicles. The scope of softcopy photogrammetry actually covers a wider range of areas than mentioned, ranging from police investigation to archeology; anywhere geometric properties of objects are useful. The application of these tools in critical issues (such as generating data concerning national security) makes national companies the first choice in obtaining licenses so even though a softcopy photogrammetry tool doesn't have a direct impact on human life, the interest in new tools don't cease. In Turkey, there is also a lack of software in this area. Another issue is the lack of flexible, user friendly software for linux platform even though some open source alternatives exist.

1.3 Project Scope

Photokan will be a softcopy photogrammetry tool which will also employ geographical information processing facilities. The basic input target of Photokan is aerial images but it will be possible to use it on other types of images if deemed necessary. This will be a result of the flexibility of the program.

Basically, Photokan will process a set of aerial images with given camera parameters. The aim is using these images to create a model of the surface covered, and combining them to create one big picture which have the properties of a map. This can be achieved by a pipeline model. The process will work its way sequentially but it will also be possible to directly jump to any stage since all phases are independent from another. The processing modules are:

• Image Registration: Image registration is performed to detect the overlapping areas of a given set of images. This process will also estimate the transformation relation between images. This data will be used in determining the missing camera parameters in case one is missing this data. Also the overlapped region will be combined in mosaicking process.

• Digital Elevation Model Creation: A digital elevation model (DEM) provides the height map of the surface. Digital elevation models will be created using multiple images of the scene. There will also be a 3-d visualization of the DEM.

• Orthorectification: This process will shift the perspective of the image so that It gains the basic property of a map.

• Mosaicking: Mosaicking yields the overall image which combines the overlapping images. All of these methods are prone to errors; this is a result of the problem. The problems are ill posed and the criterion of quality is mostly subjective. We will provide manual correction mechanisms in the methods to overcome the difficulty of reaching results which will be satisfactory.

The images that Photokan will work on are aerial images which possess geographical information. When a basic set of parameters are supplied, it will be possible compute the world coordinates under the mouse. The basic set of parameters will be provided in World file format. It will be possible to convert between various coordinate systems.

It will be possible to work on many file formats and convert between them. Photokan will also employ basic image processing and manipulation methods so that preprocessing, enhancing, restoration will be possible by operator intervention. The basic capabilities are zooming, changing contrast and brightness values, rotating, sharpening, smoothing, contrast stretching, histogram equalization, and convolution by manually entered filters. It will be possible to work on a selected area of the image.

The scope of the project extends to bringing together qualified algorithms under a flexible interface, creating the necessary space to the user. In order to achieve these goals, fine tuning of algorithms will be as much as possible.

1.4 DESIGN OBJECTIVES

The design of Photokan is based on 3 objectives. objectives.

1.4.1 Usability

Photokan is designed to be a flexible program, it has consecutive main steps in execution order but user can jump to any phase if the necessary inputs are provided, so it has to be easily understandable. A GUI

will be provided to the user and it is planned to be a simple, yet it can comprise all the functionalities of the program.

Another important fact is that, in image processing the methods are too generic and there is not one solution to all of the problems. In order to produce a usable photogrammetry tool, viable alternatives to problems should be provided to the user.

1.4.2 Efficiency:

It is a certain case that in image processing there is a high computational load. So the algorithms used in the project must be chosen and implemented carefully. The image files on which Photokan will execute on take huge amounts of storage. Both the space used and the time consumed has to be minimized in order to run Photokan efficiently on a regular Personal Computer.

1.4.3 Reliability:

In order to make Photokan reliable enough, the design mustensure that the data on which Photokan executes on has to be stored carefully, because there can be cases in which reverting to the beginning is impossible. Also the files Photokan works on are huge in size; the inevitable high computational load should not be repeated because of the program, its execution must be reliable.

2. Current State of the Project

The current state of the project had been prescheduled with the assignments given by Milsoft. Image registration which is an essential part in all other processes has been completed. Mosaicking has also been implemented. The details of the algorithms were provided by Milsoft. Since surveying is also an important part of the project, it needs to be treated in its own right.

2.1 Survey Topics

Since the project consists of sequential phases, survey on topics which will be used earlier, has been considered first. Image registration and stereopsis has been surveyed during this process and the problem has been classified. A detailed review on image registration and links to valuable sources have been provided by Milsoft.

A wiki has been setup in order to facilitate communication between team members themselves, and between Milsoft and the team. It also serves as a way of supervising and verifying the progress of the project by the supervisors.

2.2 Implementation

The implementation is moving according to a predefined plan. An assignment has been given to perform basic image viewing and coordinate transformation (georegistration) tasks. During the implementation of this assignment, the team decided on the libraries which will be adapted and a convention has been setup. A second assignment on image registration and image mosaicking has followed the first assignment. In its current state, the project is capable of opening and loading of all major image file formats and the geographical image file format variants. The associated world files of images can be parsed and loaded. An affine transformation is obtained from this world file, and real coordinates of

image pixels are computed and shown accordingly for image frames. Image sets can be registered and can be combined into mosaicks. Also project serialization is also partly implemented. A project can be deserialized from XML structure.

Ideas on graphical user design have also developed. The program consists of a workspace which provided basic tools and image viewers. Frames can be loaded under a tab manager, a tree control is available to provide the structure of the project in an accessible format. The team decided to adapt the Model-Viewer-Presenter pattern, separating the viewing logic from the data. Another decision was to adapt the wrapper pattern for image processing capabilities since many alternatives exist that can be replaced if deemed necessary.

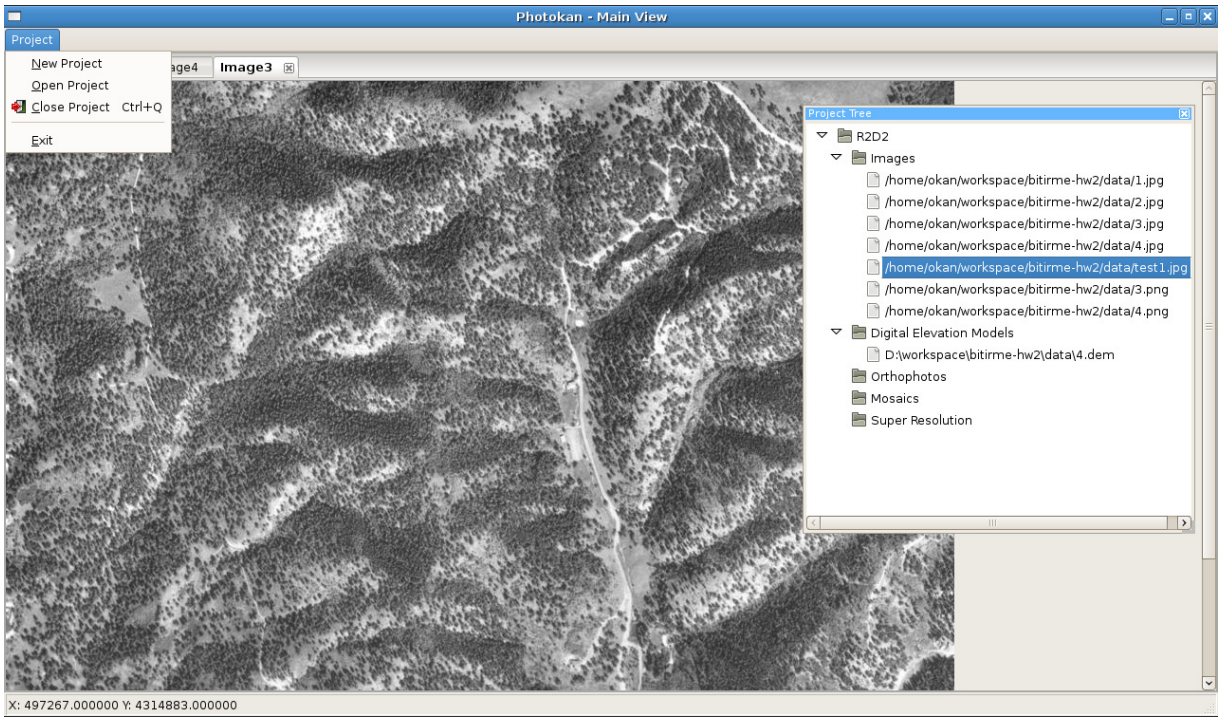

Figure.1The main view of Photokan.

Photokan consists of dockable user interface elements as can be observed on figure.1. This allows a flexible user interface. The project consists of a set of folders which are present in the tree controller. The user can traverse the project with this tree list. In order to process a set of elements, the user can select multiple elements from the tree and can apply a process on them using the popup menu. The georegistration values can be observed on the statusbar. Another important point is the tab manager which allows to handle multiple images concurrently by the user.

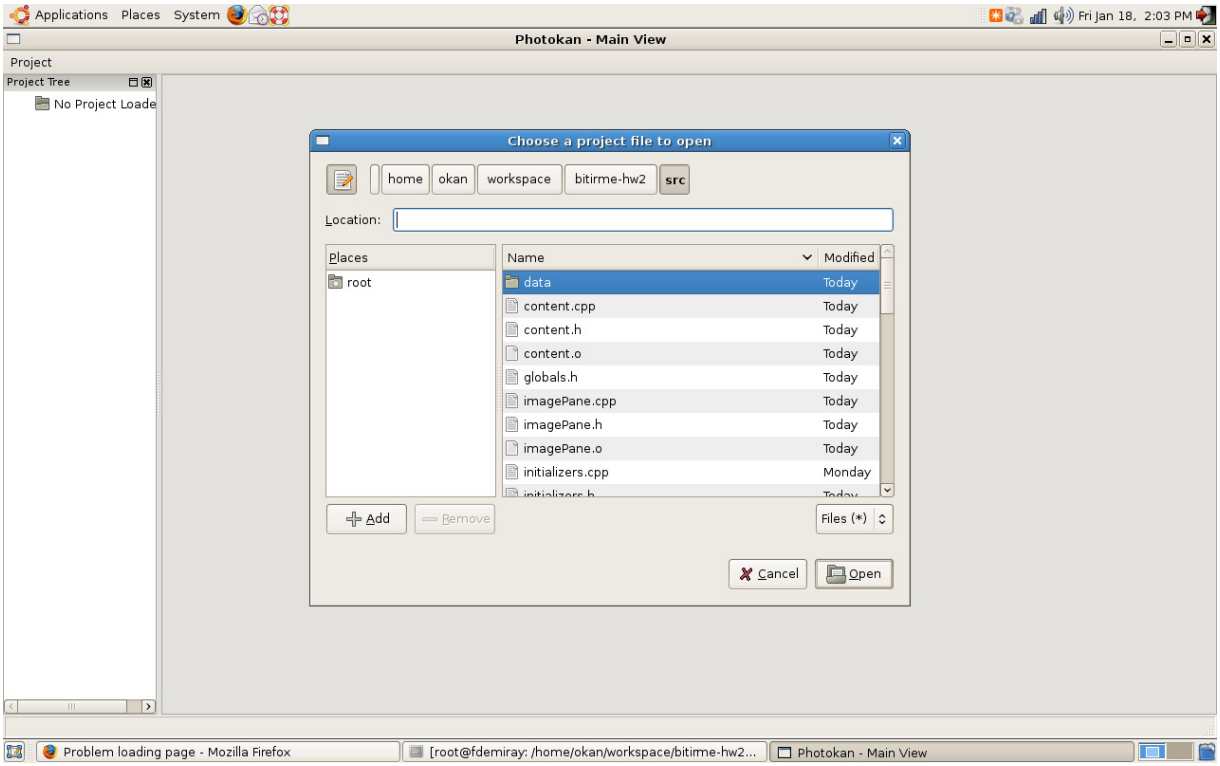

Figure.2.2 Open project dialog box

An evaluation of Photokan's mosaicking can be performed with the following figures.

Figure.2.3 Result of registration of two images. Overlapping area is shaded with blue. The circles show the matching points. Red points are interesting points.

Figure.2.4 Result of registration of two images. Overlapping area is shaded with blue. The circles show the matching points. Red points are interesting points.

Figure.2.5 Result of final mosaicking of two images.

Figure.2.5 Result of mosaicking of frames of 4 seconds video. 4 frames were used in mosaicking.

3. Data Design

The design of data in Photokan consists of both internal and external structures.

3.1 Internal Structures

The Photokan handles data of huge amounts. 3 types of data are output in the course of processing: digital elevation models, orthophotos and mosaics, all of which are essentially 2 dimensional matrixes, which all boil down to the usual images. Images are associated with cameras , camera parameters include the camera's internal parameters which affect the quality of the overall image and the external parameters which provide the information on spatial coordinates of the scene (from which georegistration can be performed). Figure.3 provides an entity relationship diagram describing the internal structures.

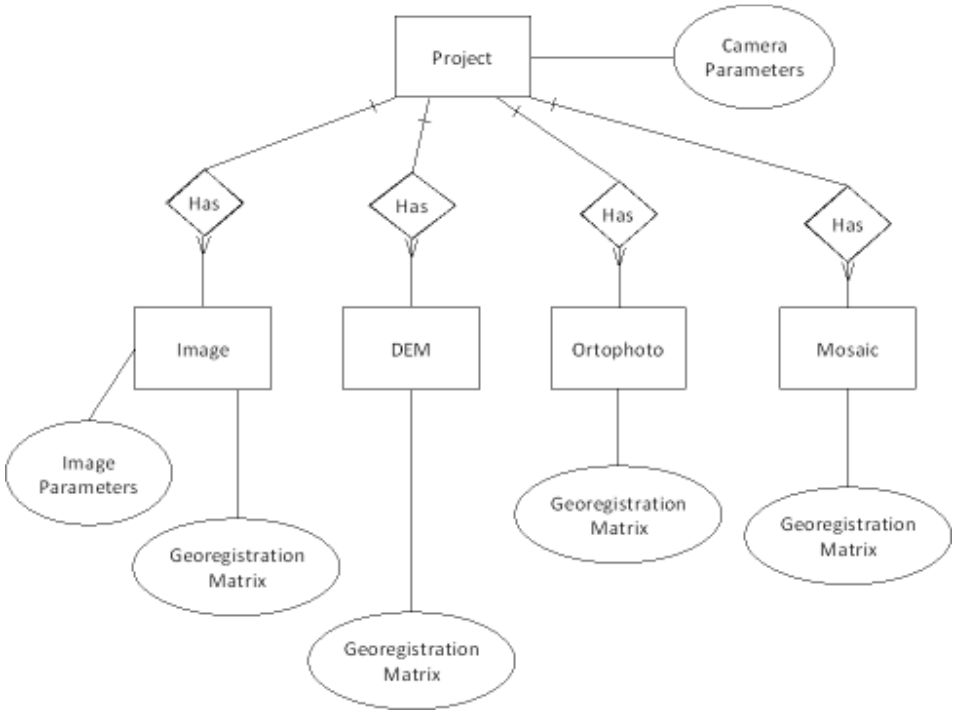

Figure.3 ER Diagram for internal structures

3. 2 External File Structures

There are three different file structures which will be created by PHOTOKAN project. The main purpose of these file structures are the serialization of the internal data.

3.2.1 - '.prj' Files

This file structure is based on XML and it will be used for "Project" class. When user is creating a new project this file will be created along the internal project data and all changes will be inserted at the time when there is a change. To open an existing project this file will be used. Main scheme of this file structure can be seen at below.

```
<project>
    <name> Project1 </name>
    <workingPath> /where/is/the/working/path </workingPath>
    <images>
        <GISObjectPath> GISObjectPath1 </GISObjectPath>
        <GISObjectPath> GISObjectPath2 </GISObjectPath>
        <GISObjectPath> GISObjectPath3 </GISObjectPath>
    </images>
    <dems>
```

```
                <GISObjectPath> GISObjectPath4 </GISObjectPath>
                <GISObjectPath> GISObjectPath5 </GISObjectPath>
                <GISObjectPath> GISObjectPath6 </GISObjectPath>
        </dems>
        <ortophotos>
                <GISObjectPath> GISObjectPath7 </GISObjectPath>
                <GISObjectPath> GISObjectPath8 </GISObjectPath>
        </ortophotos>
        <mosaics>
                <GISObjectPath> GISObjectPath9 </GISObjectPath>
        </mosaics>
</project>
```

As seen '.prj' files are self describing.

3.2.2- '.cob' Files

This file structure is based on XML and it will be used for "CameraObject" class. When user is creating a new camera this file will be created along the internal CameraObject data. When there is a reference to this '.cob' file this file will be used.

```
<cameraObject>
        <name> CameraObject1 </name>
        <parameters>
                <f> fVal </f>
                <pixelSize> pixelSizeVal </pixelSize>
                <principlePointX> principlePointXVal </principlePointX>
                <principlePointY> principlePointYVal </principlePointY>
        </parameters>
</cameraObject>
```

Parameters are the internal parameters of the camera.

3.2.3- '.gob' Files

This file structure is based on XML and it will be used for "GISObject" class. When user is creating a new GISObject this file will be created along the internal GISObejct data and all changes will be inserted at

the time when there is a change. To open an existing GICObejct this file will be used. Main scheme of this file structure can be seen at below.

```
<GISObject>
        <name> GISObejct1 </name>
        <sources>
                <GISObjectPath> GISObjectPath1 </GISObjectPath>
                <GISObjectPath> GISObjectPath2 </GISObjectPath>
                <GISObjectPath> GISObjectPath3 </GISObjectPath>

        </sources>
        <products>
                <GISObjectPath> GISObjectPath1 </GISObjectPath>
                <GISObjectPath> GISObjectPath2 </GISObjectPath>
        </products>
        <baseImagePath> baseImage </baseImage>
        <parameters type = "GISImage">
                <param> param1 </param1>
                <param> param2 </param1>
                <param> param3 </param1>
        </parameters>
</GISObject>
```

Sources are the source GISObjects which are used for the creation of this GISObject and products are the GISObjects which created from this GISObject. Parameters field contains the type of the image. It can be "GISImage" , "GISDEM" , "GISOrthophoto" , "GISMosaic". According to this attribute down casting of the GISObject can be made at when reading. Also other parameters will change according to this attribute. For instance, a GISImage will contain the external parameters of the camera but a GISDEM cannot hold this parameters.

4. Architectural Design

In Photokan, the modules are totally independent of each other even though they make use of each other's products as can be observed from figure.4. This split allows modeling them without interfering the other module's internals. Photokan is partitioned into 5 modules: Project manager, digital elevation model extractor, orthophoto generator, mosaic maker, image processor. Project manager module checks the dependencies, drives the other modules and harnesses the serialization facilities of project and the contained objects.

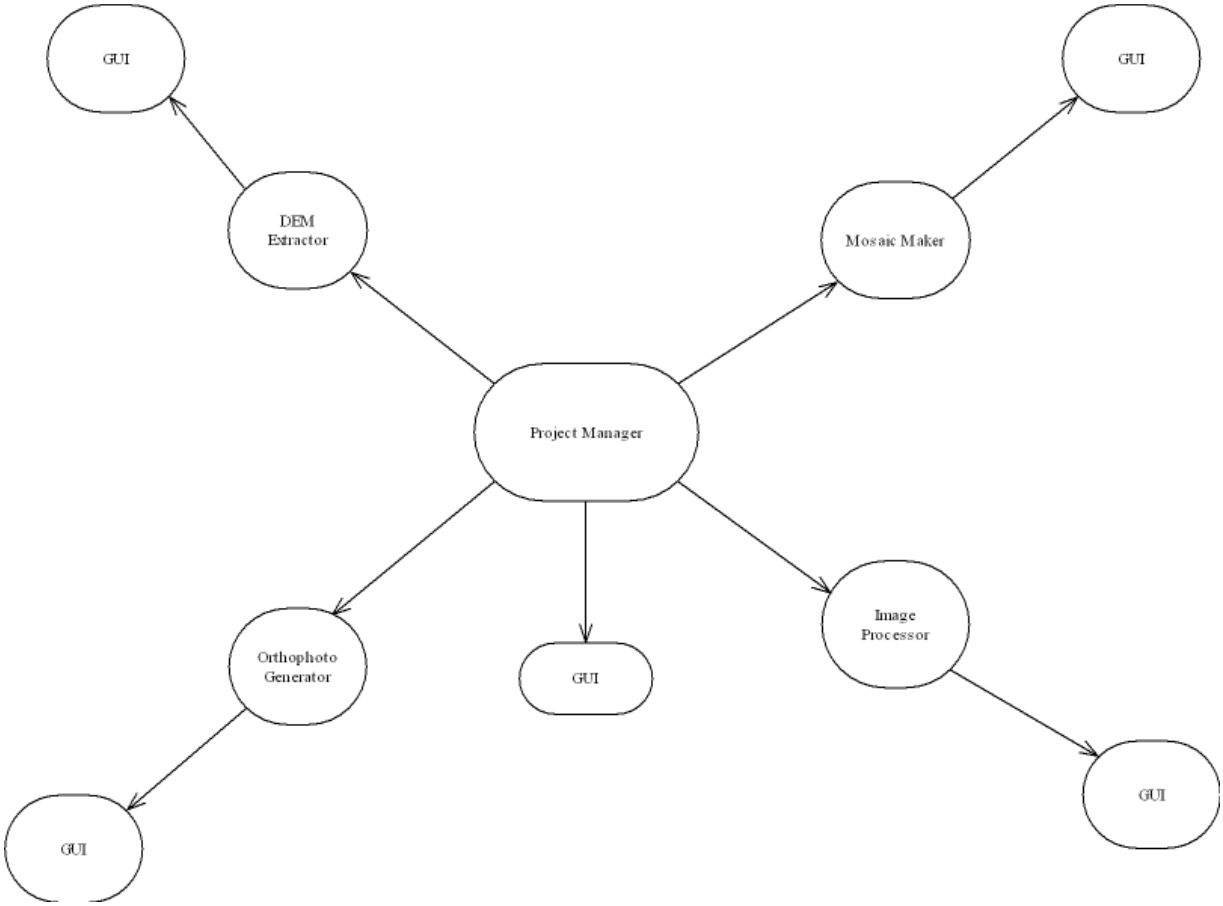

Figure.4

The project data of Photokan is complex, and the processes that this data will go through are also complex so separating the viewer from the data will certainly ease maintaining of the project. Also, as can be observed from the figure, a graphical user interface is attached to every module which behaves as the viewer.

The modules except for the project manager are all a product of the problem definition. The aim of the project is creating a photogrammetry tool that will process given images and will obtain a mosaicked, orthorectified big picture along with its elevation model. The fact that a sequential flow of processes is

present was harnessed to reduce the overall complexity and to obtain interchangeability of modules if deemed necessary.

Another important point is the database of the Photokan. The project manager module houses the project data which holds the objects belonging to the project scope. Every other module that will work on a dataset obtains its input from the project manager. This centralized management also avoids clashes.

5. Design of Program Interfaces

As mentioned before, Photokan provides an interface for every one of its modules. These interfaces are graphical and provide the capability of driving the module and viewing its outputs. Use cases and activity diagrams are provided to make the intent clearer,

5.1 Use Cases

The use cases give an overview of the usage of Photokan. Use cases for project manager, image processor module, digital elevation extractor module, orthophoto generator module, mosaic maker module.

i. Project Manager

Project manager provides the project manipulation, module driving, dependency checking functionality. The figure is split into four parts since it is too big.

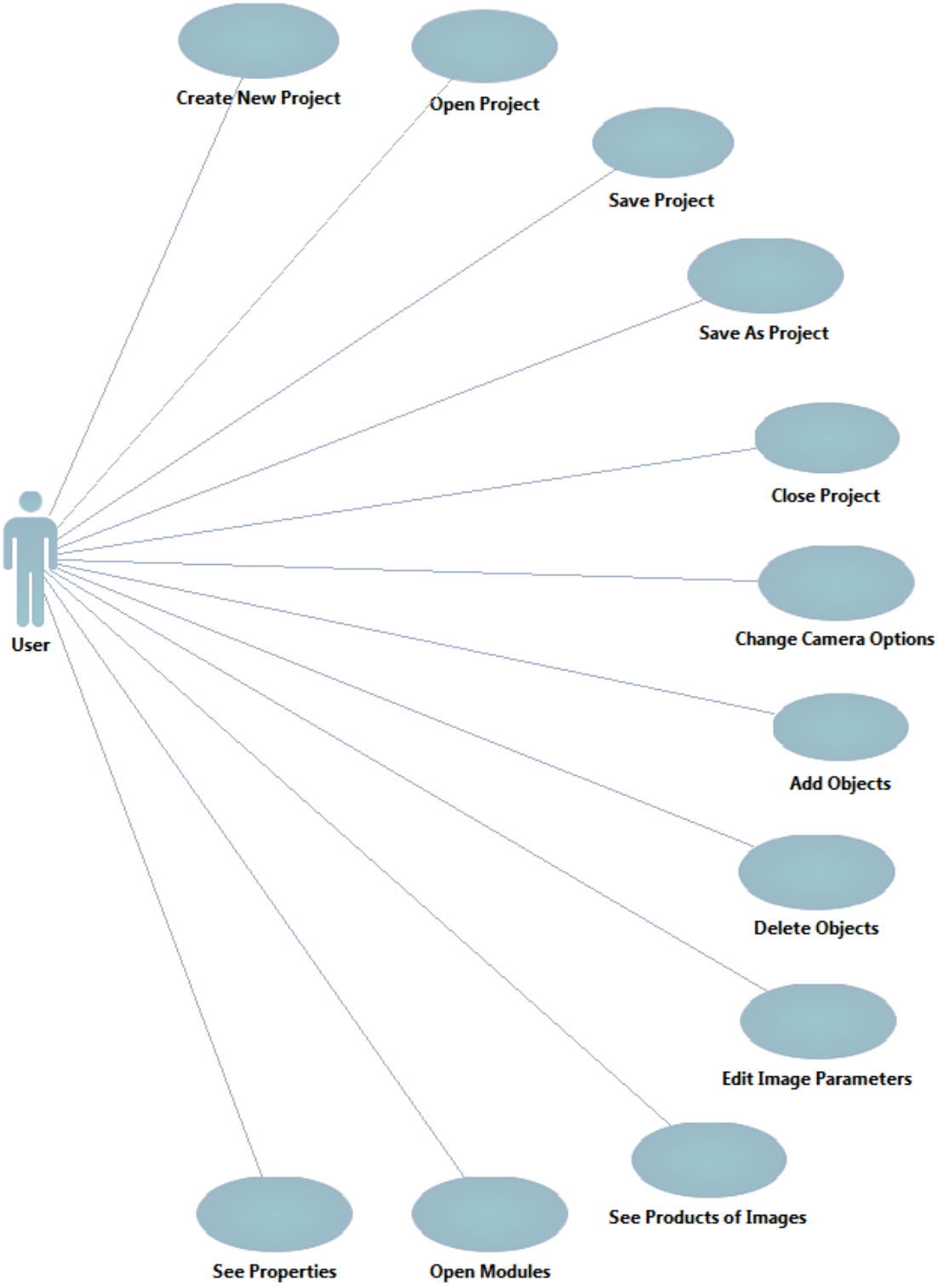

Figure.5.1 Project Manager use case

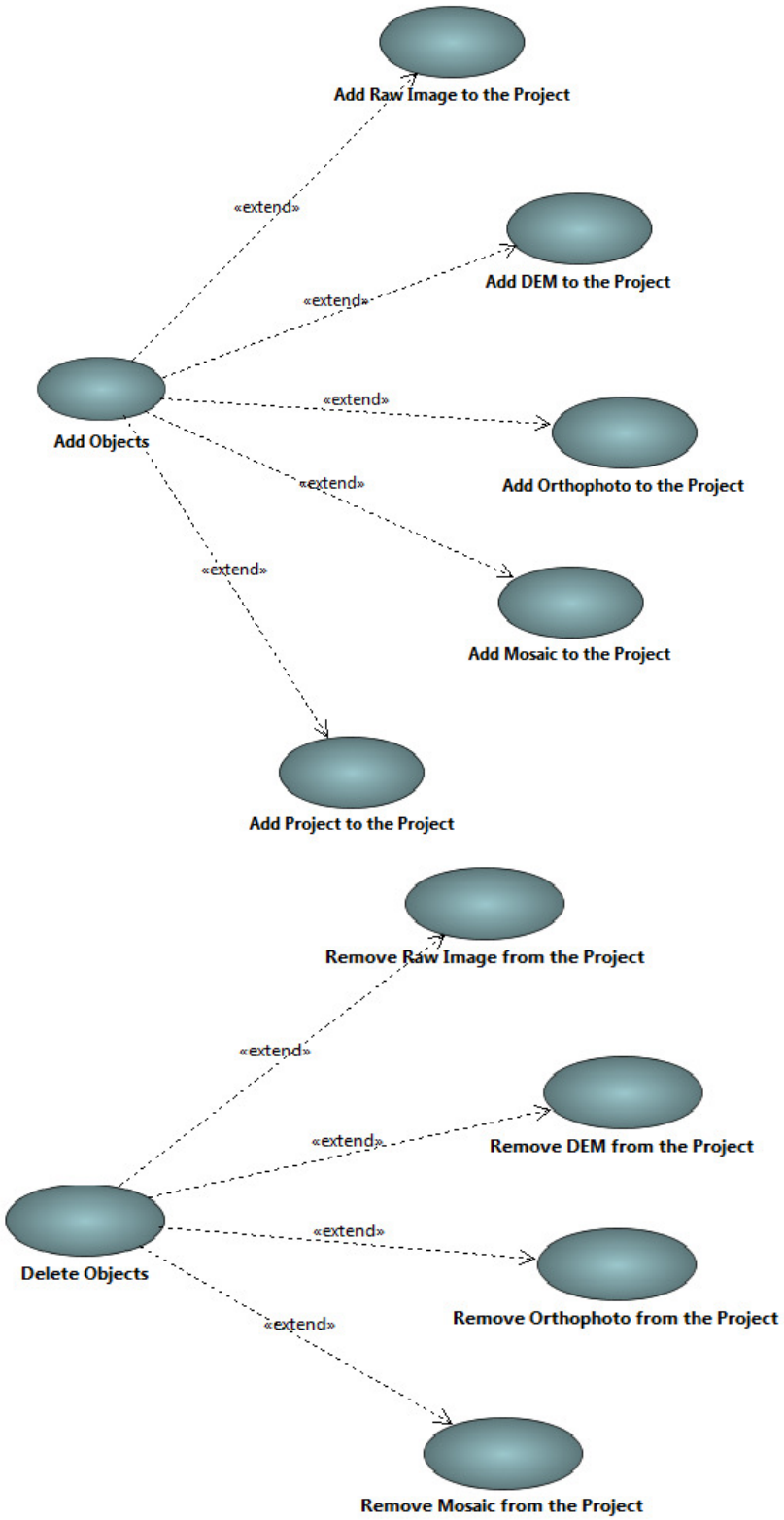

Figure.5.2 (cont.)

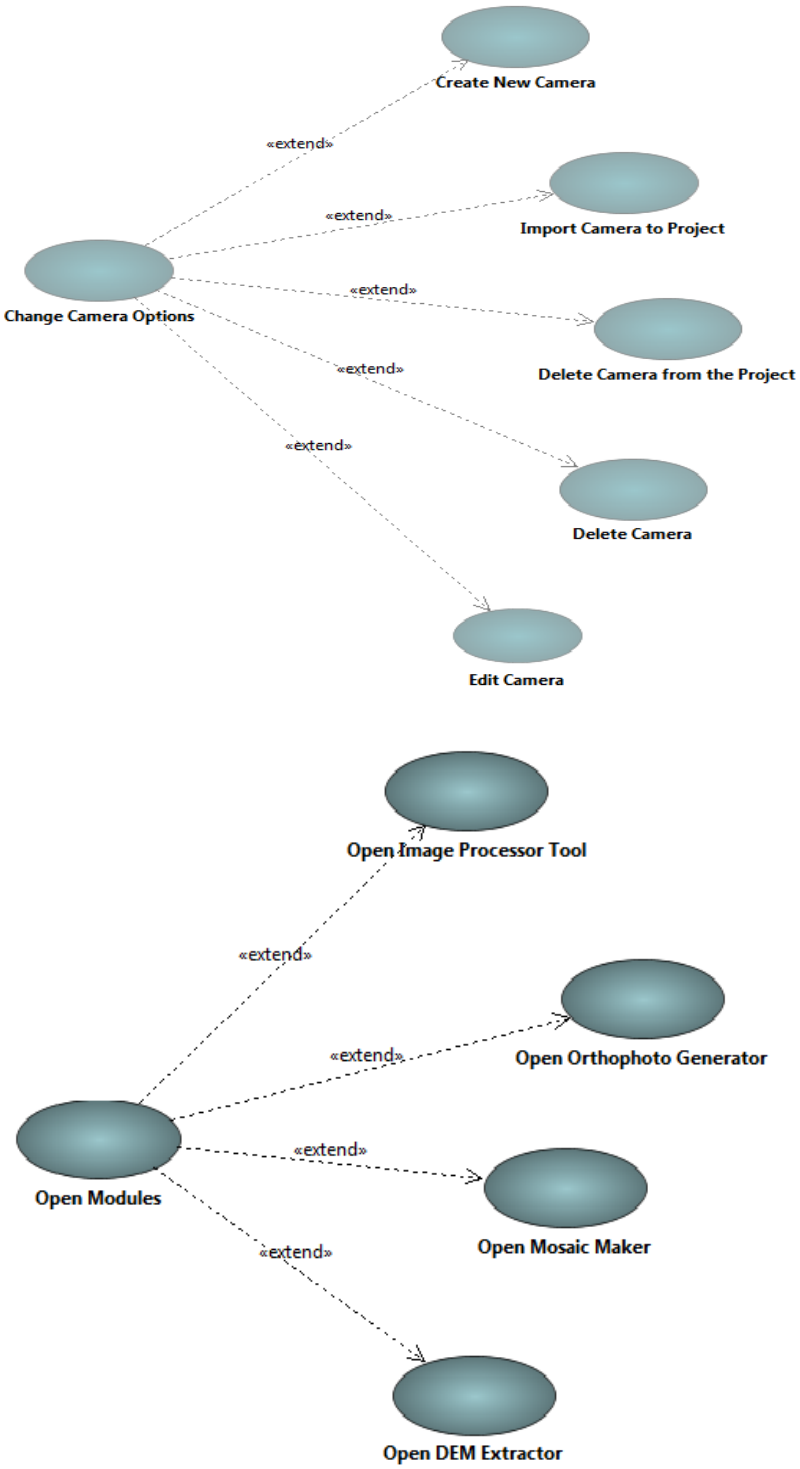

Figure.5.3 (cont.)

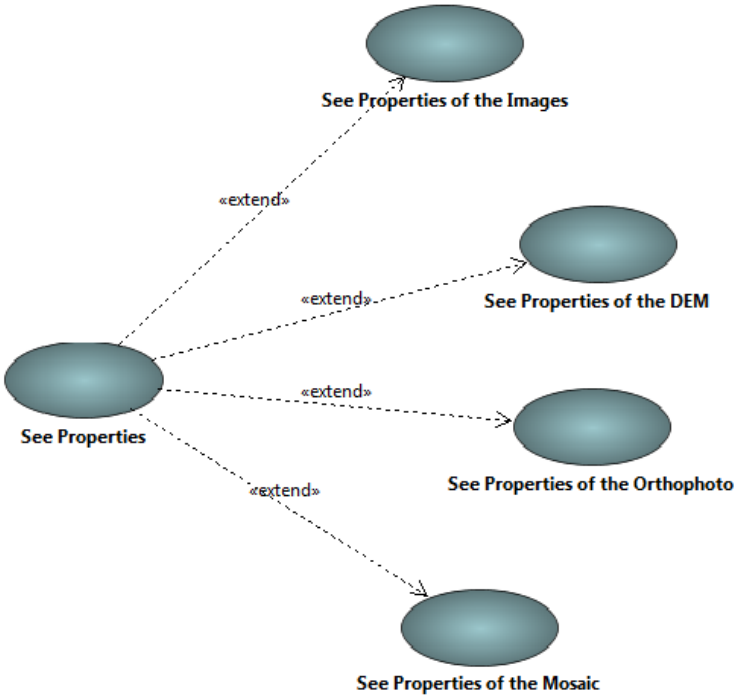

Figure.5.4 (cont.)

ii. DEM Extractor

Dem Extractor module extracts the digital elevation model.

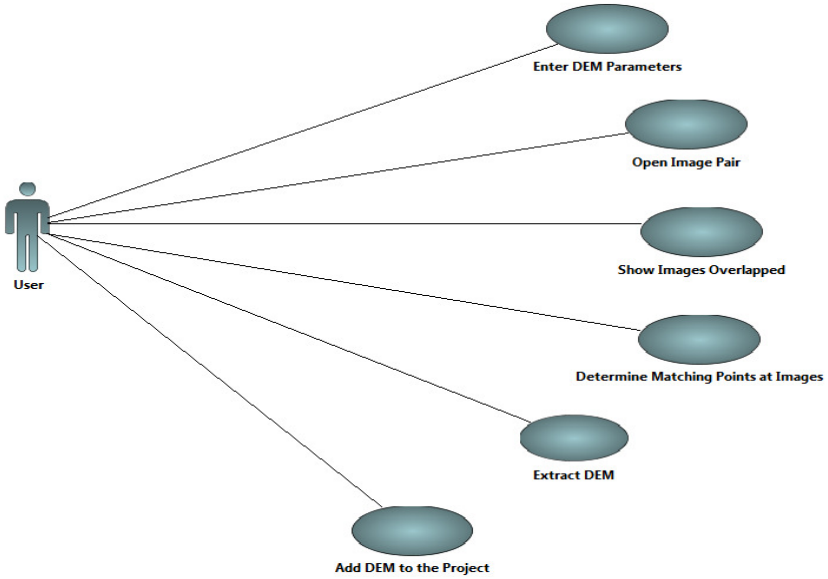

Figure.5.5 DEM Extractor module use case

iii. DEM Displayer

DEM displayer module provides the option of viewing digital elevation models from different viewers.

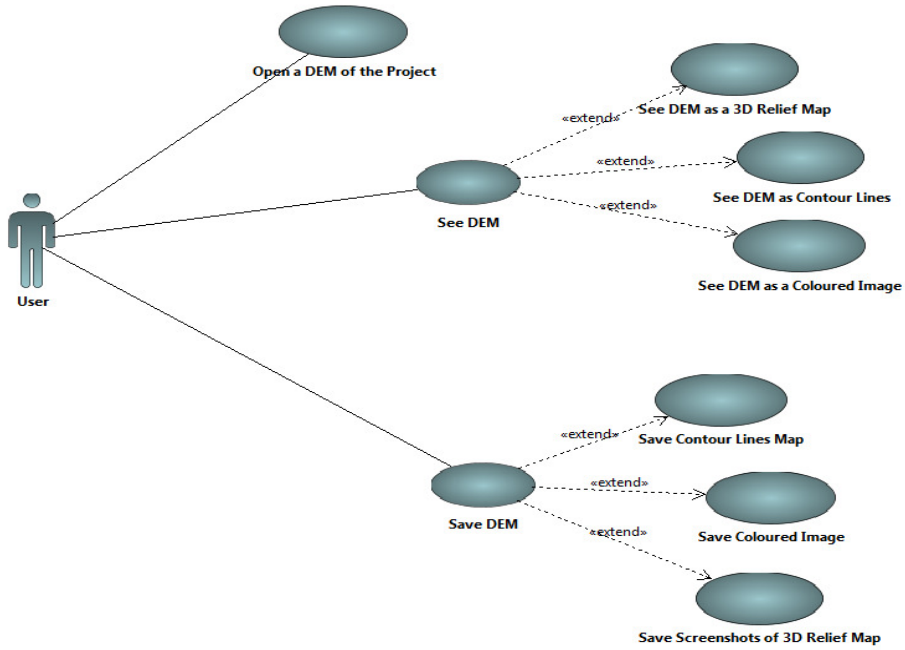

Figure.5.6 DEM Displayer

iv. Mosaic Maker

Mosaic maker module provides the mosaic making capability given two images.

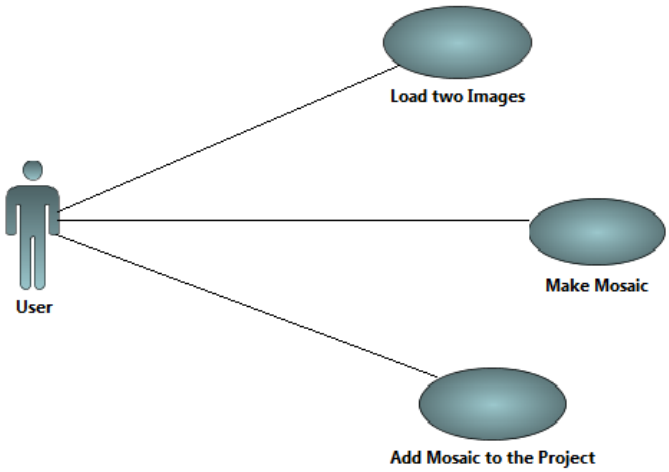

Figure.5.7 Mosaic maker module

v. Orthophoto Generator

Orthophoto generator module produces an orthorectified version of an image.

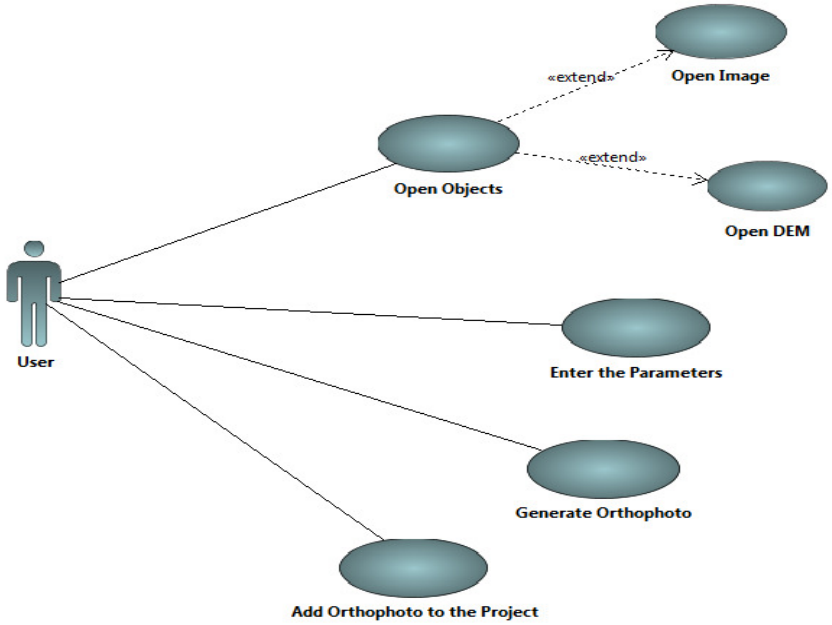

Figure.5.8 Orthophoto generator module

5.2 Activity Diagrams

Diagrams for major activities are provided.

5.2.1 Project Creation

Project creation scenario is considered.

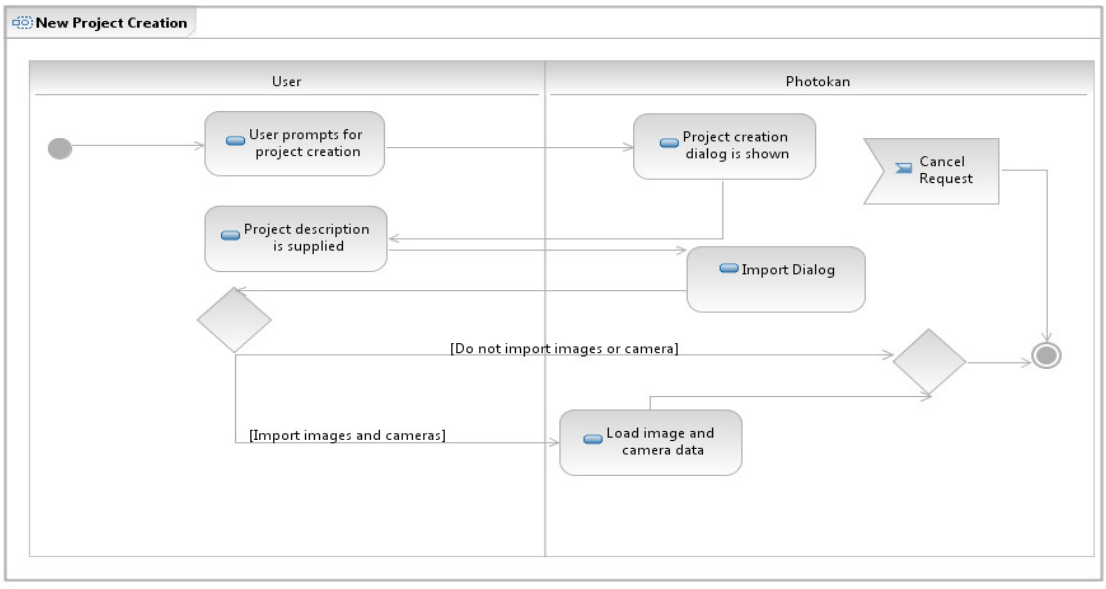

Figure.5.9 Project creation activity diagram

5.2.2 DEM Extraction

In DEM extraction the user may intervene to the procedure in order to make changes to the results of image registration.

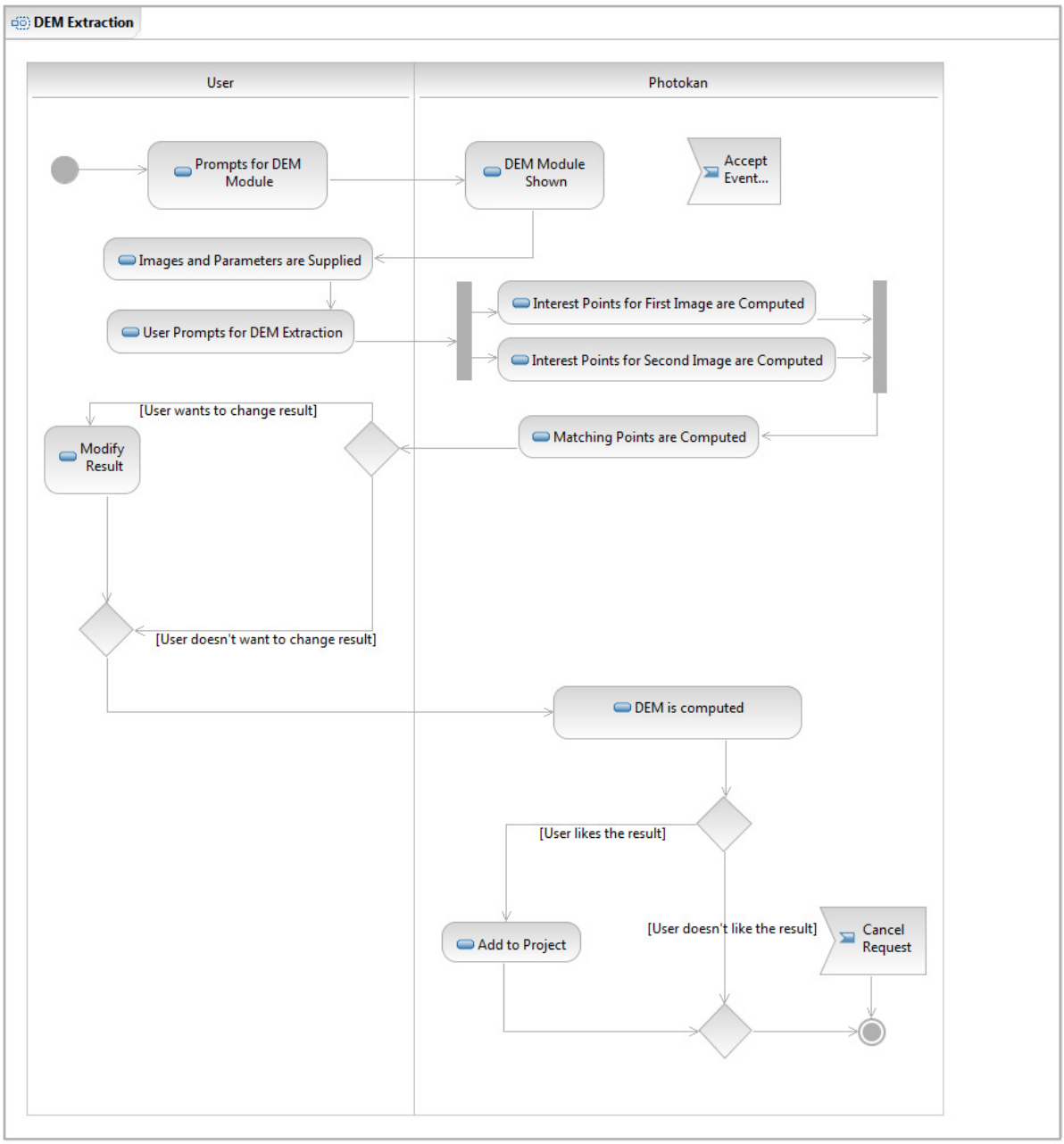

Figure.5.10 DEM Extraction

5.2.3 Orthophoto Generation

During orthophoto generation, the digital elevation model is needed. If the elevation model is not currently generated, the user is directed to this phase first, otherwise the model is retrieved from the database.

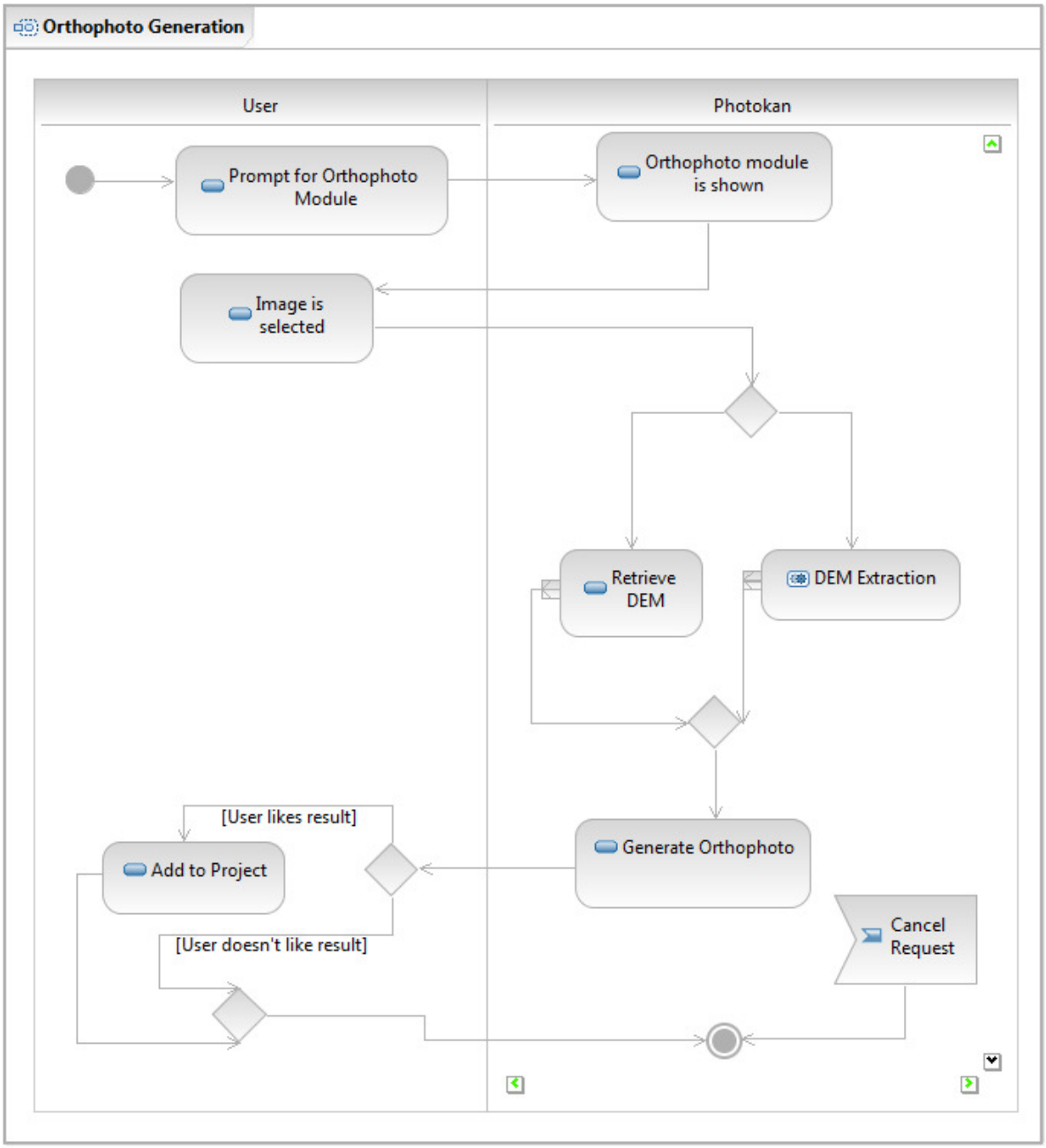

Figure.5.11 Orthophoto Generation

5.2.4 Help retrieval

Documenting the software is one of the most important aspects of software development, Photokan provides indexed help and search capability to ease the burden on users.

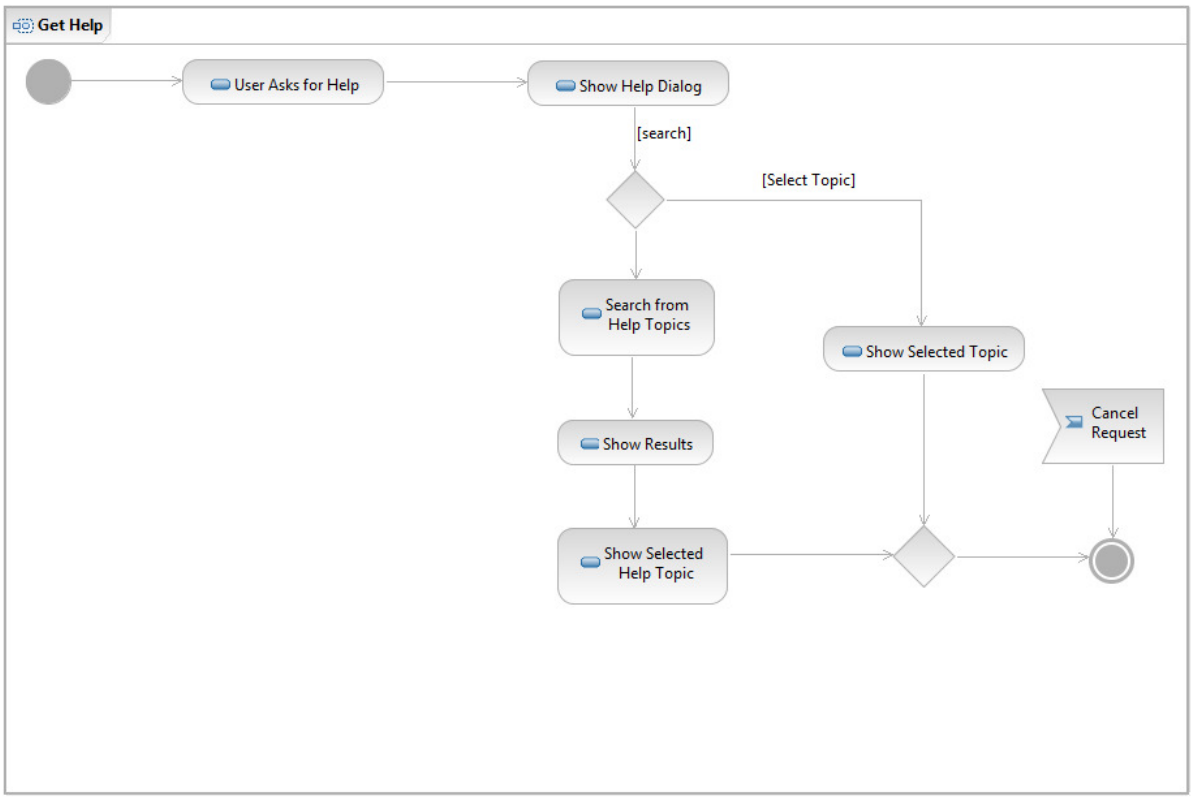

Figure.5.11

6. Component Design Descriptions

Basically, Photokan is divided into 6 modules. The project manager module among these, drives the other modules. During the startup an instance of it is created, the lifetime of the project manager spans the program execution time. It must be noted that the user interface of project manager lies in a separate class. This also holds for every other module. The user interfaces drive the modules by events produced by the user which result in calls to corresponding methods. The trigger* methods in dialog and window classes basically correspond to the fore mentioned feature and every module has a counterpart of this methods which are named without the trigger prefix. The ProjectManager class contains one instance of Project class (it manages only one project at a time). It can open every other module and every request for project manipulation is directed over project manager.

6.1 ProjectManager Module

This class is the core of ProjectManager module. An instance of this class will be generated when Photokan starts and it will exist till the end of the program. Actually this class is a simple controller. It controls the project related data. All graphical user interface requests will be sent to this class and it will call appropriate procedures. Besides that, it is the portal of other modules. For opening another module

user should send a request to this class. At this module, a use of model-view-controller design pattern can be seen.
As data, it contains only an instance of the 'Project' class. It is a method container more than data because it is a controller at all.
As functions; it contains project methods, camera methods and run methods of other classes. The GUI of this module will be at interaction with this class only. All methods of them will actually call another method simply.

6.1.1 Project Class

This class contains all data of the project. Other modules will use this class to see the existing 'GISObject' instances. Every image, DEM, orthophoto or mosaic should be a member of a project for processing.
As data, it contains the name of the project, the working path, a default camera and pointers of other GISObjects. As seen, there are four containers. Each one of them is for another type of GISObject. First one is for images, second one is for DEMs, third one is for orthophotos and last one is for mosaics.
Other modules will use this class's methods to access or modify GISObjects. 'serialization' method of this class will create a '.prj' file.

6.1.2 GISObject Class

This class is base class of all GISObjects like image, DEM, orthophoto or mosaic. Actually all this types have some common properties; therefore we used this inheritance schema. Firstly all of them is a simple image. Even DEM is a simple image with 1-band rather than 3-band like other ones. Secondly all of them can be source of other GISObjects or products of other GISObjects. For instance a DEM is a product of two images. An orthophoto is a product of DEM and an image. A mosaic can be a product of more than one image or orthophoto and their corresponding DEMs.
When the bitmap data is requested by a module the data will be taken from hard disk and when a new bitmap is given by a module it will be placed at hard disk. After that by delete method the memory which was taken for bitmap data can be given back to CPU.

6.1.3 GISImage Class

This class is a super class of GISObject. Besides normal data it has the external parameters of the camera and a camera object which contains internal parameters of camera.

6.1.4 CameraObject Class

This class is a container of internal parameters of a camera. Every project can have a default camera.

6.2 ImageProcessor Class

This class is the core of ImageProcessor module. It is a simple image editor at all. When the openImageProcessor method of the ProjectManager class called an instance of this class will created. It has access all the data and methods of the current project. Any GISObject of the current project can be modified by operations of this class.
As data, it contains a pointer of the current project and a set of GISObject pointers which are opened at ImageProcessor. It can open more than one image at one time. Operations will be applied on current GISObject.
It has complex image processing methods like filter convolution besides simple operations like adjusting brightness or contrast.

6.3 DEMExtractor Class

This class is the core of DEMExtractor module. It can access all data of the project. To put it simple, it will take two GISImages and by using their parameters it will register second image to first image and then extract the digital elevation model of the intersection region. For registering it will use static ImageRegisterer class.

6.4 OrtophotoGenerator Class

This class is the core of OrtophotoGenerator module. It can access all data of the project. To put it simple, it will take one GISImage and one GISDEM. By using the parameters of the GISImage it will generate corresponding orthophoto of the given image.

6.5 MosaicMaker Class

This class is the core of MosaicMaker module. It can access all data of the project. To put it simple, it will take two GISObjects and produce a GISMosaic. For the registering part as DEMExtractor module it will use static ImageRegisterer class's methods.

6.6 DEMDisplayer Class

This class is the core of DEMDisplayer module. It can access all data of the project but it can not change any of them. To put it simple, it will take one DEM and displays the area at different types. It can display as a contour-based map, a color-based height map or a 3-D relief map. At 3-D relief map user can explore or view the area by a free camera.

7. Component Design

The component diagrams are were partitioned into subsections in order to fit them on the report so some connections are not shown deliberately.

7.1 Project Manager

The GISObject is not shown in order to avoid clutter.

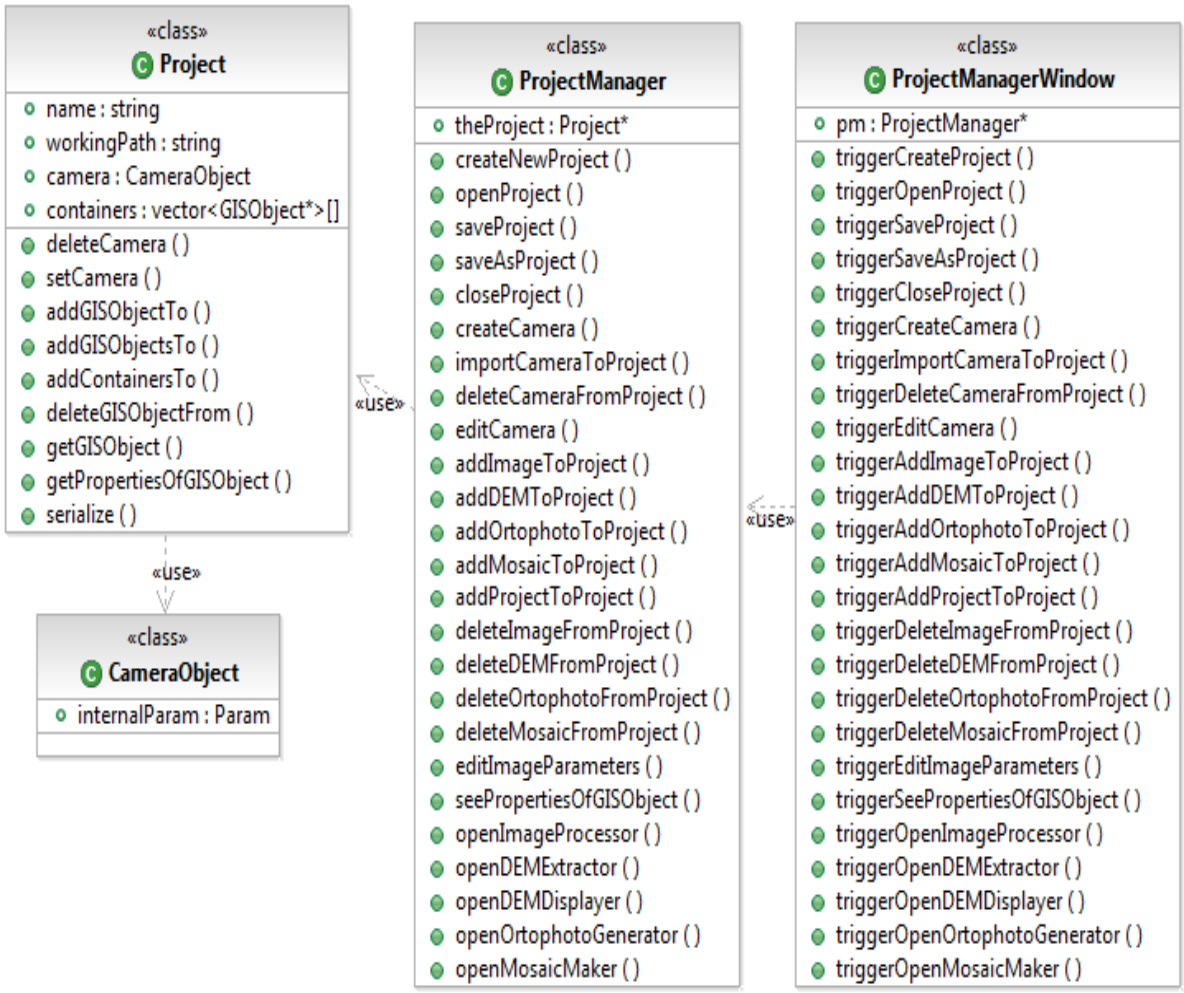

Figure.7.1 Project Manager Component

7.2 DEM Extractor Module

GISDEM object was removed from the figure.

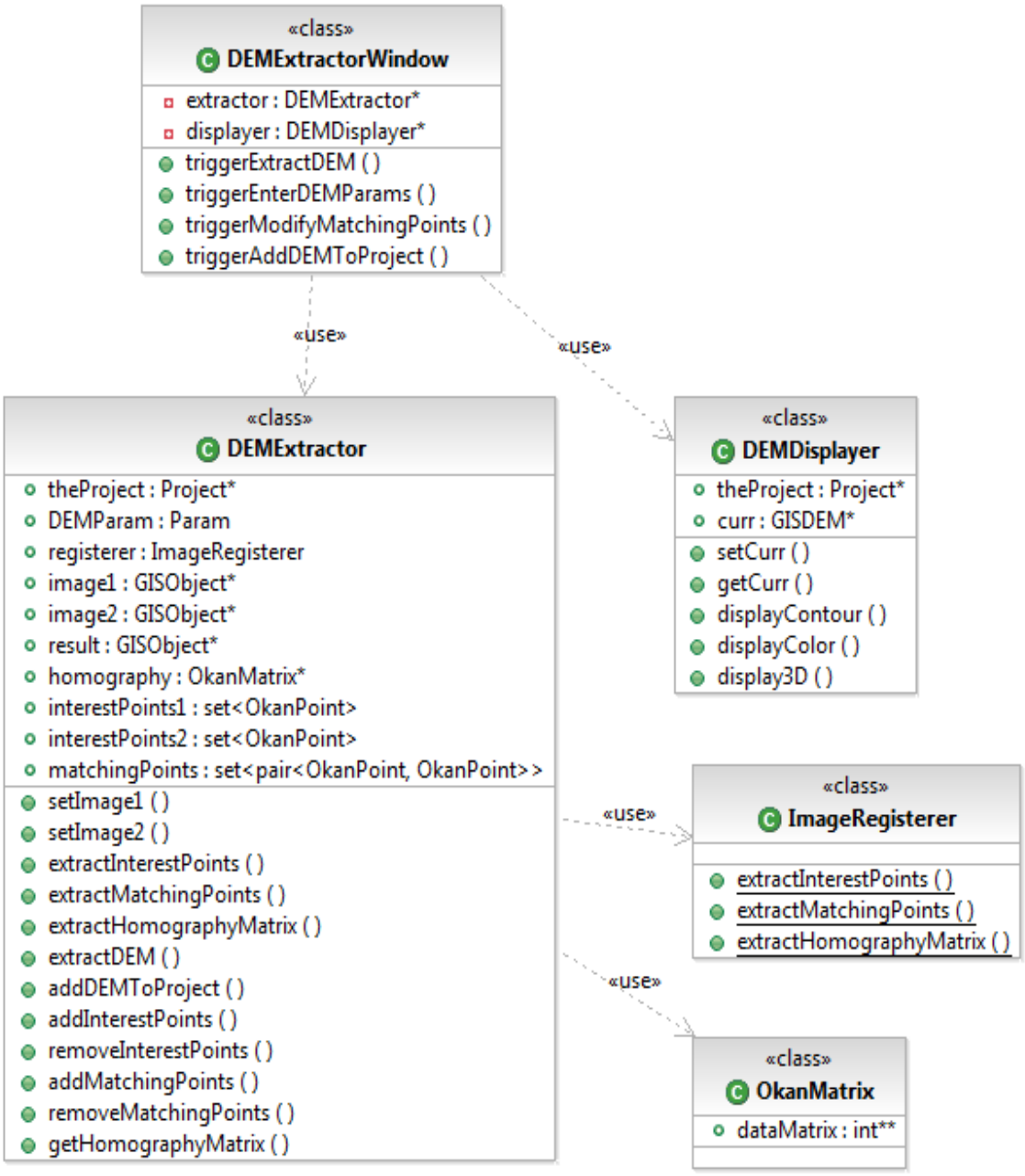

Figure.7.2 DEM Extractor Module

7.3 Mosaic Maker Module

GISObject and OkanPoint classes were removed from the diagram.

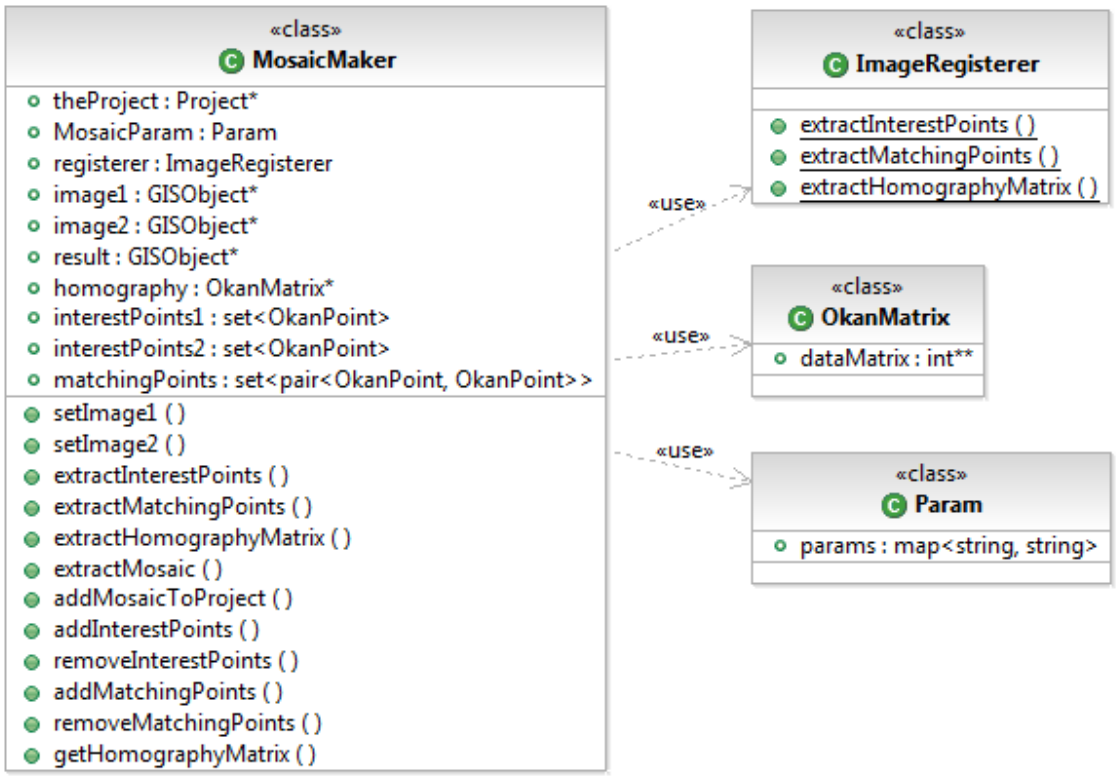

Figure.7.3 Mosaic Maker Module

7.4 Image Processor Module

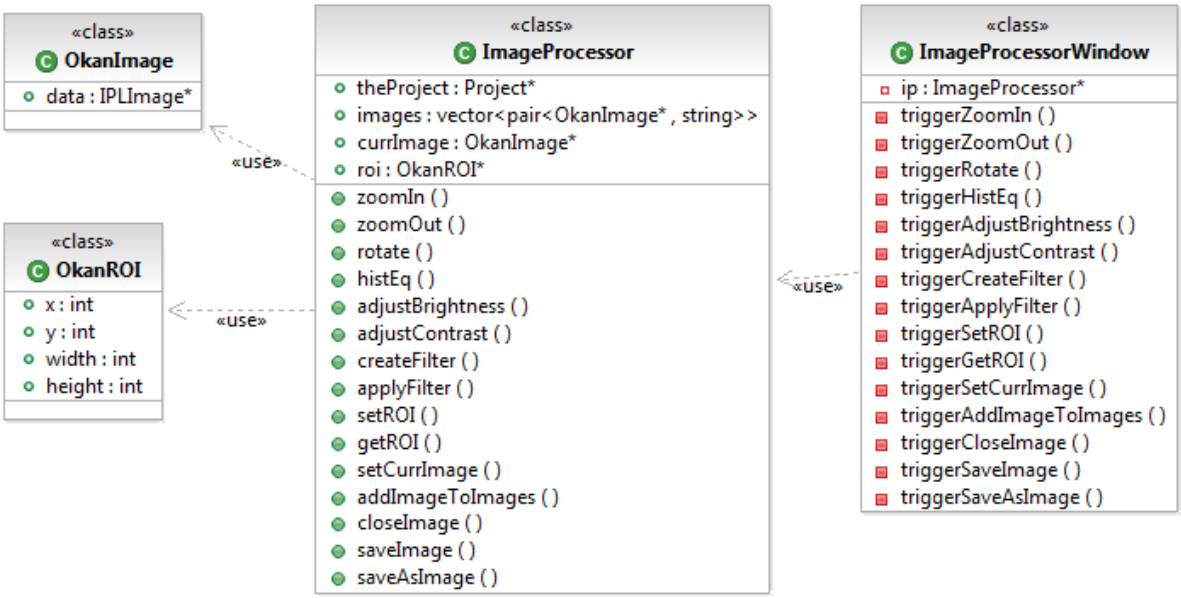

Figure.7.4 Image Processor Module

Image processor module works independent of the other module's workflow. The ROI class is a helper class which defines a region of interest.

7.5 Output Objects

These objects are the basic outputs of modules. Their common properties were realized with inheritance hierarchy. Main vectors holding up casted GISObjects exist in Project class.

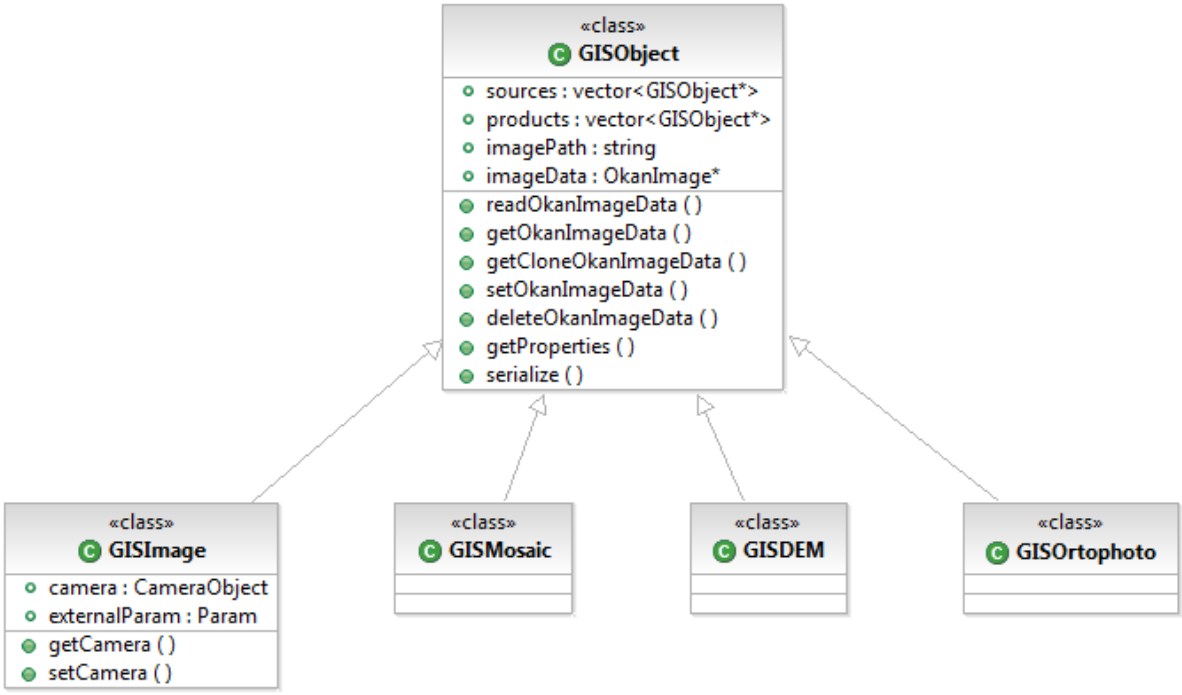

Figure.7.5 Output objects' class diagram

7.6 Helper Classes

These classes are used throughout the system. Basically, a wrapper class for matrices, point and region of interest classes exist.

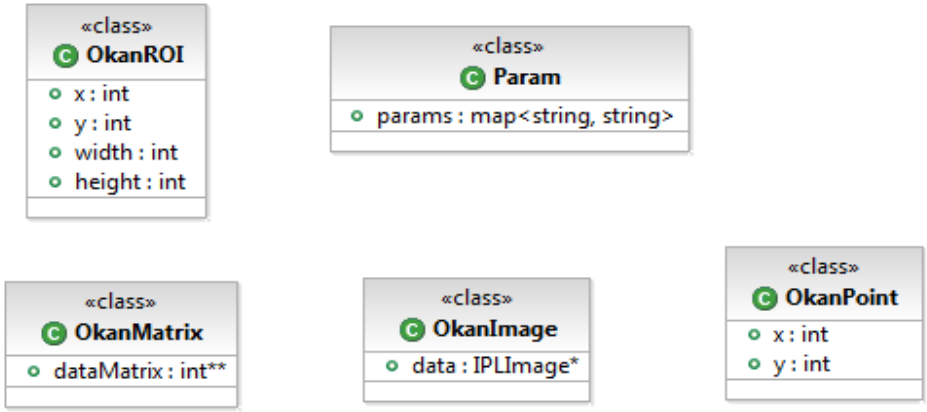

Figure.7.6 Helper classes

7.7 Image Viewer and Help Window

Image viewer is the frame in which images and their corresponding world data can be viewed.

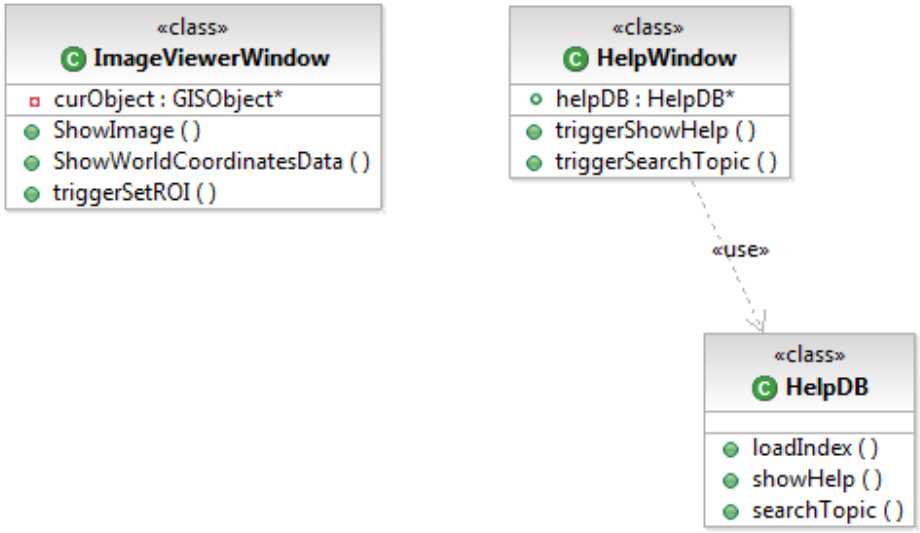

Figure.7.7 Image Viewer and Help Window

8. Process Description

Data flow diagrams and state diagrams provide a basic view of the system and its dynamic properties.

8.1 Data Flow Diagrams

These data flows give an overview of the general data flow among processes. The details are left aside since component design is already detailed and includes information about dynamic content.

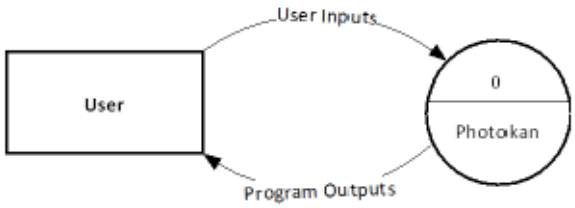

Figure.8.1 Level 0 DFD

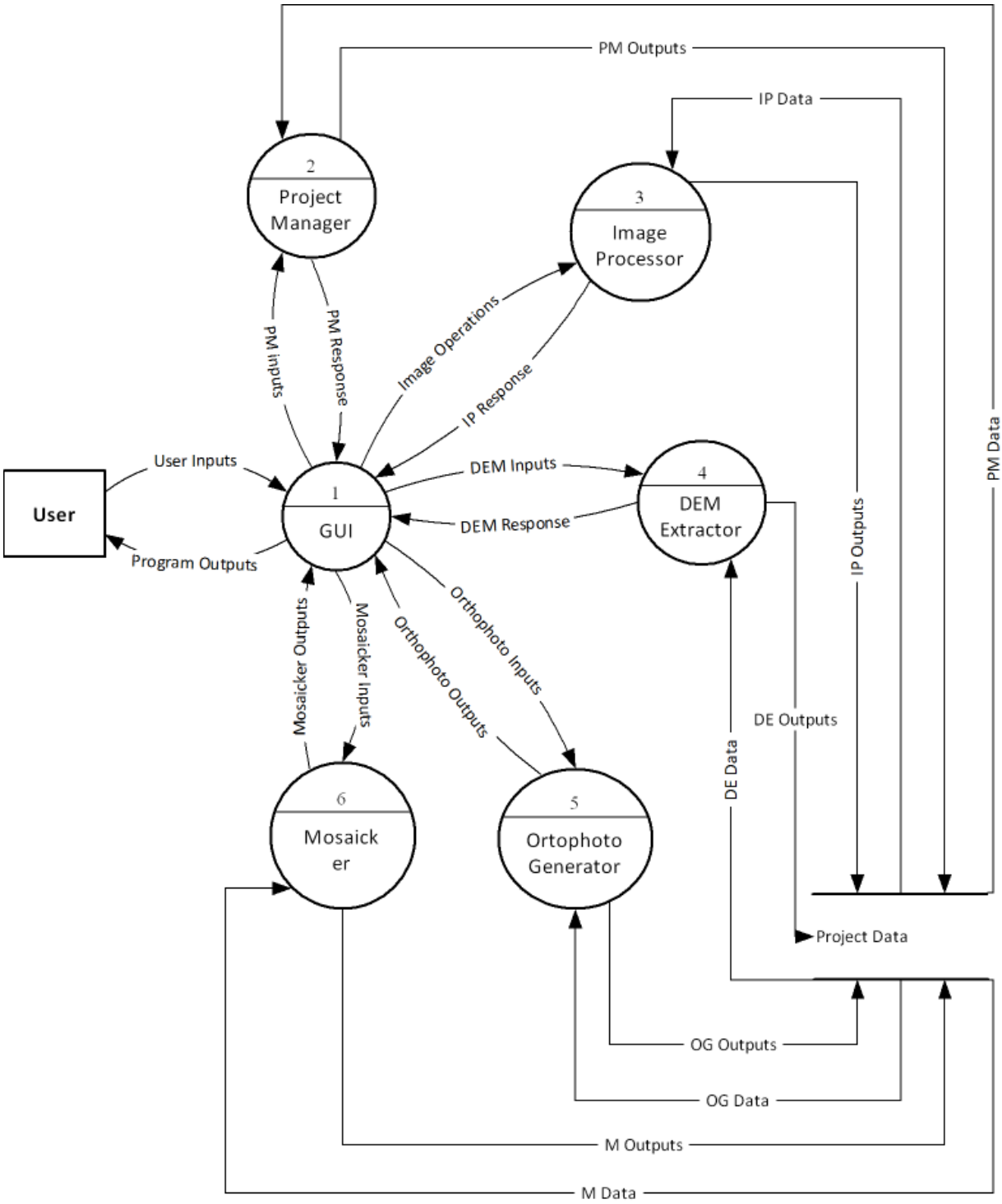

Figure.8.2 Level 1 DFD

8.2 Data Dictionary

Corresponding definitions are given for the term used in the data flow diagrams.

Name	User Inputs
Where/how used	Photokan (0) *input*
Description	Every command that the user issues and every input the user supplies

Name	Program Outputs
Where/how used	Photokan (0) *output*
Description	Every output that the program generates

Name	PM Outputs
Where/how used	Project Manager (1) *output*
Description	The registration data generated by project manager

Name	PM Data
Where/how used	Project Manager (1) *input*
Description	Every kind of information available in Project Data datastore

Name	IP Data
Where/how used	Image Processor (3) *input*
Description	Image information available in Project Data datastore

Name	IP Outputs
Where/how used	Image Processor (3) *output*
Description	Processed output image

Name	PM Response
Where/how used	Project Manager (2) *output* GUI (1) *input*
Description	The result of Project Manager operation

Name	PM inputs
Where/how used	Project Manager (2) *input* GUI (1) *output*
Description	The user command issued to the project manager

Name	Image Operations
Where/how used	Image Processor (3) *input*

	GUI (1) *output*
Description	The image to be processed along with the requested operation

Name	IP Response
Where/how used	Image Processor (3) *output* GUI (1) *input*
Description	The resulting image

Name	DEM Inputs
Where/how used	DEM Extractor (4) *input* GUI (1) *output*
Description	The user command issued to the DEM extractor

Name	DEM Response
Where/how used	DEM Extractor (4) *output* GUI (1) *input*
Description	The generated DEM representation

Name	Orthophoto Inputs
Where/how used	Orthophoto Generator (5) *input* GUI (1) *output*
Description	The issued command to the Orthophoto generator

Name	Orthophoto Outputs
Where/how used	Orthophoto Generator (5) *output* GUI (1) *input*
Description	The resulting image

Name	Mosaicker Inputs
Where/how used	Mosaicker (6) *input* GUI (1) *output*
Description	The issued command to the mosaicker

Name	Mosaicker Outputs
Where/how used	Mosaicker (6) *output* GUI (1) *input*
Description	The resulting combined image

Name	M Outputs
Where/how used	Mosaicker (6) *output*
Description	The resulting mosaick which will be stored in the Project Data datastore

Name	M Data
Where/how used	Mosaicker (6) *input*
Description	The images stored in the Project Data datastore and their registration info

Name	OG Outputs
Where/how used	Orthophoto Generator (5) *output*
Description	The Orthophoto image to be stored in the Project Data datastore

Name	OG Data
Where/how used	Orthophoto Generator (5) *input*
Description	The images stored in the Project Data datastore and their DEM data.

Name	DE Outputs
Where/how used	DEM Extractor (4) *output*
Description	The generated DEM data to be stored in the datastore

Name	DE Data
Where/how used	DEM Extractor (4) *input*
Description	The images stored in the datastore and their camera information

8.3 DEM Extraction Sequence Diagram

This diagram provides an overview of the run time aspects of the system. The other modules behavior also almost fit to this pattern. The operator feedbacks are taken as much as possible to leave space to the user to correct the mistakes.

9. Design Constraints

The design constraints can be partitioned into three classes. Time constraints exist since schedule is fixed by syllabus of the course and performance constraints are a result of the nature of the problem.

9.1 Time Constraints

Since the time constraints are already predefined at the outline of the senior project, the project must be completed by June and the prototype must be completed at the end of this semester as also defined in the Gantt chart. For a healthy progression of the project, time constraints are vital for our group. Minimizing potential risks that will harm the progression should only be avoided with a strict schedule.

Group members should be aware of their responsibilities for the project and arrange their timing carefully for a successful final product.

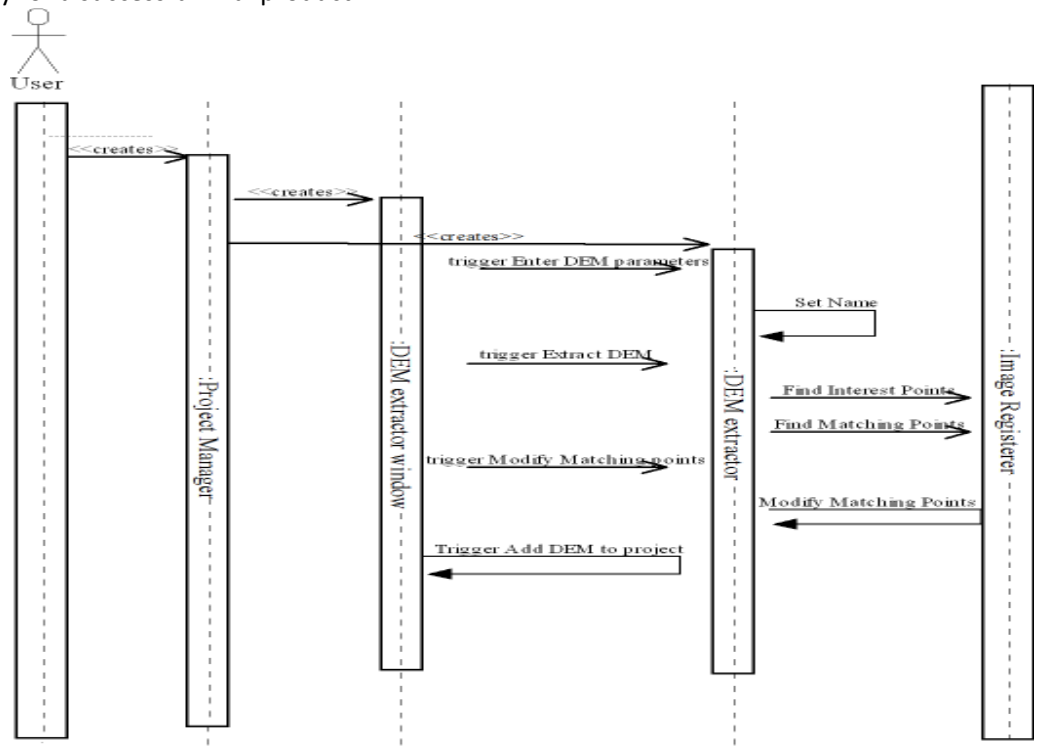

Figure.8.3 DEM extraction sequence diagram

9.2 Performance Constraints

Large file sizes are a challenging area for our team during the project, so efficiency is a critical issue. Team must cope with the efficiency problem and performance is a very important constraint. At each phase of the project, performance must be carefully considered.

9.3 Software Constraints

Software resources are needed during the project. The software requirements that our team benefit are open source components and the soft resources that we need for the project are mostly available on open source communities. Team is planning to work on Linux environment but compatibility of all environments is optional. Since we work on Linux environment, C++ meets our demands so team has decided to use C++ programming language. The graphical user interface is implemented by using wxWidgets Library. Rational Software Architect has been preferred for use case diagrams, activity diagrams and UML diagrams. Smart Draw has been used for state diagrams and sequential diagrams. Microsoft Office and Adobe Professional for documentation and Concept Draw for Gantt Chart are frequently used by group members.

9.4 Hardware Constraints

To meet software constraints, hardware requirements must be carefully planned. Basic requirements that have been estimated before are Pentium 4 or equivalent AMD processor, 512mb of ram, 1GB available hard disc space. Hardware needed by the development environment doesn't exceed the

hardware needed for the proper running of Photokan in computational power so the requirements don't vary amongst developers and users.
The requirements will not raise an issue since team members all have fast and reliable computers that meet hardware demands. The computers in the department's computer labs also offer us opportunities to overcome possible hardware troubles.

10. Design Constraints

The team has a schedule defined as a Gantt chart. The schedule has milestones which are synchronized with reports and the implementations. The current stage of the project can be viewed from the chart. Also the roadmap is discussed in the future work.

a. Gantt chart

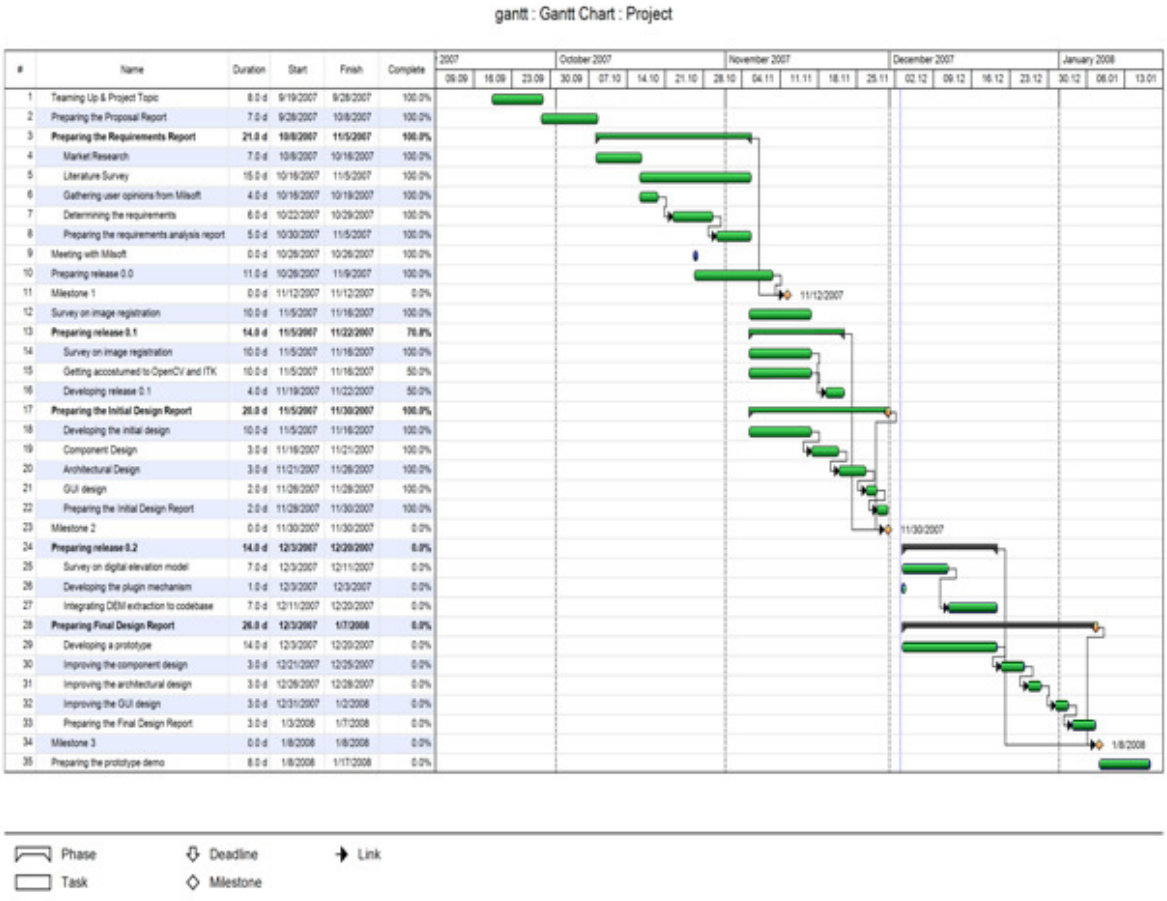

Figure.10.1 Updated Gantt chart

The Gantt chart was updated to reflect the current status of the project.

b. Future Work

Future work exists for both surveying the subjects and implementation.

10.2.1 Survey Topics

Since the project has started, Photokan has come a long way and the teams experience on softcopy photogrammetry has been enriched as the project progressed but this is a huge research area so a limit on survey topics can be determined. The members will widen their knowledge on digital elevation model extraction and orthorectification. The current knowledge on image registration and mosaicking should also be enriched with new methods since employing different methods for a problem is essential in image processing in order to able to process large amounts of data of varying contexts. Photokan is integrating the methods and experience in the area into solutions, so developing the theoretical foundations, customizing and integrating the algorithms is a never ending work.

10.2.2 Implementation

While the current state of the implementation lacks implementation of some graphical user interface elements, their design and style have been considered extensively. The future work is realizing these concepts in the developed framework.

Another important point of consideration for optimization is image pyramids which result in huge gains in speed. The image processing background of Photokan is supplied partly by OpenCV which provides quick implementation of image pyramids. Image pyramids will be integrated into the data store infrastructure (by making the pyramid generation the first step after an image is imported into the project.)

The current codebase is amongst the candidates for improvement. The coding conventions have been violated in certain places which needs revision. Design Choices

During the design phase, the team has strived to leave as much as space for flexibility in choices. The needs were provided after surveying for viable alternatives.

11.1 Software and Implementation Environment Choices

One of the important software choices is choosing the operating system on which the program will be hosted. We decided to use Linux operating system since the market lacks the photogrammetry tools for Unix derivatives; also Linux is a much more reliable system than Windows, providing a solid ground. While the primary target operating system is Linux, the team still strives to make the project cross platform.

The team members are free in their choice of development environment as long as they do not break the rules and the conventions.

As for the implementation environment, Photokan is written using C++ and makes use of the open source libraries:

- OpenCV: An open source computer vision library and image processing library.

- ITK (Insight Toolkit): An image segmentation library specifically developed for medical imaging but still applicable in other areas.
- wxWidgets: A graphical user interface library which also has other generic functionalities such as string processing etc.
- GDAL (Geographical Data Abstraction Library): A library capable of loading almost all geographical image file formats along with their world data files.

Another important thing is that the team has made decisions on conventions to provide a uniform codebase. It was decided to use Sun Microsystems' coding conventions [1].

11.2 Hardware Choices

Linux is capable of running on many known platforms so our hardware choices are bound by the Linux operating system and the fore mentioned libraries. Photokan does not specifically require a hardware but it is tested on Intel architecture and still a lower bound on computational power exists.

As for the team, members are free in their hardware choices for the development process. All members' hardware satisfies the minimum requirements and in case a hardware problem occurs, the department's laboratories exist.

References

[1] http://java.sun.com/docs/codeconv/