



Middle East Technical University
Department of Computer Engineering

Computer Engineering Design I
fall 2010

Detailed Design Report

for

bookwiser augmented reality for libraries

The Ballmer Peak

Content

1 Introduction	4
1.1 Problem Definition.....	4
1.2 Purpose.....	4
1.3 Scope.....	5
1.4 Overview.....	5
1.5 Definitions , Acronyms and Abbreviations.....	6
1.6 References.....	7
2 System Overview	8
3 Design Considerations	10
3.1 Design Assumptions, Dependencies and Constraints	10
3.2 Design Goals and Guidelines	13
4 Data Design	15
4.1 Inherited OpenCV Helper Data Structures.....	15
4.2 Data Description	19
4.3 Data Dictionary	29
5 System Architecture	35
5.1 Architectural Design	35
5.2 Description of Components	38
5.2.1 Object Recognizer Component.....	38
5.2.2 World Model Component.....	47
5.2.3 Graphic Component.....	53
5.3 Design Rationale.....	57
6 User Interface Design	58
6.1 Overview of User Interface	58
6.2 Screen Images	61
6.3 Screen Objects and Actions	70
7.Detailed Design	77
7.1.Object Recognizer Component.....	77
7.2.World Model Component.....	90
7.3.Graphic Component.....	96
8 Libraries and Tools	101

9 Time Planning (Gantt Chart)	104
10 Conclusion	105

1 Introduction

1.1 Problem Definition

Libraries are the most important places for the people who are reading books and doing researches; that is to say, people who want to learn something new usually go to the libraries.

However, this process is not so easy that people waste a lot of time while searching what they want. People have to spare considerably much time while searching for a book, or books that s/he can like in a genre that s/he wants.

Moreover, the same situation happens for the people who knows exactly what they want, namely search for a specific scientific magazine or specific book or even the books of a specific author.

Since the current databases of libraries lead to the user only the correct floor of the library and gives the correct bar-code of the shelf that the user is looking for, this waste of time is inevitable. Also, A user who wants to read a book but does not have a decision on which book to read has to take each book from the shelf, open each book and try to find some information that may help her/him to decide.

Making the user's travel in the library more efficiently by recognizing objects namely the shelves, the books and even the librarian, after that showing information about the library environment on user's screen using augmented reality concept is a perfect solution to these kind of problems. Bookwiser is the application that leads the user to the correct place in an reasonably rational time.

1.2 Purpose

This document includes initial design of Bookwiser Software Project.

The project uses augmented reality concept to make user get easily informed about the organization of the library, namely the books, content of the books and also interact with the librarian.

The project is essentially composed of three parts.

First part is the object recognition part which includes the shelf recognition, book recognition and librarian recognition by using object detection techniques.

The second part is the interaction with the library database from which all the information about a book is extracted.

The last part of the project is the GUI part which is going to show the modified video stream

augmented with information related to recognized objects on the camera, provide options for user to choose the flow of the application and make user's experience in the library more effective.

The intended audience is for this document is people who deal with further development of Bookwiser.

1.3 Scope

The project is named "Bookwiser", since the application seems to know everything about every book in the library, and helps user about books

Our project will be a augmented reality application for a library environment which works on mobile devices. When a user starts the application, Bookwiser first enables the video in the library. While the user is moving with the camera in the environment, Bookwiser will capture data from environment.

The user may choose to search a specific book or decide just walk and get informed about different kind of books in the library. In the first option the user gives the name of the book that he/she wants to reserve and Bookwiser will guide the user in order to find the correct shelf and the correct book. The other option is that the user has no idea about what he/she exactly wants , therefore Bookwiser will guide the user that while user is walking around the library, it will give information about the shelves and books that is in the predefined range of the camera.

When Bookwiser comes up with a book shelf belonging to a specific kind of books, it will detect the shelf by using object recognition in a predefined range. After the shelf detection & classification, the project will inform the user about the content of the books in the shelf.

When Bookwiser recognizes a book, it will show the rating of the book to the user using augmented reality and it will give detailed information to the user if the user wants. It will give user choices like rating the book or reserving the book.

Moreover, there will be librarians in the library for further questions and they will be detected using the predefined specifications. We think that people going to the library are mostly wasting their time for searching the correct place of a book or deciding which book to read. So that, Bookwiser very helpful to guide people in libraries.

1.4 Overview

The contents of this document consist of 9 basic parts:

- Introduction where the problem definition, scope, overview, definitions/abbreviations and references are explained,

- System Overview where the general description of the system is provided,
- Design Considerations where the special design issues, dependencies, constraints, goals and guidelines are noted,
- Data Design,
- System Architecture where the description of the program architecture is presented by describing each of components,
- User Interface where the user perspective of the project is explained,
- Libraries and Tools,
- Time Planning where the Gantt Charts are presented,
- Conclusion where the all parts concluded

1.5 Definitions and Abbreviations

Already-recognized Object: An object that is recognized when the previous frame is processed.

Augmented Reality: Term for a live direct or indirect view of a physical real-world environment whose elements are augmented by virtual computer-generated sensory input such as sound or graphics. For Bookwiser application, augmented reality represent the user graphical output that shows the recognized object properties.

Bookwiser: Name of the project.

Camera: A device that provides a video stream

Candidate Object: An assumption of boundary areas for possible images on an image

COI: Color channel of interest in an image .

CPU: Hardware component also named as processor.

Filter: A sequence of image processing operations to enhance an image.

Frame: An image captured from the camera.

GPU: Graphics Processing Unit

GUI: Graphical User Interface

Image Processing: Image processing is any form of signal processing for which the input is an image, such as a photograph or video frame. For Bookwiser application, image processing is the process of generating information from video stream captured by user camera.

Library: Real building where books or other kind of documents are stored for public use.

Object recognition: Object recognition is the process of retrieving identifying information about objects. Object recognition process is directly connected with image processing.

Object recognition range: The closeness of the camera to the real object which provides enough detail to recognize the object.

ROI:Region of interest in an image

Video Stream: Video stream is the common name of Bookwiser's input and output format. It consists of stream of real time images.

1.6 References

1. IEEE Std 830-1998: IEEE Recommended Practice for Initial/Detailed Design Requirements
2. Class Diagrams, IBM,
<http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/>
3. Sequence Diagram, IBM
<http://www.ibm.com/developerworks/rational/library/3101.html>
4. Component Diagram, IBM
<http://www.ibm.com/developerworks/rational/library/dec04/bell/>
5. Data Flow Diagram, Wikipedia,
http://en.wikipedia.org/wiki/Data_flow_diagram

2 System Overview

Bookwiser is a real-time image processing and augmented reality application for libraries. Its main purpose can be described as assisting the library user in his/her library trip via a display device. To be more precise, Bookwiser provides useful information about library objects which are books, shelves and librarians.

To perform its task, Bookwiser processes the video captured by the camera frame by frame, generates the object properties, retrieve context information for these objects from external library database, manipulates the video according to these contextual and visual properties of objects and displays the final modified video frames via a display device.

Since the information shown to the user is purely based on instant camera view, performance is a crucial requirement for Bookwiser. What indicates the performance of the system is the execution times of image processing and manipulation actions. It is better to say that, Bookwiser can be classified as an augmented reality application more than a database driven system.

The core of its design consists of image manipulation and object recognition. On top of this core, there is the user interface which complements Bookwiser's functionality from the aspects of usability and attractiveness.

Except from the internal object database, which is planned not to be an actual database system, library database operations to get context information are almost completely out of the scope of the design for two reasons.

- First reason can be described as the aim of the system, which is recognizing objects in real time. Once the objects are recognized and identified by Bookwiser, desired information can be retrieved by simply querying it.
- Second reason is the obvious system's independence of the database design. It is nonsense to design an internal context database while there is already a well-established library database.

From the aspect of software design, Bookwiser is a usable, reliable and crucially fast object recognition and augmented reality visualization system by which the user can be able to freely walk around the library without spending time for examining the books by himself/herself.

Block diagram of the system is given below:

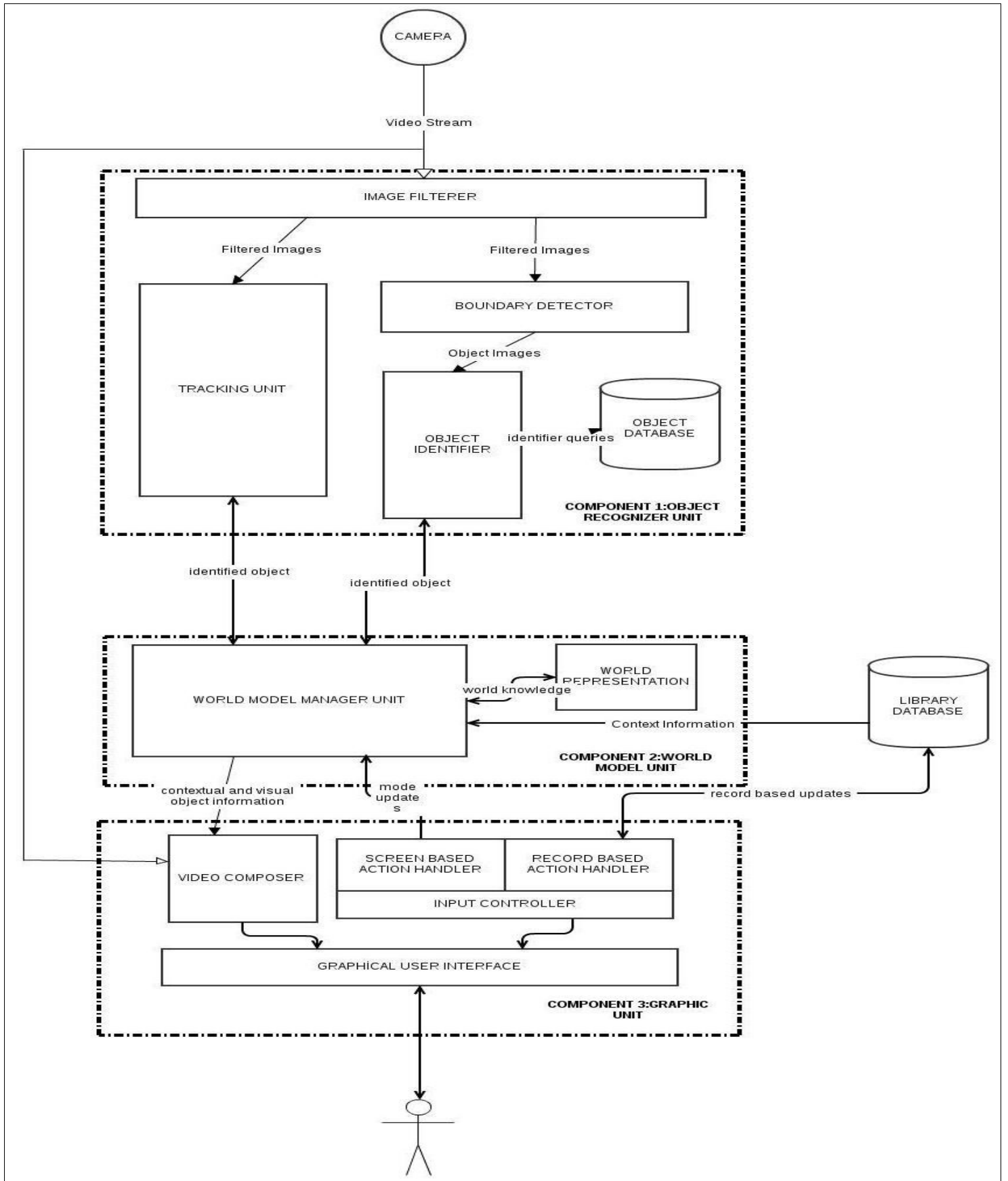


Diagram 2.1 – Block Diagram

3 Design Considerations

In this section, special design issues that should be considered during design and development are stated.

3.1 Design Assumptions, Dependencies and Considerations

3.1.1 Hardware related Assumptions, Dependencies and Considerations

- Bookwiser is designed to work on a portable device , including a camera, a display screen, RAM , hard-disk and processor units , and a graphics card.
- The system is assumed to a work on a device with a minimum of 2.0 GHz Intel processor and a minimum of 500 core GPU, no shared memory.
- The system is assumed to work on a device with a minimum of 20GB hard-disk and 1024Mb RAM.
- Bookwiser is assumed to operate on a device that is able to use wireless network communication. Therefore the device should have a wireless communication device and should be able to connect to a local network or the Internet.
- It is assumed that the system needs a minimum of 5.0 mega-pixels camera to feed the system with images with necessary quality and detail to do object recognition.
- Bookwiser will not use external GPS or any other satellite/GSM based navigation systems or devices to locate or recognize books / shelves.
- User will interact with the system via a keyboard.

3.1.2 Software related Assumptions, Dependencies and Considerations

- The system that Bookwiser will operate on should have a minimum version of Ubuntu 9.0 or Windows XP installed on it. The system should be suitable for QT and OpenCV platforms.
- Bookwiser needs to communicate with the library database , therefore it is assumed that the library database is open to Bookwiser and the library database is able to share information via Internet or a local network. If there's a local network, also the library should provide access to the network via wireless.
- There may be a need for creation of an extra table or a change in the existing table in the library database for user registration and related user information if the library

database doesn't suit application needs. Otherwise, Bookwiser should not make any changes on the structure of the library database in any cases.

- Bookwiser uses the library's own database to show data about recognized books. It is assumed that book information exists in the library database is correct and Bookwiser doesn't have to check the correctness of the data. The application itself is not responsible for book information organization in the database.

3.1.3 Performance related Assumptions, Dependencies and Considerations

- Bookwiser needs to assist the user during its travel in the library therefore the system should respond to the user immediately. The object recognition work should be done in near real-time and the overhead of this operation should be maximum 0.8-0.9 seconds on the worst case.

- To get the information about recognized object from the library the system needs to connect the library, because a wireless connection may take time, after an object is recognized, the overhead of gathering information about the object can take longer. But in the worst case it should take maximum 5 seconds on the worst case to gather the information from library database and to show the user via the graphical user interface.

- Bookwiser should have the capacity to serve all the visitors in a library and because each device that is provided for different users own a different instance of the system scalability is not a problem, on the other hand, because the library database may not be able to serve all Bookwiser users at the same time, connections to the database should be made under the consideration of limits of the server of the library database.

- Bookwiser may not be able to detect objects from long distances, but the system should be designed to at least recognize objects from the distances listed below:

- Books : 2-2,5 meters.
- Shelves : 4-6 meters.
- Librarians : 3-5 meters.

3.1.4 Interface related Assumptions, Dependencies and Considerations

- The interface to interact with the camera is provided by OpenCV library that used by the system. To gather information from the camera, Bookwiser will use OpenCV's camera methods and classes.

- Bookwiser uses a library's own database to gather user, book and shelf data. The interface to communicate with the library database is dependent on library database's architecture. Therefore the system should be designed flexible enough to be able to interact with different library databases.

3.1.5 Safety & Security related Assumptions, Dependencies and Considerations

- The devices which Bookwiser is run on should be tested for health issues, the screen and the device components should obey the standards.
- The library database manipulations done by Bookwiser are limited by the library constraints, book information addition/deletion/manipulation or access to other users' information is not made by Bookwiser to protect library database from possible misuse.
- Users' passwords and detailed information should not be shared with other users to protect the system from illegal access.
- Before unrecoverable operations users should be asked if they are sure about the operations.
- The library, according to its own rules, may block a user's access to the system or just book reservation functionality due to misuse of the system. The decision is library's and only responsibility of Bookwiser is to inform the user via graphical interface.

3.1.6 Standards

- It is assumed that the library have adequate light sources around the objects
- It is assumed that shelves in the library are equally spaced with a minimum of 1 meters and they are placed parallel to each other.
- It is assumed that shelves in the library are standard and they have the same height/width and shape. Shelves should have a color that can be differentiated from the library environment and special marks on the shelves should be apparent and proper enough to classify them.
- It is assumed that books are placed on the shelves with their front side facing the camera and the user. The books are assumed to be placed with equal spaces between them, a minimum of 10 centimeters. Books should have a color that can be differentiated from the library environment and special marks on the books should be apparent and proper enough to classify them.
- Librarians should wear a standard t-shirt with the same color and a specific apparent shape on top-left of their t-shirts` front and back sides, making them recognizable.
- The marks, values or features to recognize and classify books, shelves and librarians should be predefined at the object recognition database. **Bookwiser** will only recognize and classify objects with predefined marks , shapes and colors.

3.2 Design Goals and Guidelines

3.2.1 Usability & Real-time processing:

Bookwiser is a system that assists users actively during their tour in the library. Because the user needs to be responded immediately during their tour, the system should react the changes on the screen or inputs of user via graphical user interface without losing time. Otherwise the system can not be usable for library visitors.

This requirement brings the need that all the algorithms and structures in the system should be implemented to work very efficiently. The object recognition work should be done in near real-time and the overhead of this operation should be maximum 0.8-0.9 seconds on the worst case.

3.2.2 Minimalism:

Bookwiser is a project with a main goal of helping its users; therefore, the user interface of the system should be implemented in a way that it should not bother the user during his/her travel in the library. The graphical elements on the screen that shown as a part of the augmented reality concept should not be blocking the user's view , or disturbing the user. These graphical elements should be designed to be minimalistic and should look natural. User should always be able see the frames coming from the camera in a big part of the screen so that user is always able to move or see the objects around him/her.

3.2.3 The KISS Principle :

All the components , data structures and methods in the design should be implemented in the simplest way to serve the user the better. The main aim of Bookwiser is to create shortcuts for everyday procedures of a library user, so features like getting book information, registering a book etc. should do the maximum work with the least effort from the user. Use of different features of the system should be very simple for the users to use.

The project should not make it more difficult to travel in the library rather that making it much more simple for users.

3.2.4 Don't Repeat Yourself Principle :

To avoid repeats of parts of the system, the system is designed to use some core parts in different sub-components of each components, using an object oriented approach. Developers should stick to this property of the design and avoid duplicate system parts or data structures to increase the performance of Bookwiser.

3.2.5 Principle of Good Enough :

Bookwiser project is designed with a strong consideration of principle of good enough.

Components and functionalities of all system parts are designed to be simple and they can be seen as cores. These cores are suitable for further developments, or additions of extra functionalities. Developers should obey this principle to make the system suitable for future extensions.

4 Data Design

In this section, data elements are described and defined briefly.

4.1 Inherited OpenCV helper data structures

In this section all the related data structures to be used by the developer while using OpenCV library are described.

Because other data structures inherit these helper structures to use OpenCV library efficiently, understanding of OpenCV helper data structures are important to understand Bookwiser's own data structures, components and methods better.

IplImage	
Data structure to represent images.	
int nChannels;	Number of color channels (1,2,3,4)
int depth;	Pixel depth in bits: IPL_DEPTH_8U, IPL_DEPTH_8S, IPL_DEPTH_16U, IPL_DEPTH_16S, IPL_DEPTH_32S, IPL_DEPTH_32F, IPL_DEPTH_64F
int width;	image width in pixels
int height;	image height in pixels
char* imageData;	pointer to aligned image data Note that color images are stored in BGR order
int origin;	0 - top-left origin, 1 - bottom-left origin (Windows bitmaps style)
int widthStep;	size of aligned image row in bytes
int imageSize;	image data size in bytes = height*widthStep
struct _IplROI *roi;	image ROI. when not NULL specifies image region to be processed.
Char *imageDataOrigin;	pointer to the unaligned origin of image data (needed for correct image deallocation)

CvMat		
2D Array		
int type;	elements type (uchar,short,int,float,double) and flags	
int step;	full row length in bytes	
int rows, cols;	dimensions	
int height, width;	alternative dimensions reference	
union data;	uchar* ptr;	data pointer for an unsigned char matrix
	short* s;	data pointer for a short matrix
	int* i;	data pointer for an integer matrix
	float* fl;	data pointer for a float matrix
	double* db;	data pointer for a double matrix

CvScalar		
Scalars		
double val[4];	4D vector	

CvPoint		
2D point with integer coordinates		
int x;	x-coordinate, usually zero-based	
int y;	y-coordinate, usually zero-based	

CvPoint2D3F	
2D point with floating point coordinates	
float x;	x-coordinate, usually zero-based
float y;	y-coordinate, usually zero-based

CvSize	
pixel-accurate size of a rectangle	
int width;	width of the rectangle
int height;	height of the rectangle

CvSize2D32f	
sub-pixel accurate size of a rectangle	
float width;	width of the rectangle
float height;	height of the rectangle

CvRect	
offset and size of a rectangle	
int x;	x-coordinate of the left-most rectangle corner[s]
int y;	y-coordinate of the top-most or bottom-most rectangle corner[s]
int width;	width of the rectangle
int height;	height of the rectangle

CvSeq	
Growable sequence of elements	
int flags;	miscellaneous flags
int header_size;	size of sequence header
struct CvSeq* h_prev;	previous sequence
struct CvSeq* h_next;	next sequence
struct CvSeq* v_prev;	2nd previous sequence
struct CvSeq* v_next;	2nd next sequence
int total;	total number of elements
int elem_size;	size of sequence element in bytes
char* block_max;	maximal bound of the last block
char* ptr;	current write pointer
int delta_elems;	how many elements allocated when the sequence grows (sequence granularity)
CvMemStorage* storage;	where the seq is stored
CvSeqBlock* free_blocks;	free blocks list
CvSeqBlock* first;	pointer to the first sequence block

CvSURFPoint	
Detected SURF keypoint	
CvPoint2D32f pt;	position of the feature within the image
int laplacian;	-1, 0 or +1. sign of the laplacian at the point.
int size;	size of the feature
float dir;	orientation of the feature: 0..360 degrees
float hessian;	value of the hessian

More detailed information about the mentioned data structures and related functions is available in OpenCV documentation at Official OpenCV page :

<http://opencv.willowgarage.com/wiki/>

4.2 Data Description

Our system Bookwiser, is transformed into files that contains classes. There are mainly three data units in our system in a data description manner. These units are Input Controller, Graphical User Interface Unit and Object Recognizer Unit. These units generate interfaces as data structures to interact with other components. In addition to these, there are two libraries and two external devices, a Camera and a keyboard, that Bookwiser is in a relationship with.

4.2.1 Library Database:

The Library Database and it is not directly related to our system, because, this is the database where all book, shelf and user related, “object recognition independent” information are kept in and this database is not a part of the design of Bookwiser. Organizing this database is not a responsibility of the application, instead this external database is only used to gather information except the registration and rating/reservation based user interactions.

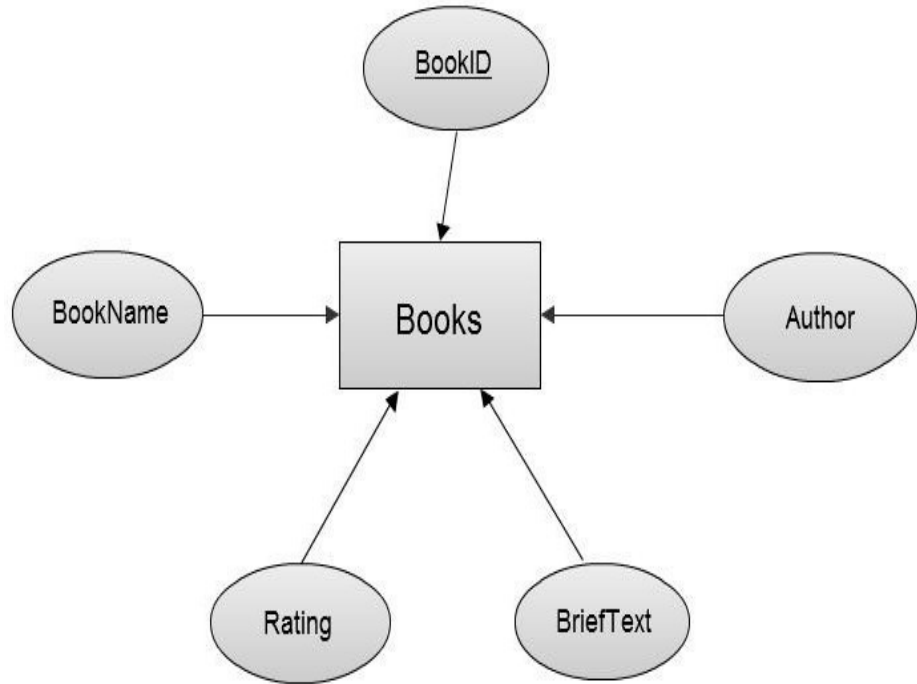
- The data entities that assumed to be provided by the library database which will not be modified by Bookwiser is listed below:

Books : table for book information

This table corresponds to the books in the library, it holds the information about the books in the library.

- BookID: integer
- BookName: text
- Author: text
- Rating: float
- BriefText: text

Diagram 4.1 – ER Diagram



Shelfs : table for shelf information

This table is needed to represent the shelves in the library, it holds the information about the shelves in the library.

- ShelfID: integer
- Genre: text

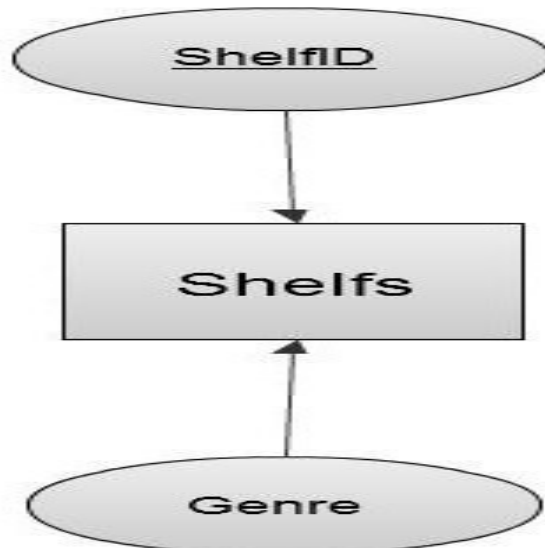


Diagram 4.2 – ER Diagram

Librarian: table for librarian information

This table is needed to represent the librarians working in the library.

- LibrarianID: integer
- Name: text
- SurName: text

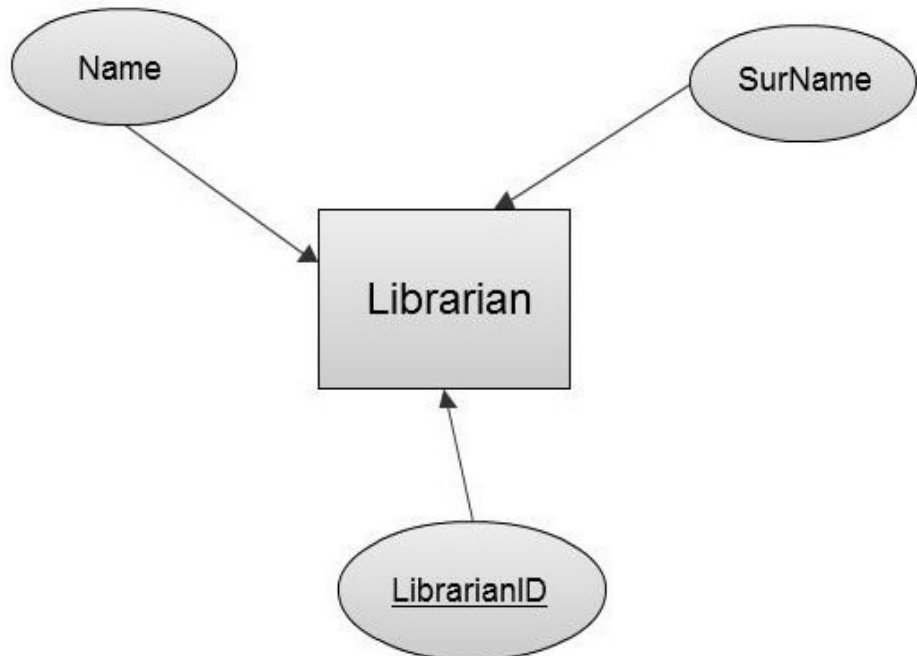


Diagram 4.3 – ER Diagram

- The only exception to this rule is where user registration, login and unregistration processes. For these operations mentioned above, the tables and fields which a library database should hold are listed and if they does not exist in a library database, they should be added:

Enrollees: table for user information to be used in registration procedures

This table is needed in order to keep some basic information for library enrollees registered to the Bookwiser system. There is only very necessary information about a library enrollee.

- UserId: integer
- Name: text
- SurName: text
- Job: text
- Gender: text
- Telephone: integer
- Address:text
- MailAdress: text

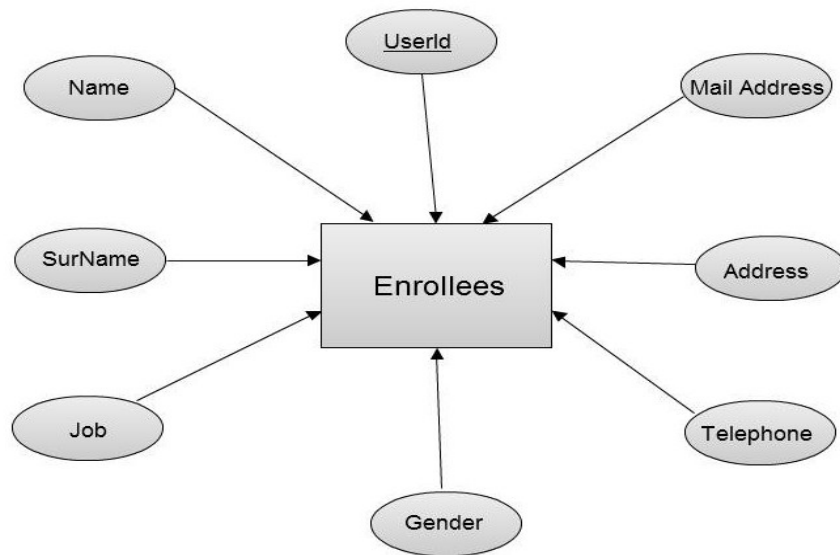


Diagram 4.4 – ER Diagram

Users: table for user information to be used in login procedures

This table is definitely needed in order to keep the information of the Bookwiser Software System users in a safe manner.

- ID: integer
- Password: text
- PreferredLanguage: text

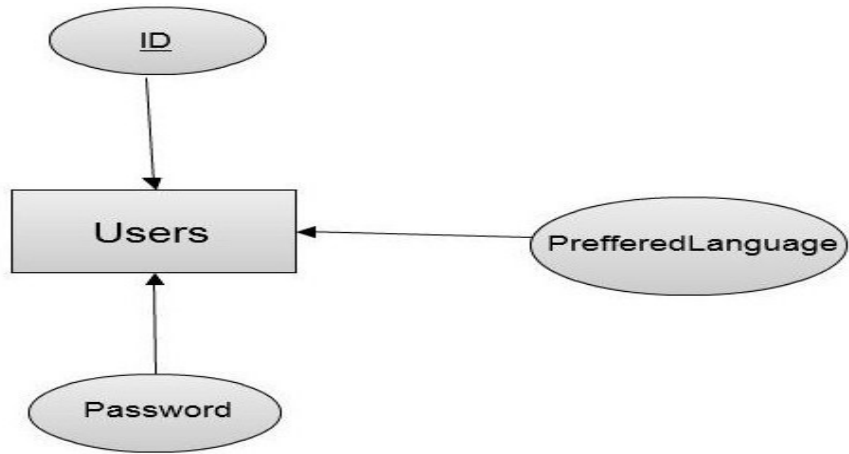


Diagram 4.5 – ER Diagram

- The relationships between the mentioned entities are represented as 3 different tables on the database. FavoriteBooksOfUser and RecommendedBooksOfUser are able to be changed by Bookwiser whereas BookInShelf relation can not be modified by Bookwiser :

FavoriteBooksOfUser : A relationship between Books and Users

We have Books and Users. When a user rates a book highly it is added to the favorite book of that user.

- BookID from Books
- ID from Users

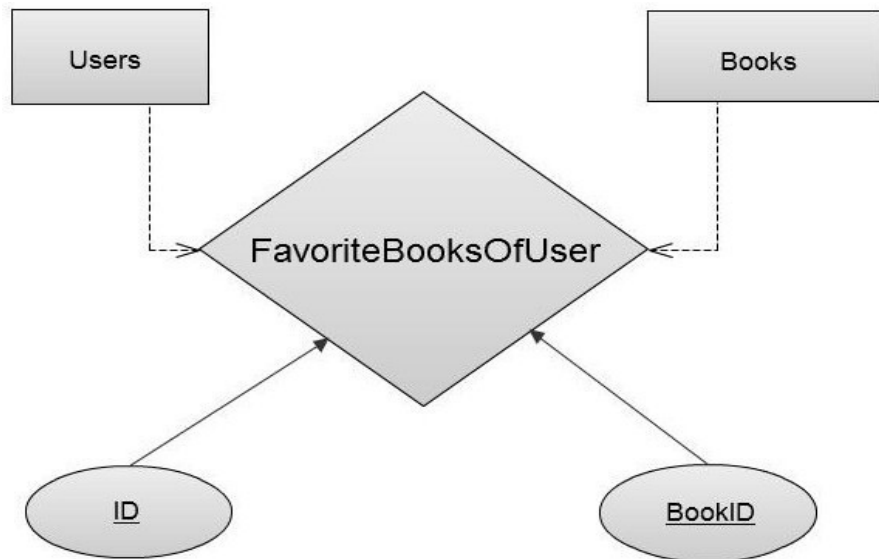


Diagram 4.6 – ER Diagram

RecommendedBooksforUser : A relationship between Books and Users

The books that a user whether like or not can be guessed by some search on his favorite books, etc. Then, the respective books and the user can be put in a relationship.

- BookID from Books
- ID from Users

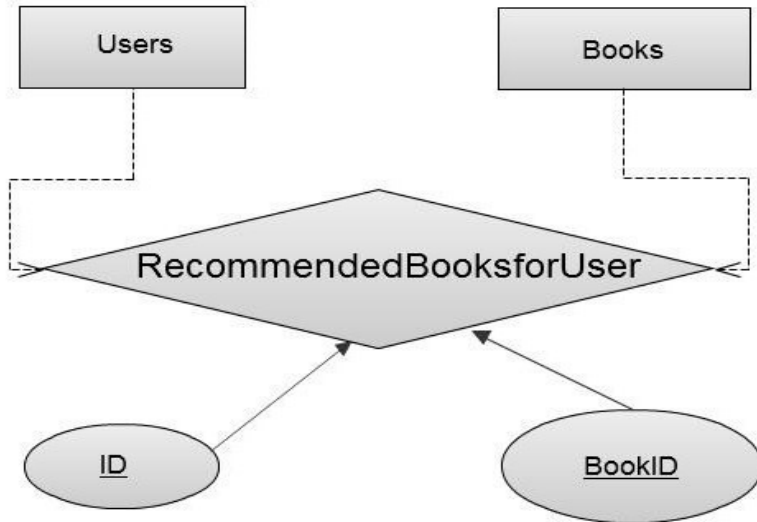


Diagram 4.7 – ER Diagram

BookInShelf : A relationship between Books and Shelves

The fact that a book stands on a specific shell is a relationship, and with the information of the shelf that the book lays on, he genre of book also can be determined.

- BookID from Books
- ShelfID from Shelves

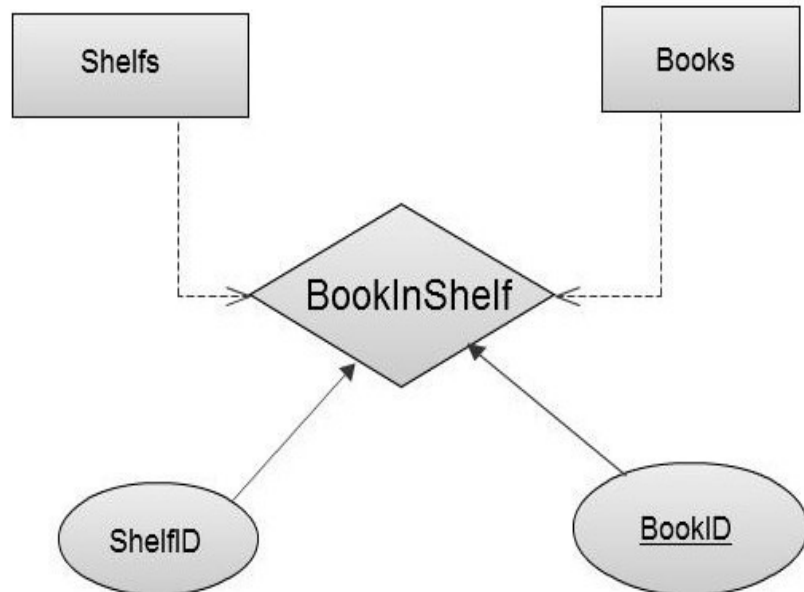


Diagram 4.8 – ER Diagram

4.2.2 Internal Object Database:

The second library is the main library that Bookwiser uses, which is Internal Object Database.

The database is not in actual database form, instead a text file that contains book features consequently. On the start of the application the database is loaded in the memory and used from the memory.

The database keeps features for object recognition purposes.

This database contains:

- id of the book to match the book in the library database after recognition
- width and height of the sample image as integers for SURF matching procedure
- number of SURF features extracted from the sample image of the book
- KeyPoints as CvSurfPoint structures of OpenCV ,
- For each feature 128 SURF descriptors as floats,
- id fields as integers
- type fields as integers
- mean, median and standart deviation values on color histograms

The organization of the text file can be described as:

Internal Object Database
For each book:
<type> <BookId> <# of features> <Width> <Height> <Mean> <Median> <StDev>
For each extracted SURF feature: (Values from CvSurfPoint structure)
<x> <y> <laplacian> <size> <dir> <hessian> <descriptor 1 > <descriptor 2>
....
....

4.2.3 Object Recognizer Component and World Interface:

Object Recognizer Component contains 4 major data structures:

Feature	Contains information of detected object features
Position	Contains information of detected object position
Object	General structure for all information about detected information including features and position of a book.
ObjectList	List for detected objects in a frame

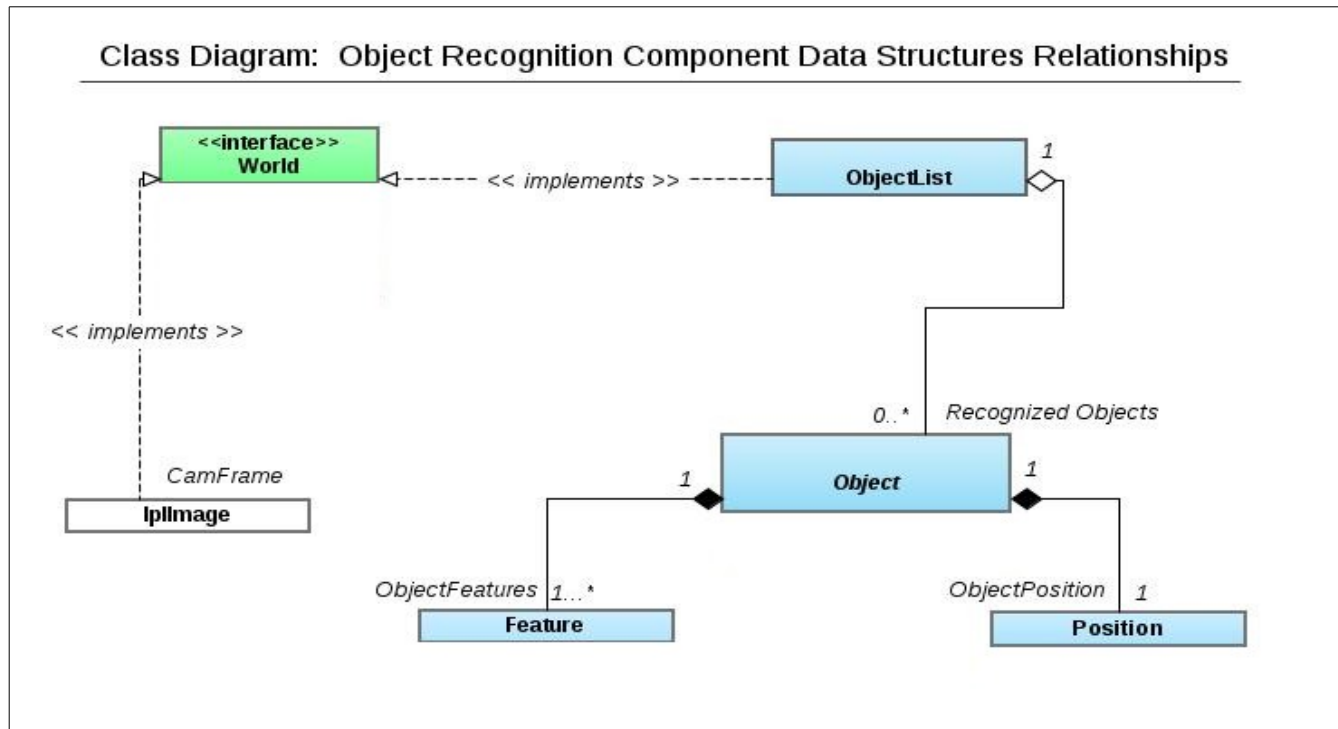


Diagram 4.9 – Class Diagram

4.2.4 WorldModel Component and WorldRepresentation Interface:

World Model Component contains 6 major data structures:

Shelf	Extends Object structure from the World Interface, also containing information gathered from Library Database about shelves
Book	Extends Object structure from the World Interface, also containing information gathered from Library Database about books
Librarian	Extends Object structure from the World Interface, also containing information gathered from Library Database about librarians
ObjectTable	Keeps List of Shelf, Book and Librarian data structures.
State	Keeps information of application state at a moment
StateChart	Stack of application states.

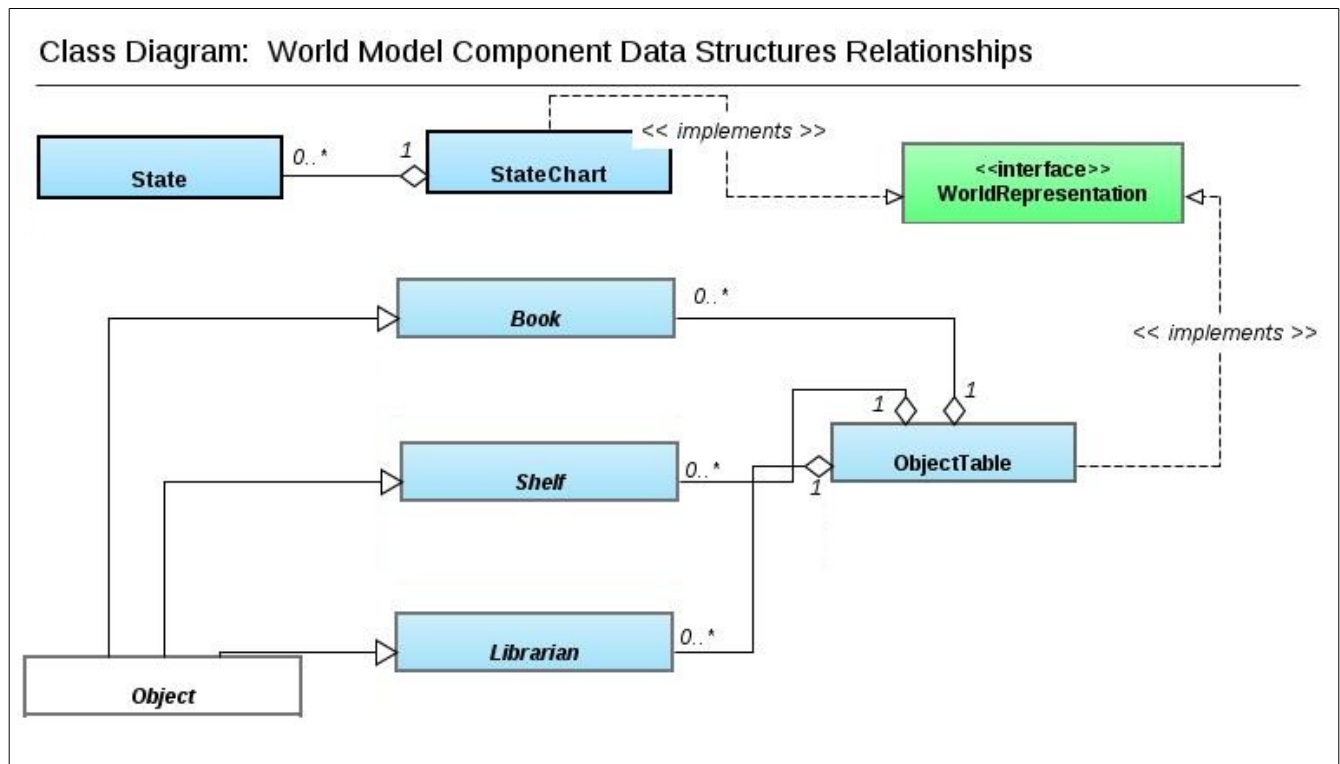


Diagram 4.10 – Class Diagram

4.2.5 Graphic Component and States Interface:

The graphic component doesn't contain any major data structure; on the other hand it generates States Interface by creating a state due to user interaction and serves this to WorldModel Component.

- To understand the data flow between components and the major sub components the following data flow diagram can be followed.

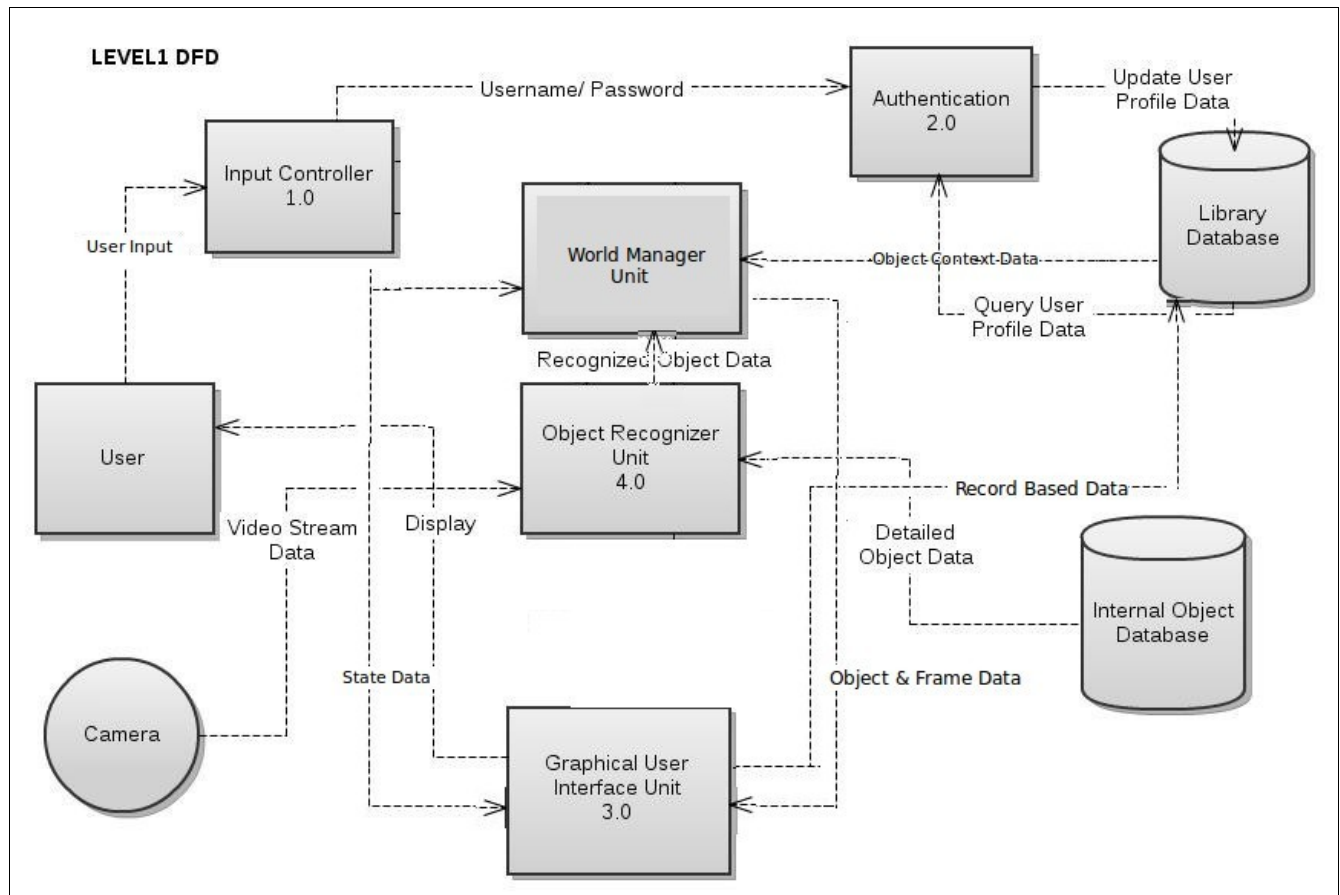


Diagram 4.11 – Data Flow Diagram

4.3 Data Dictionary

Since the approach in the Bookwiser is object-oriented, related data dictionary is composed of data system entities, which are data classes and their related methods. The whole entities and instances are separated according to component which belongs to, in an alphabetic order.

Book	
Holds all the information about the recognized books.	
Data:	
Inherits all data defined in the Object Class.	
int id	Holds the id number of the shelf
String name	Holds the name of the book
int rating	Holds the rating of the book
String author	Holds the name of the author
Methods:	
Inherits all methods defined in the Object Class.	
int getId(void)	Returns the id of the book
void setId(int)	Sets the id of the book
void setName(String)	Sets the name of the book
String getName()	Returns the name of the book
String getAuthor	Returns the name of the author
void setRating(int)	Sets the rating of the book
int getRating(int)	Returns the rating of the book

Feature

Holds the information for a detected object feature

Data:

CvSurfPoint kp;	Extracted key point for detected feature
Float* dc;	Surf descriptors array, with length 64 or 128

Librarian

Holds all the information about the recognized librarians.

Data:

Inherits all data defined in the Object Class.

int id	Holds the id number of the shelf
String Name	Name of the librarian

Methods:

Inherits all methods defined in the Object Class.

int getId(void)	Returns the id of the librarian
void setId(int)	Sets the id of the librarian
String getName()	Returns the name of the librarian
void setName(String)	Sets the name of the librarian

Object	
Holds all the information related to object recognition process for a recognized object.	
Data:	
Position location	Position information of an object.
Feature[] features	List of detected Feature instances for an object
int type	Type id of an object
int id	Id of an object to match the object in library database
Methods:	
Position getLocation()	Returns the Position of an object.
Void setLocation(Position)	Sets the Position of an object to the input Position
Feature[] getFeatures()	Returns the list of Feature's of an object.
addFeature(Feature[])	Add's a new Feature to Object's Feature list
delFeature(int)	Delete's the Feature from the list with the given index.

ObjectList	
Keeps the list of recognized objects	
Data:	
Object[] objects	List of detected objects.
Methods:	
void removeObj(int id)	Removes the Object from the objects list with given index
void addObj(Object)	Adds the input Object to the list
int size()	Returns the number of Object's in the list.

ObjectTable

Keeps the list of recognized and categorized objects

Data:

Object[] objects	List of objects.
------------------	------------------

Methods:

void updateObject(Object)	Updates the Object information
---------------------------	--------------------------------

void insertObject(Object)	Adds the input Object to the list
---------------------------	-----------------------------------

Void delObject(Object)	Removes the Object from the list
------------------------	----------------------------------

Position

Holds the position information for an object

Data:

Float [] [] posinf	Position information array consisting of floating point numbers. For each corner an x and y float is included.
----------------------	--

Methods:

Void SetPos(Float [] [])	Updates the posinf data with the float array input.
----------------------------	---

Shelf

Holds all the information about the recognized shelves.

Data:

Inherits all data defined in the Object Class.

int id	Holds the id number of the shelf
--------	----------------------------------

Methods:

Inherits all methods defined in the Object Class.

int getId(void)	Returns the id of the shelf
-----------------	-----------------------------

void setId(int)	Sets the id of the shelf
-----------------	--------------------------

State	
Keeps the information of state of the application in a moment	
Data:	
int type	Integer that represents the type of the state
Methods:	
int getType()	Returns the type of the state
void setType(int)	Sets the type of the state

StateChart	
Stack of application states	
Data:	
State[] states	Array of State elements.
Methods:	
setState(State)	Adds the state to end of the list
getState(State)	Gets and the state at the end of the list
removeState(State)	Removes the state at the end of the list

States <<interface>>	
Interface provided by Graphic component for WorldModel Component	
Data:	
Bool mode	Is search mode enabled?
Int[] operation	List of operations to be done in the current state

World <<interface>>

Interface provided by Object Recognizer component for World Model component

Data:

IplImage frame	Current frame image captured from the camera.
ObjectList objects	List of recognized objects

WorldRepresentation <<interface>>

Interface provided by World Model component for Graphic component

Data:

StateChart states	Stack of current program states.
ObjectTable object	List of recognized and categorized objects

5 System Architecture

In this section a general description of the system architecture is given.

5.1 Architectural Design

From the most basic view, Bookwiser has a sequential way of operating. What it does is simply processing an image, retrieving some info from image, enriching this graphical information with some extra contextual information, combining them together in image and showing the reconstructed image to the user.

This operating style is reflected onto the design as three components of the system:

- Object recognizer to process images and create abstract object models about the library world.
- World model unit to compose a world view inside Bookwiser .
- And finally graphic unit to convert this abstract view to a graphical view for user. Functionally, world model units needs object recognition unit to operate, graphic unit needs world model unit to operate.

On the other hand, this style of abstraction -in its pure form- could not answer the performance goals of the system. To achieve timing goals which described in the section 3, it should be ensured that the system performs in predefined time limits. Performing this in a sequential flow of events is unrealistic.

Thus, three main components of Bookwiser (object recognizer unit, world model unit, graphic unit) should be designed as different threads which runs parallel and collaborates with each other. Three components performs their tasks to achieve timing goals of the system with synchronization.

Graphic unit has a higher priority than other two components since the main function of the Bookwiser is to answer the needs of user quickly. Yet it is logical to allow other two units to occupy the process time, because those two components have more processing weight.

While graphic units uses world model unit as a source of information, object recognizer unit and world model unit proceed collaboratively to create a representative picture of world. They identify the current world state one object at a time. In this way, Bookwiser can perform in its time limits. According to world state and this time limits, world model unit arranges the tasks of object recognizer unit. It can restrict the object recognizer unit with certain tasks or it can assist the object recognizer with world state information.

These three components are shown in the below component diagram:

Component Diagram

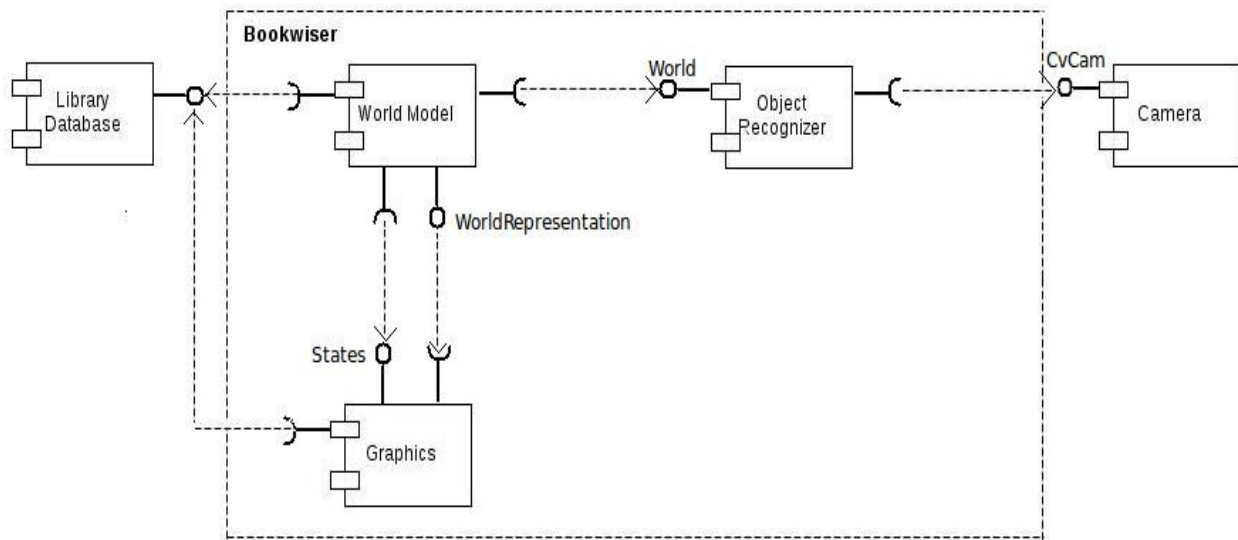


Diagram 5.1 – Component Diagram

A brief explanation of the components are given:

Object Recognizer Component:

This unit operates directly on the digital image sequences captured by the camera. It uses its own database to define objects. After every identification of a particular object-either a re-identification with tracking or discovery of a new object- it sends this information to world model unit and accordingly, it is assigned another identification task by the world model unit. Its functionality can be particularly or completely cut off according to timing constraints or user events.

World Model Component:

World model unit performs management tasks. All information gathered from the outside sources-camera, library database and the user- comes to the unit, is classified and used to produce an abstraction of world. It controls the working mechanism of object recognizer and serves as a knowledge base to the graphic unit. The world representation sub-component can be seen as a data table which arranged according to the object classifications and user inputs which defines the state of the Bookwiser. The manager sub-component is the interface of the world representation.

Graphic Component:

Graphic unit is responsible form every action which considers the user. These actions are two-way. Giving information to the user and getting information from him/her. Giving

information means displaying the camera view with augmented contextual information. Video composer is responsible for this task. Getting information is handled by input controller. User inputs are classified and sent to related components. Note that user inputs are not processed by the graphic unit. The graphic units functionality is restricted with maintaining the connection between user and the Bookwiser.

5.2 Description of Components

5.2.1 Object Recognizer Component

In this section , an explanation for the Object Recognizer Component and the related class and sequence diagrams are provided.

5.2.1.1 Processing Narrative for Object Recognizer Component

Object Recognizer Component can be described as the eye-brain channel of the Bookwiser system. All the object recognition, tracking and classification process is completed inside this component.

The component receives frames from the camera, filters the frame for a better quality image and operates on this image.

The component first finds assumptions for borders of possible objects on the scene. Using this assumptions , searches related areas of the image for SURF features. When an object is recognized, adds the object to a global recognized object list so that other components can use this available information to progress on their own duties.

Another responsibility of this component is to track already-recognized objects in the new frame. When an object is recognized in a frame, on the next frame the object's position can be changed or the object can be completely out of the scene. The component therefore decides to update the position of the object instance on the global object list or remove the object from the recognized object list.

This component has the heaviest load of the system and therefore is the most detailed component of the Bookwiser.

5.2.1.2 Interface Description for Object Recognizer Component

Generally, it can be said that the component has relations with two main interfaces.

- As an input interface, the component uses OpenCV Library's CvCam class to interact with the camera and to get sequences of frames. The detail of this interface will be given in the Detailed Design section.
- Output interface for the component is named as World. Since the component gives a description of the scene which the user is currently looking at to the system, using this interface, the component shares a global recognized object list and the current frame sent from the camera with the other components. Current frame and global object list , together implements the World interface for the use of other parts of the system.

5.2.1.3 Processing Detail for Object Recognizer Component

Object Recognizer Component describes the user's view to other components of the system.

- When a new frame is supplied by the camera , RecognitionHandler creates a sequence of filters to enhance the input image. With different FilterHandler instances, the image is enhanced for future operations.
- RecognitionHandler then calls the ObjectTracker . ObjectTracker looks at the ObjectList and if any recognized Object instances exist from the previous frame, ObjectTracker tries to find the new positions of objects in the frame.
- At this point ObjectTracker may find out that an object is no longer in the frame, than it removes the related Object instance from the ObjectList . If not so, then ObjectTracker updates the position information of the related Object instance in the ObjectList.
- When tracking operation is done, RecognitionHandler starts for a search of new objects in the frame.
- First BoundaryFinder is called to find assumptions for borders of book, shelf and librarian objects. BoundaryFinder may decide to remove some candidate objects it has found if it detects that the candidate object is a duplicate of one of already-recognized objects.
- The list of candidate objects than is used by ObjectRecognizer to find if they really are one of the objects that in the object recognition database.
- If it decides that the candidate object is one of the known objects, than it prepares an Object instance and adds it to ObjectList . If thats not the situation than the candidate object is deleted.
- After then RecognitionHandler asks for a new frame from the camera and the whole process starts again.

A more detailed explanation of the flow of the component and the details for the methods and algorithms of the component are left for the Detailed Design section.

The relationships between mentioned classes can be seen in the Class Diagram for Object Recognizer Component below.

Class Diagram - Object Recognizer Component

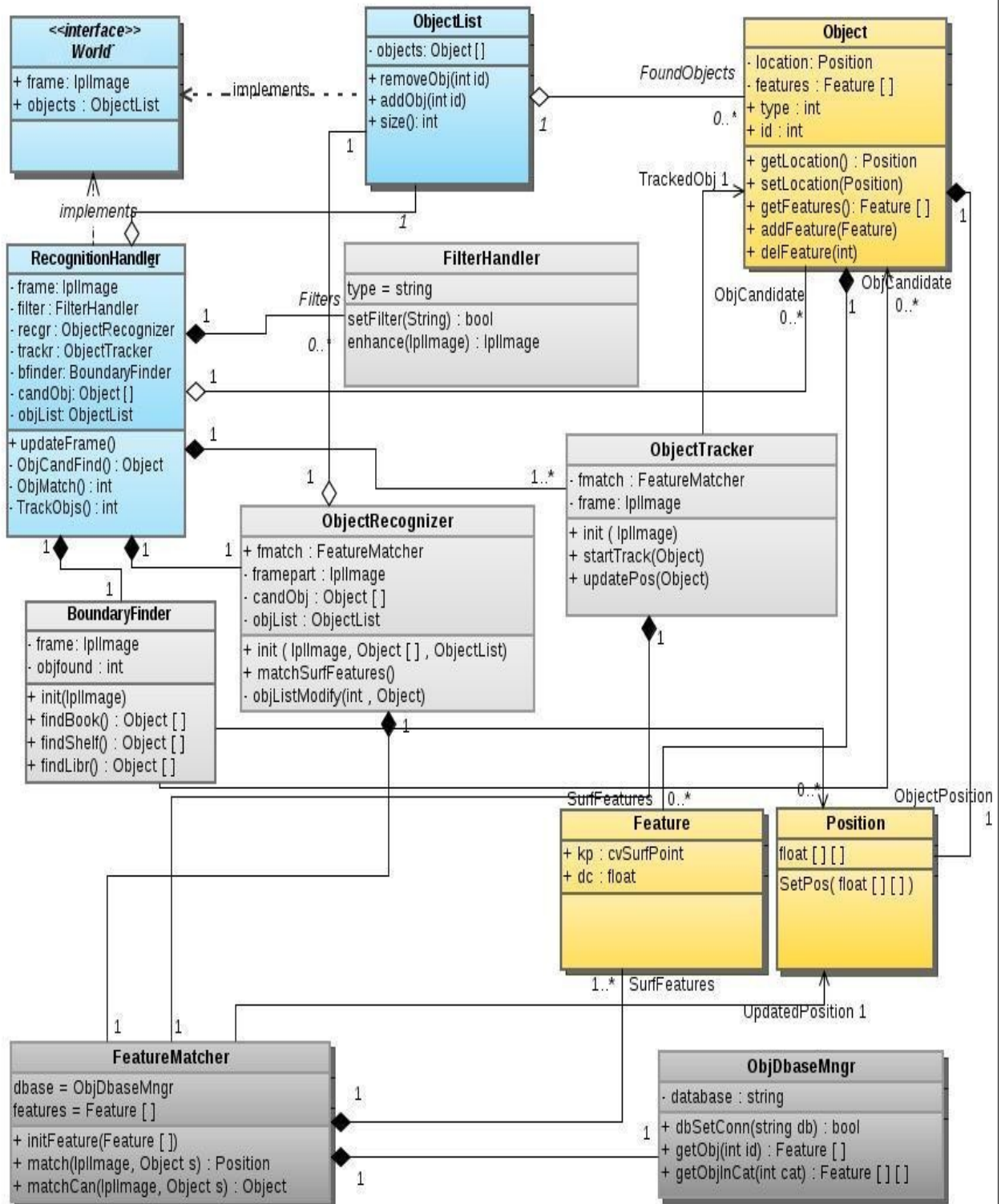


Diagram 5.2 – Class Diagram

5.2.1.4 Dynamic Behavior of Object Recognizer Component

Image Filtering:

When the RecognitionHandler gets a new frame from the camera , first image filtering is done.

- RecognitionHandler creates a FilterHandler instance and initializes it with a call to `setFilter()` method.
- If the initialization is successful then the frame is sent to the FilterHandler via the `enhance(IplImage)` method.
- After the enhancement operations image is returned back to the RecognitionHandler.
- The sequence is repeated for all filters defined by RecognitionHandler

Detailed information of filters that will be used in this part will be given in the Detailed Design section.

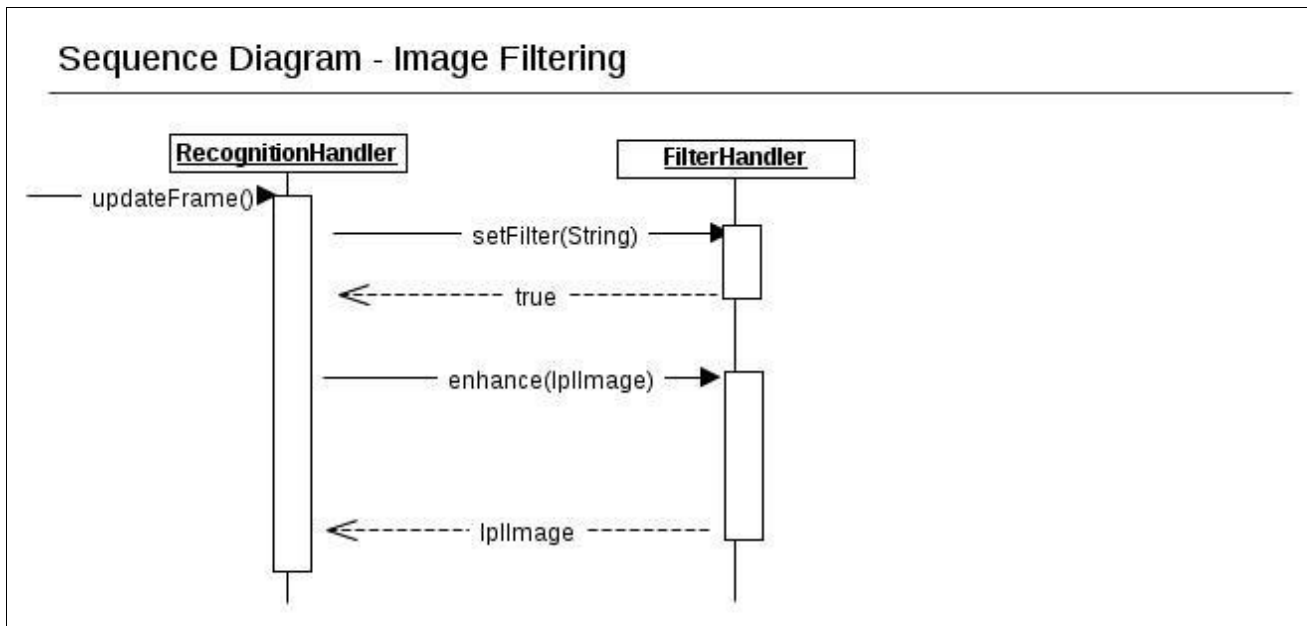


Diagram 5.3 – Sequence Diagram

Object Tracking :

After the filtering operations are done, object tracking is activated.

- RecognitionHandler , for each already-recognized object initializes an ObjectTracker instance with init(IplImage) method.
- After the initialization is done, startTrack(Object) starts the tracking operations. With getFeatures() method.
- ObjectTracker gets features from the related Object instance.
- ObjectTracker initializes a FeatureMatcher instance with a call to initFeature(Feature[]) method
- During initialization FeatureMatcher initializes a ObjDbaseMngr via a dbSetConn(String) call for object database operations.
- With a call to match(IplImage,Object) function, FeatureMatcher requests object features from the ObjDbaseMngr with getObj(int) and returned features are used to identify the position and orientation of the object.
- If it is not null , the new returned Position is used to update the Object instance's Position element with setLocation(Position) call.
- If the returned Position is null then the Object is deleted from the ObjectList with a call to removeObj(int).

Tracking algorithms that used in this part will be explained in the Detailed Design section.

Sequence Diagram - Object Tracking

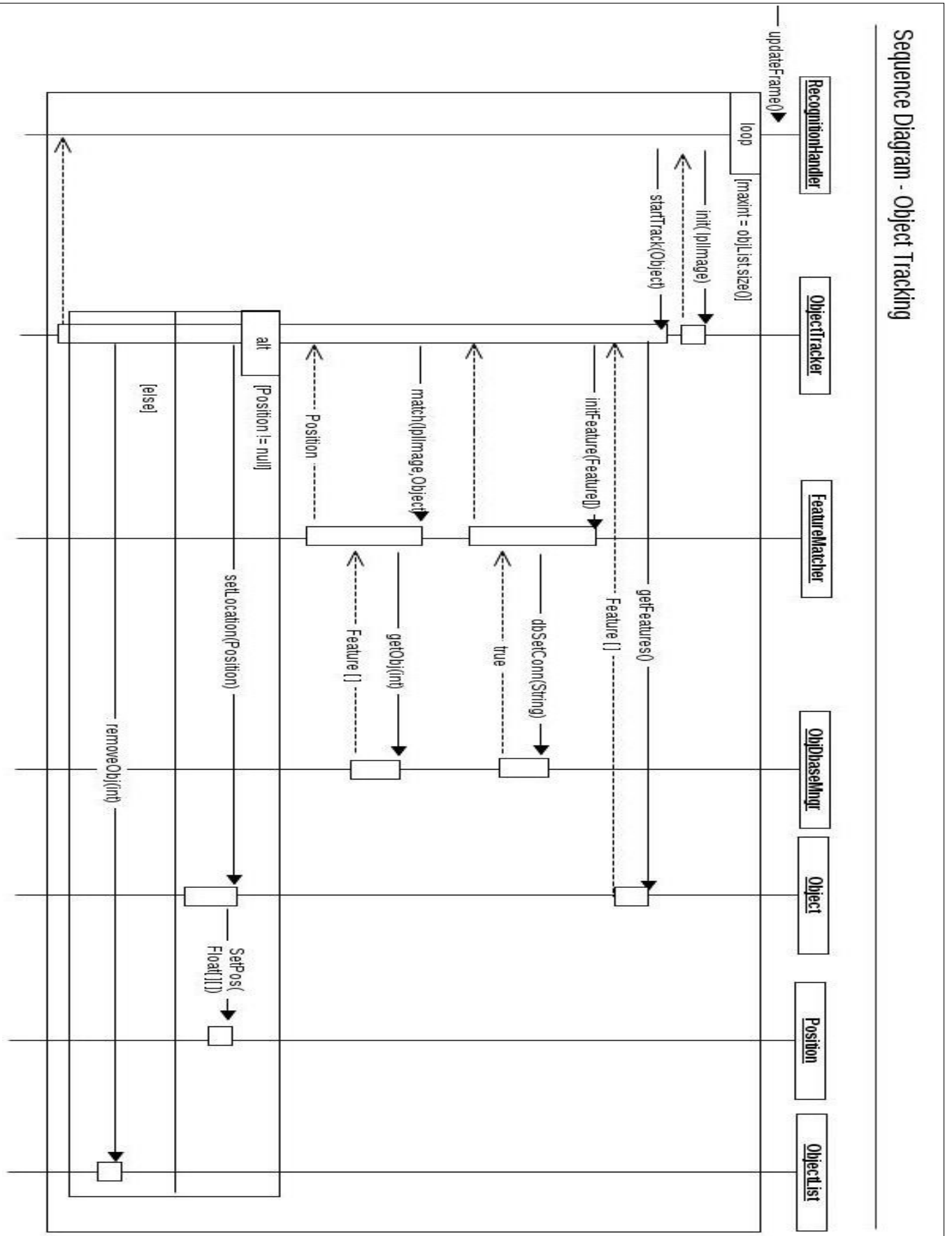


Diagram 5.4 – Sequence Diagram

Candidate Object Boundary Finding:

After Object Tracking is done, a search for new objects is started.

- RecognitionHandler initializes BoundaryFinder with `init(IplImage)` call.
- There are 3 methods that starts BoundaryFinder to search for candidate objects : `FindBook()` , `FindShelf()` and `FindLibr()` . The mentioned methods search for possible boundaries for books, shelves and librarians respectively and each method applies a different algorithm for possible boundaries of objects.
- For each found possible boundary a candidate Object is set .
- A list of found candidate objects are returned to the RecognitionHandler.

Algorithms for finding possible boundaries used in this part will be explained in the Detailed Design section.

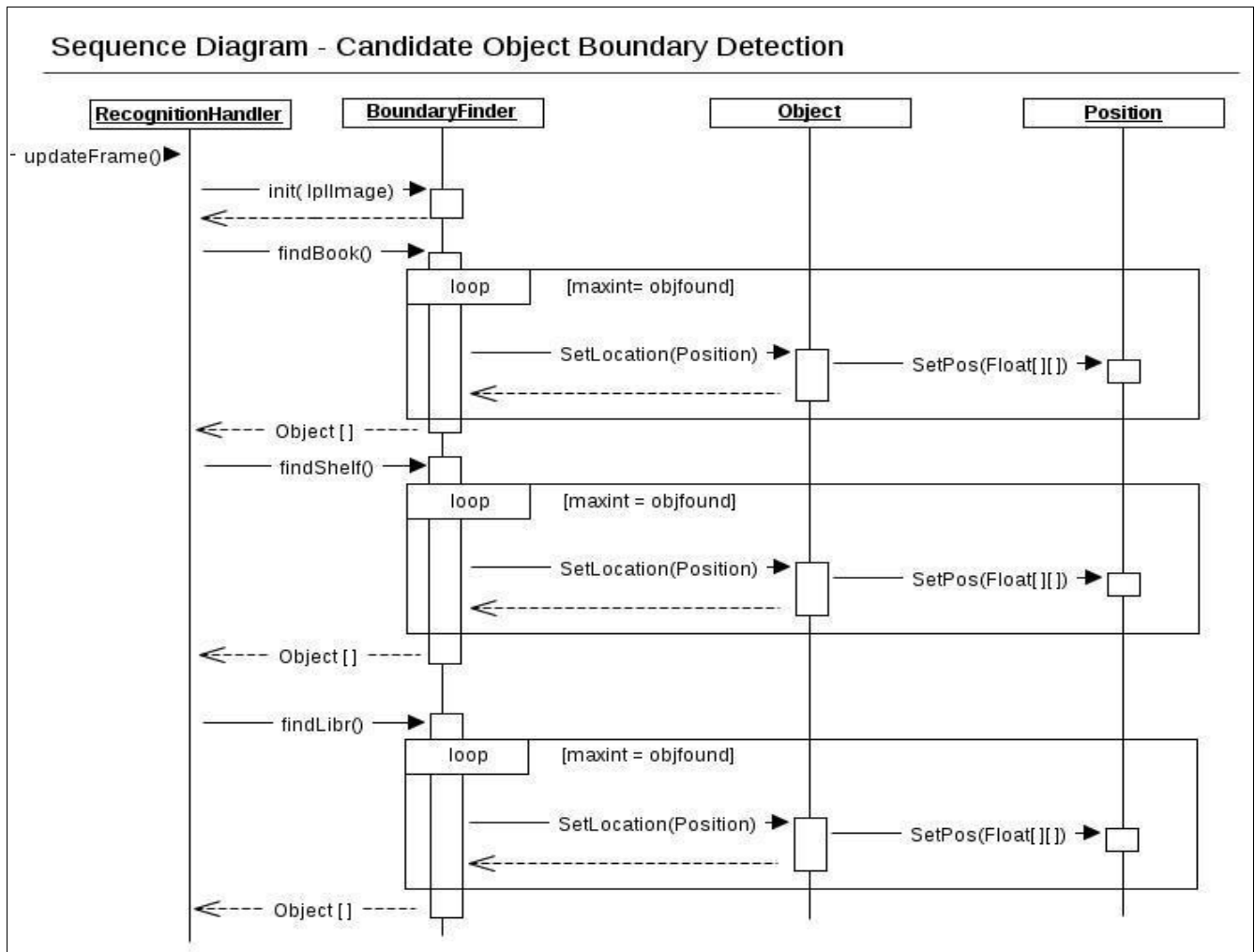


Diagram 5.5 – Sequence Diagram

Object Recognition:

- After candidate Objects are found, ObjectRecognizer is initialized by RecognitionHandler with a call to the method `init(IplImage, Object[], ObjectList)` .
- `MatchSurfFeatures()` method for each candidate Object starts a recognition process.
- ObjectRecognizer initializes a FeatureMatcher for each candidate object with `initFeature(Feature[])` call.
- FeatureMatcher sets a database manager, ObjDbaseMngr with `dbSetConn(string)` call.
- When these operations are done successfully , `matchCan(IplImage, Object)` function makes the FeatureMatcher compare features of the candidate Object with features in the object database after getting them from database with a call to the `getObjInCat(int)` method of ObjDbaseMngr .
- The matched candidate Object, is filled with known features with the `addFeature(Feature)` call and its Position is set via `SetLocation(Position)` method.
- After the candidate Object is transformed into an Object the Object is sent back to ObjectRecognizer .
- ObjectRecognizer adds the Object to ObjectList with `addObject(int)` call.
- If the candidate Object is not matched with any object records in the database, the candidate Object is deleted.

Feature matching algorithms will be explained in detail in the Detailed Design section.

Sequence Diagram - Object Recognition

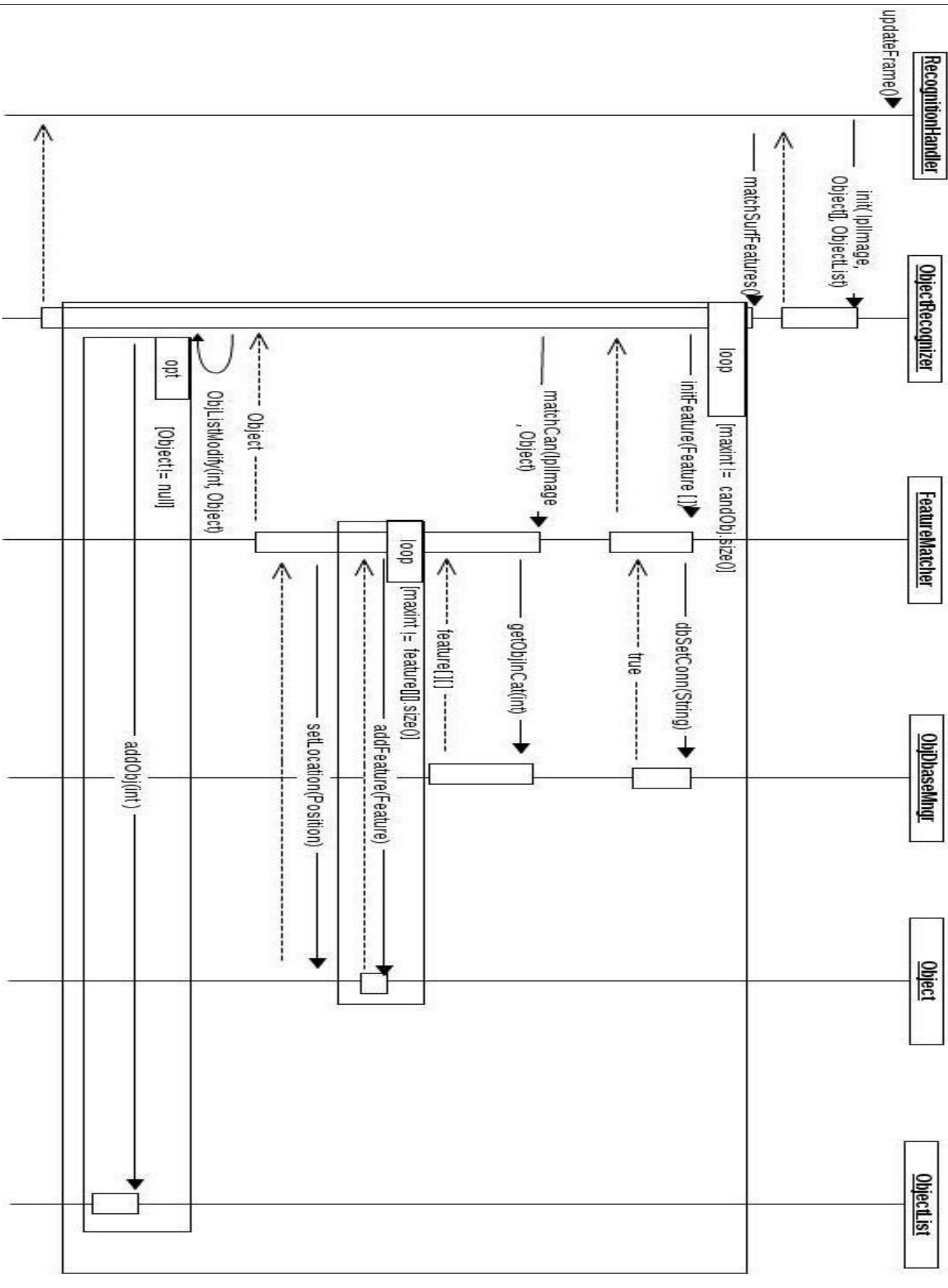


Diagram 5.6 – Sequence Diagram

5.2.2 World Model Component

In this section , an explanation for the World Model Component and the related class and sequence diagrams are provided.

5.2.2.1 Processing Narrative for World Model Component

World Model Unit is the main system component in the Bookwiser. World Model Unit is the system that collects all information from all components and sum up them to the user by the Graphic Unit because it connects the Object Recognizer Unit and Graphic Unit together.

World Model Unit is mainly the spine of the Bookwiser that all the components and data flow are controlled in this unit. All the information that flows in the application from the user to the user again should interact with this component.

The object information comes from the Object Recognizer Component. All the information about the detected objects like books, shelves and the librarian, are controlled in this component. The objects detected by the Object Recognizer Component, comes to the World Model Component. Now, this information has no meaning to the application because the recognized objects should be processed and filled with related information. For this purpose, World Model Component retrieves context information from the Library Database. Using the existing information in the database, World Model Component fills in the information fields of recognized objects . The object detection part is actually completed in this unit in this manner. After these operations , Bookwiser will show the related information to the user by Graphic Component.

On the other hand this component also keeps track of user-intercation based state changes. Different states of different user choices are stored and processed via this unit. According to different inputs from the user, the output that will be sent to the Graphic Component is manipulated in this component.

As a conclusion it can be said that this component is the brain of the Bookwiser system.

5.2.2.2 Interface Description for World Model Component

The interfaces in the World Model Component is not related with the user directly. Instead the two input interfaces of the component interacts with the Object Recognizer Component and the Graphic Component , and the output interface only interacts with the Graphic Component.

- The interface “World” is the input interface for interactions with Object Recognizer Component.
- The interface “States” is the input interface for interactions with Graphic Component.
- Moreover, “WorldRepresentation” is the output interface to the Graphic Component.

5.2.2.3 Processing Detail for World Model Component

An algorithmic description for the World Model Component as follows:

- 1) The information comes from the Object Recognizer Component to the World Model Manager Component.
- 2) After, “setConn” function is called and returned by true, Context Information coming from the Library Database is brought to the ObjectManager class in order to be combined with the information coming from the Object Recognizer Component.
- 3) Step2 is repeated whenever the information coming from Object Recognizer Component is updated.
- 4) Mode updates information comes from the Graphic Component to the World Model Manager Component.
- 5) Step4 is repeated when the user changes the state of the system or select one of the modes by the input controller such as Search Mode and FreeWalk Mode.
- 6) When a book, shelf or librarian is chosen by the user, incoming states from the Graphic Component are stored in the state bank of the component.
- 7) Output of the component is manipulated by the changes on the states.

The detailed explanation of the methods used for mentioned operations is left for the Detailed Design section.

Class Diagram - World Model Component

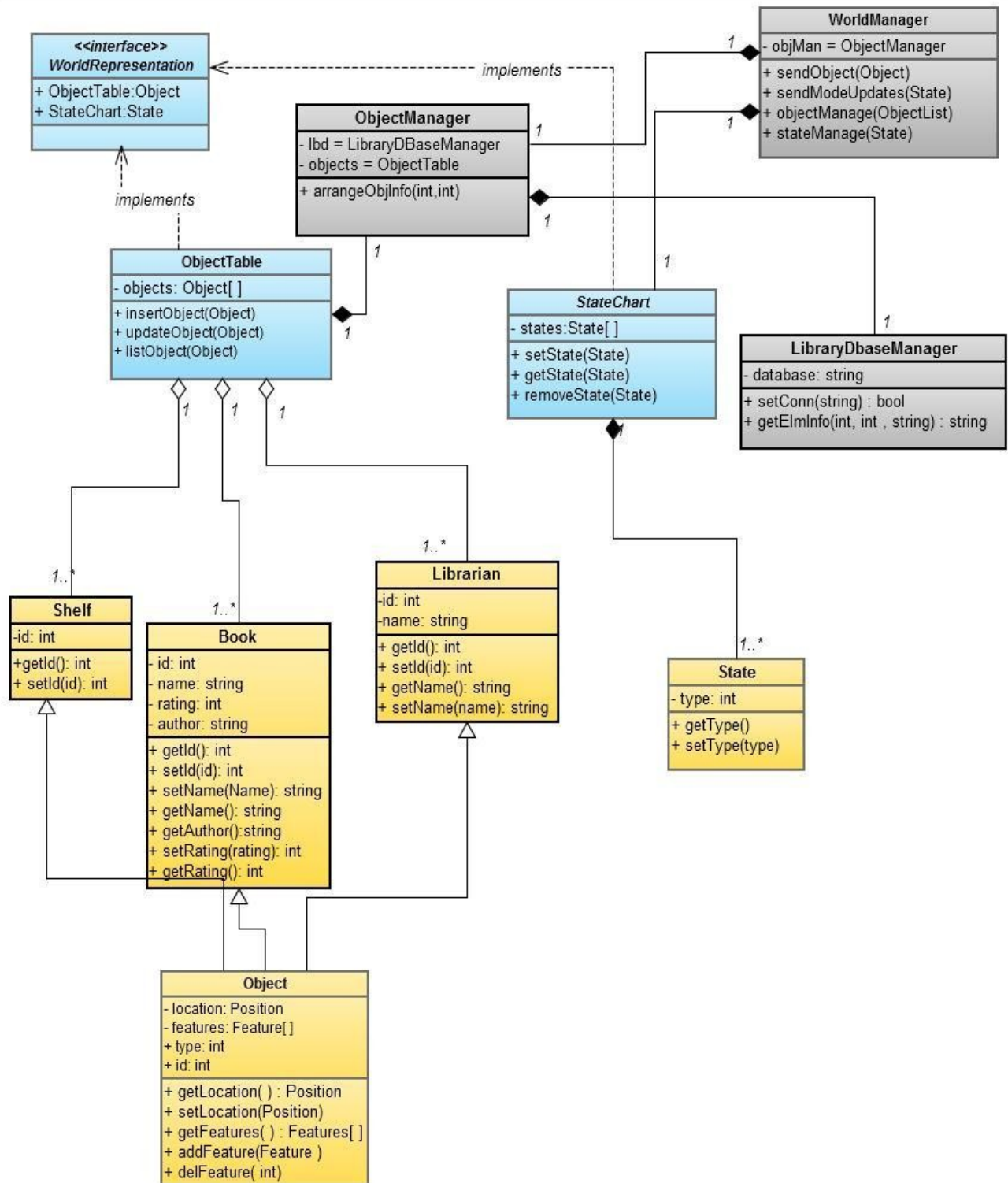


Diagram 5.7 – Class Diagram

5.2.2.4 Dynamic Behavior of World Model Component

In the World Model Unit, the interactions between the components is explained not in a very detailed manner, but in an initial detailed design manner.

After the information coming from the Object Recognizer Unit is processed firstly in the WorldManager class. WorldManager class interacts with two basic classes namely: ObjectManager and StateChart.

Object Management:

- WorldManager interacts with two basic classes, namely: ObjectManager and StateChart. These classes implements an interface called WorldRepresentation.
- ObjectManager interacts with the library database manager , LibraryDBaseManager and ObjectTable .
- The interaction between the ObjectManager and ObjectTable is processed after the interaction between ObjectManager and LibraryDBaseManager is established. This sequence is clearly shown in the diagram.
- ObjectManager 's interaction with the ObjectTable is established, the specific objects comes to the interaction. Since our objects are mainly books, shelves or librarians, Book, Shelf and Librarian classes are inherited from the Object class.
- ObjectTable gathers information from the Book, Shelf and Librarian instances by using their specific methods.

Sequence Diagram - Object Management

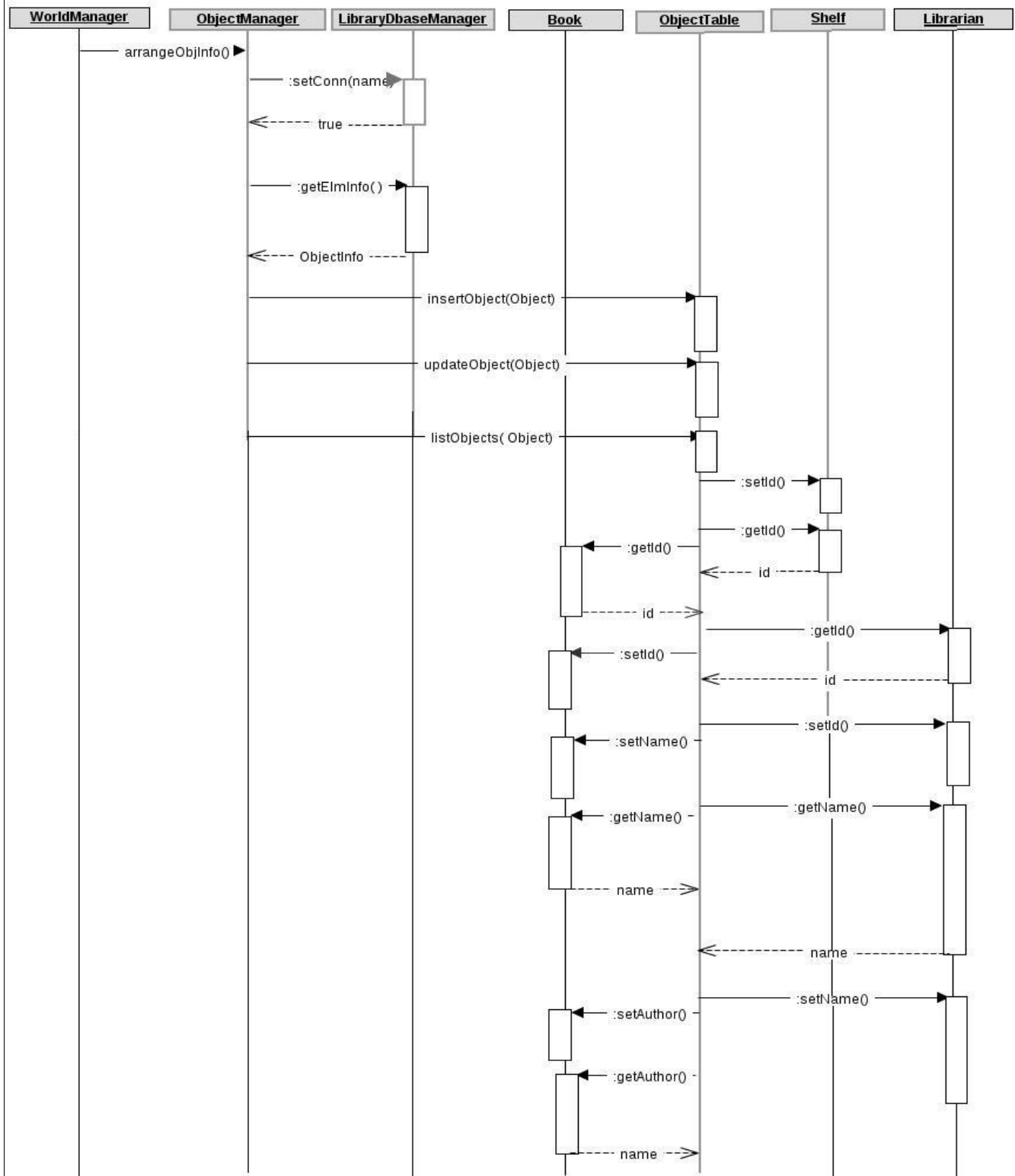


Diagram 5.8 – Sequence Diagram

State Management:

Now, it's time for the second class that the WorldManager interacts with, namely, StateChart. The following explanations and related sequence diagram as follows:

- WorldManager Unit interacts directly StateChart class.
- A State instance is created in order to answer the state changes in the Bookwiser.
- When a State is expired, the state is removed from the StateChart via `removeState()` call.

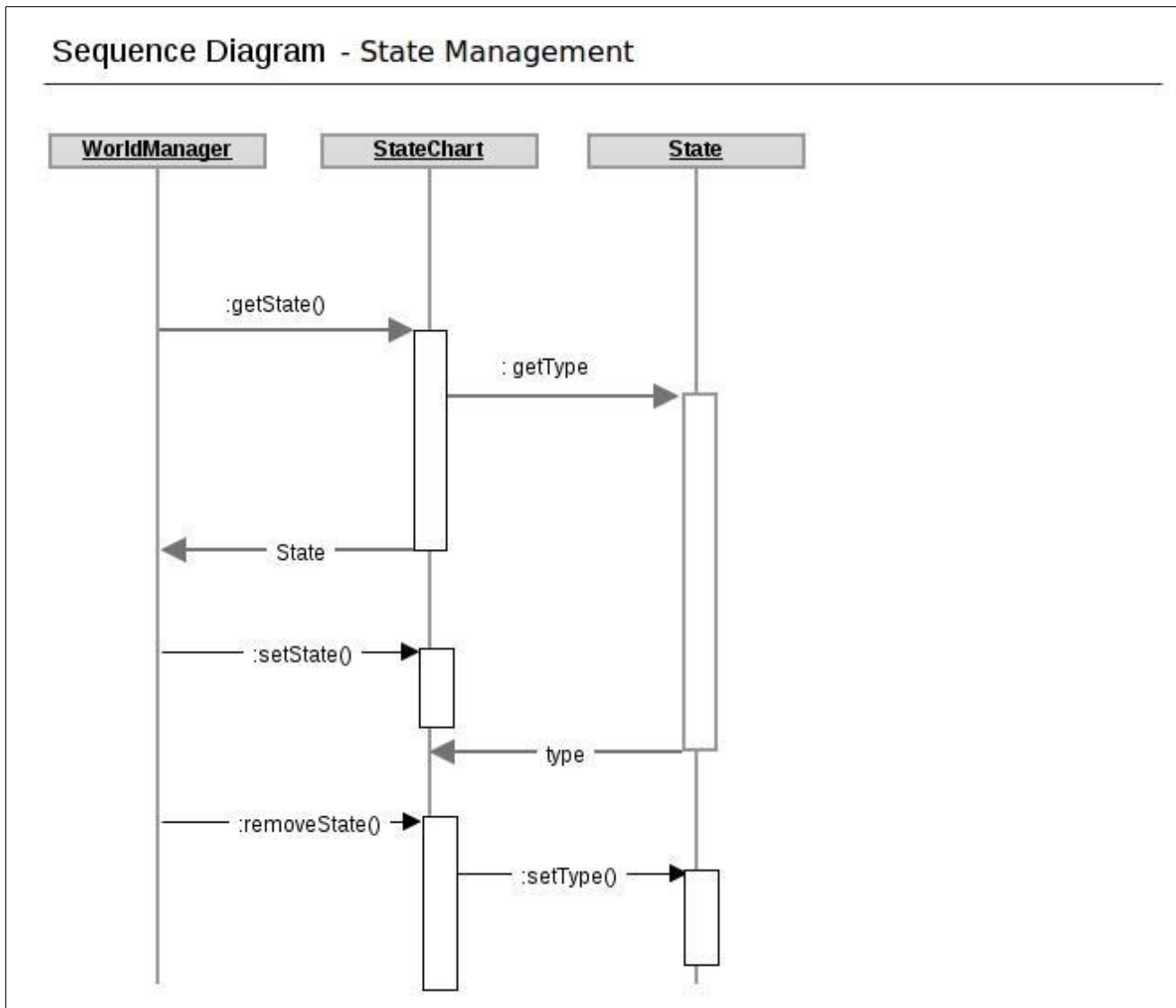


Diagram 5.9 – Sequence Diagram

5.2.3 Graphic Component

In this section , an explanation for the Graphic Component and the related class and sequence diagrams are provided.

5.2.3.1 Processing Narrative for Graphic Component

This is the third component of Bookwiser Software Project. Graphic Unit is the component that directly interacts with the user. This component does not process on anything, but just controls the button clicks, the user interface and the functionalities that directly lies between the user and the system without any calculation like updating the rate of some book or reserving some book.

Graphic Unit consists of three parts; video composer, input controller and graphical user interface, respectively.

5.2.3.2 Interface Description for Graphic Component

The interfaces that Graphic Component has is with the second main component of Bookwiser System, World Model Unit and with the Library Database.

- The Video Composer subcomponent of Graphic Component interacts with the World Model Component via WorldRepresentation interface as an input interface. This subcomponent combines the contextual and visual object information taken from World Model Component and the video stream captured by the camera. So, it must interact with the second main component.
- The second interface also is with second main component and is an output interface called States. Since handling the buttons is one of the missions of Graphic Component, also sending the states for different user choices to the World Model Component is its duty.
- The last interface that the Graphic Component deals with is with the library database. When there is a need of an direct on the update the Graphic Unit handles it. The updates can be reserving, unreserving or rating a book.

5.2.3.3 Graphic Unit Processing Details

Video Composer:

Video composer is the simplest section of the Graphic Unit component. The Video Composer unit combines the captured video stream by the camera with the object information coming from the World Model Unit. The combined video is sent to the GUI to be translated to the user.

Input Controller:

Input controller is based on two subunits: Screen Based Action Handler and Record Based Action Handler, namely.

Screen Based Action Handler:

Screen Based Action Handler subunit is responsible for the button-click handles and mode changes.

There are two button-click handles concerning the screen information: displaying detailed information for a selected shelf and displaying detailed information for a selected book. World Model Unit is informed about these button-clicks to process the objects as required. The mode changes also are handled in this subunit. Whenever the mode is changed (the mode can be free-walk mode or search mode) the World Model Unit is informed.

Record Based Action Handler:

Record Based Action Handler is related to the direct database updates, which there is two of them : reserving a book and rating a book.

The Record Based Action Handler subunit directly connects to the library database to update the rate or the reservation information of some book according to the information coming from user.

Graphical User Interface:

Graphical User Interface is the subcomponent of Graphical Unit that directly interacts with the user. It directly deals with the user's demands and process according to those.

Also, this subcomponent takes the composed video coming from the Video Composer Unit and displays that to the user.

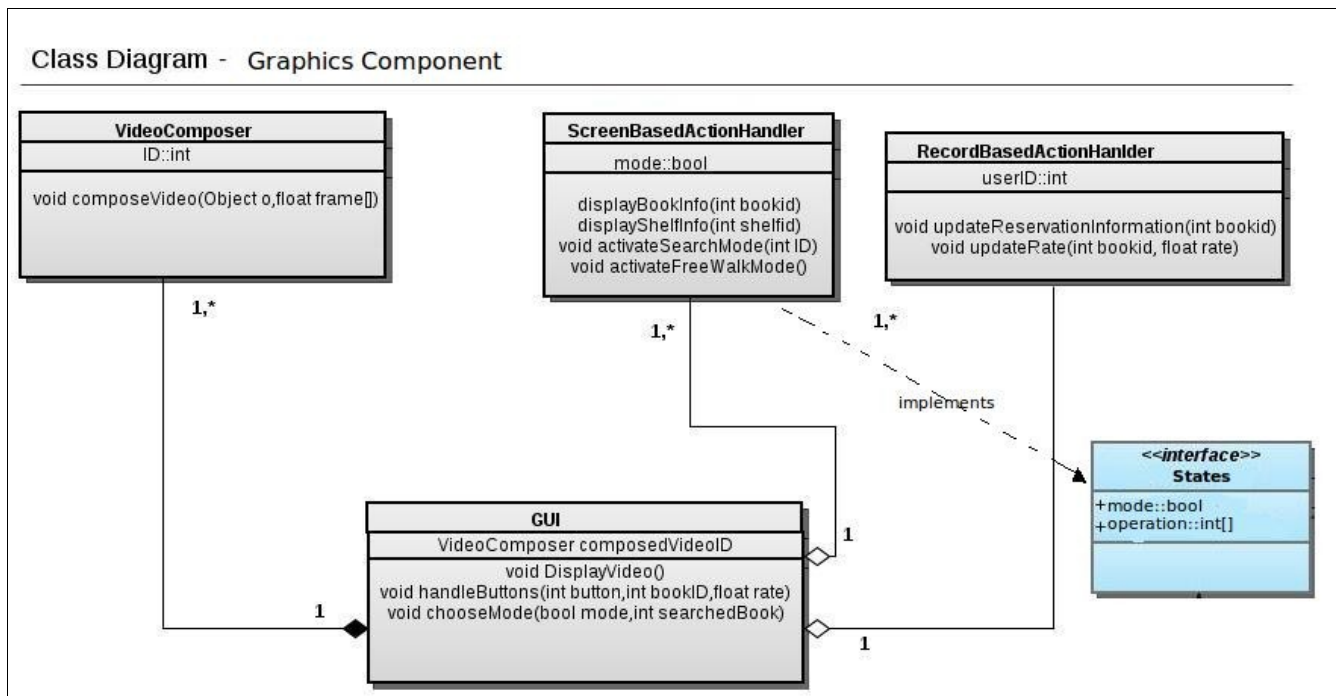


Diagram 5.10 – Class Diagram

5.2.3.4 Dynamic Behavior of Graphic Component

Choosing Objects:

- `handleButtons(1,shelfID)` method of GUI instance is calling `displayShelfInfo(shelfID)` method of ScreenBasedActionHandler instance which is going to request the information about the given shelf.
- `handleButtons(2,bookID)` method of GUI instance is calling `displayBookInfo(bookID)` method of ScreenBasedActionHandler instance which is going to request the information about the given shelf.

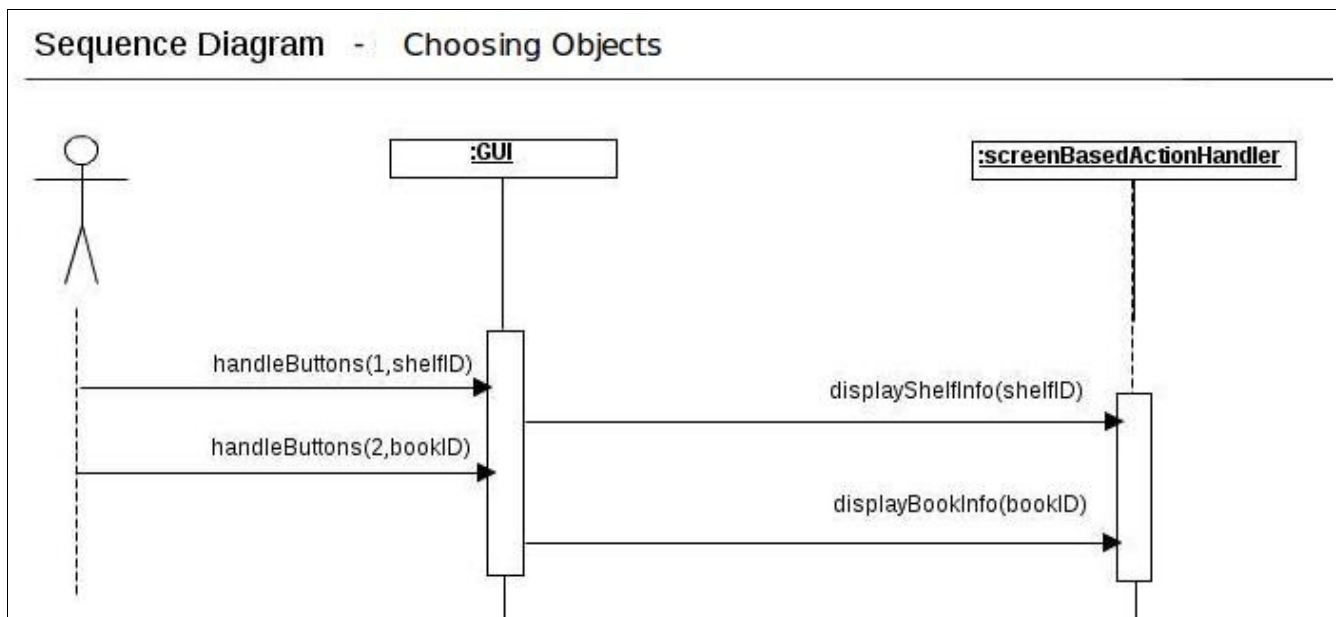


Diagram 5.11 – Sequence Diagram

Choosing Modes:

- `chooseMode(true,ID)` method of GUI instance is calling `activateSearchMode (ID)` method of ScreenBasedActionHandler instance which is going to request an alert when the location of the book with the given ID is in the vision of the user. The system mode is in search mode.
- `chooseMode(false,ID)` method of GUI instance is calling `activateFreeWalkMode ()` method of ScreenBasedActionHandler instance which is going to change the mode of the system to free-walk mode. In this situation the argument 'ID' is not used, is ignored.

Sequence Diagram - Choosing Modes

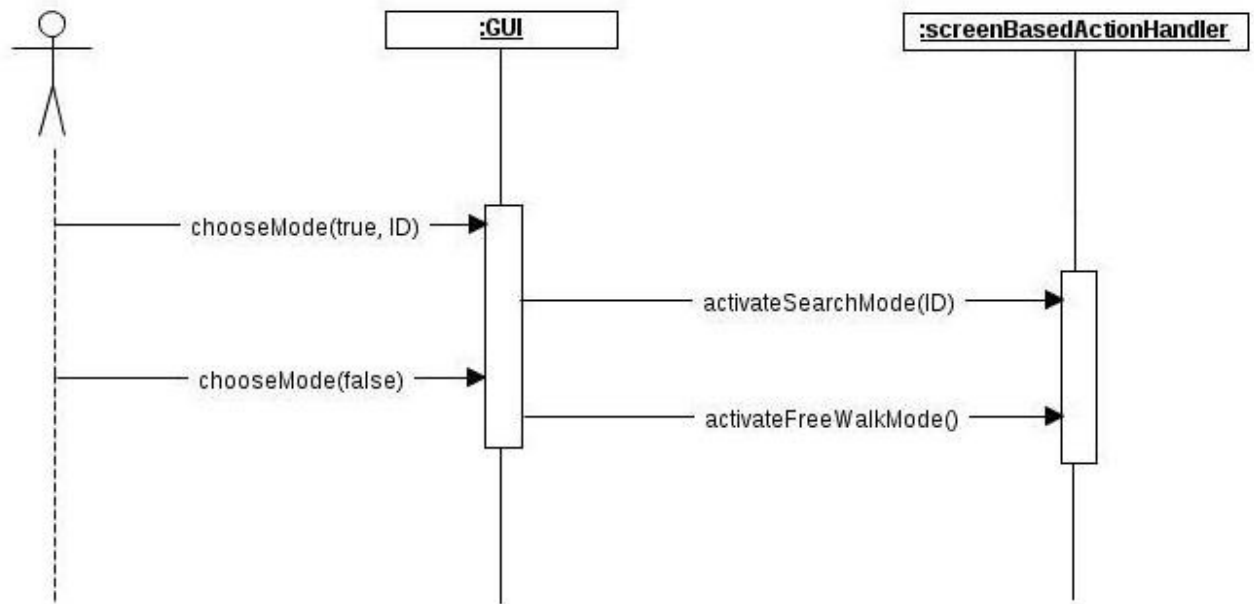


Diagram 5.12 – Sequence Diagram

Record Updates:

- handleButtons(3,bookID) method of GUI instance is calling updateReservationInformation (bookID) method of RecordBasedActionHandler which is going to update the reservation possession of the book with the given bookID.
- handleButtons(3,bookID,9,5) method of GUI instance is calling updateRate (bookID,9.5) method of RecordBasedActionHandler which is going to update the rate of the book with the given bookID and with the given rate, 9.5 in this sequence diagram.

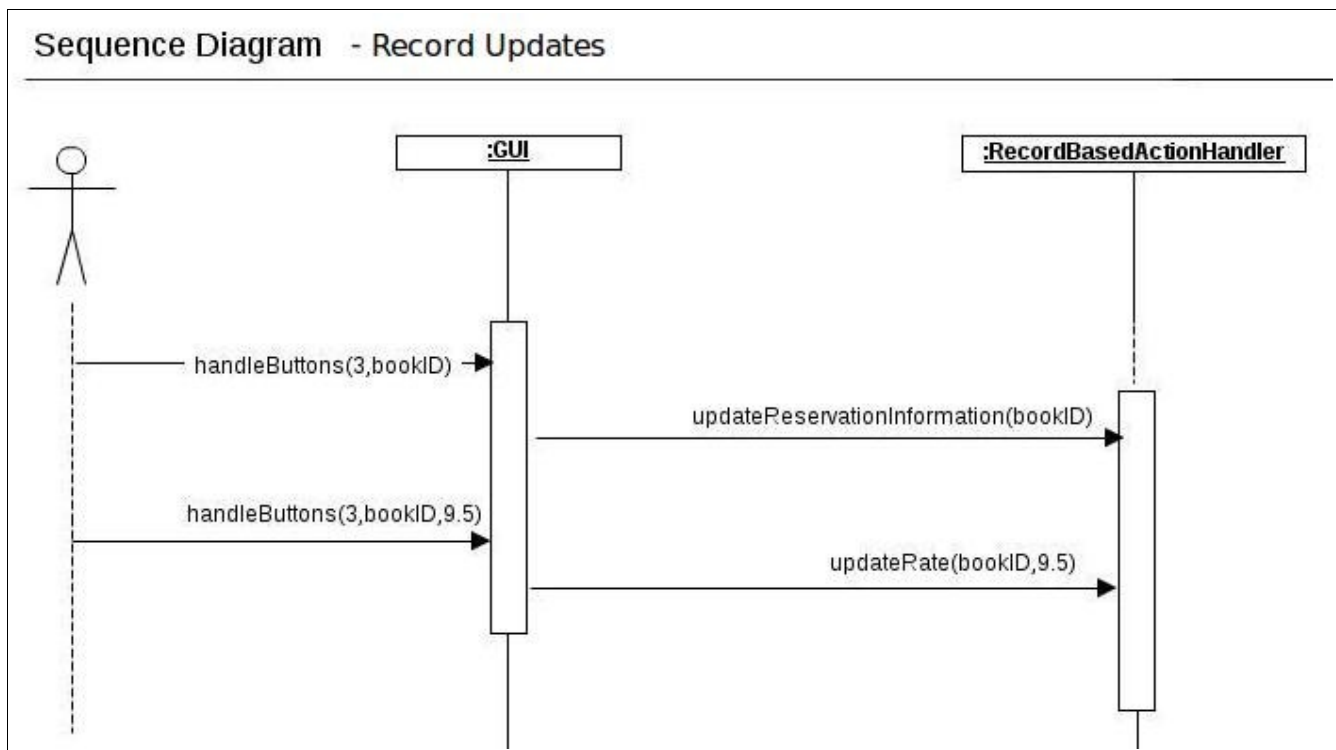


Diagram 5.13 – Sequence Diagram

5.3 Design Rationale

The abstraction of three components is directly arisen from the need to handle three external elements which system interacts: camera, user and library database. Object recognizer and graphic unit are exactly designed for handling two of them.

World model unit on the other hand is not designed for handling library database. Although it use library database to get contextual information, this is only a realization of its function. The idea that creating a world model for library reduces complexity of the system greatly and provides a basis for all sub-components of the system when they perform their tasks according to world state. Upon this architecture, it is possible to improve Bookwiser's functionality and enhancing its capability with extra services.

6 User Interface Design

In this section user interface design details are provided.

6.1 Overview of User Interface

Bookwiser's user interface is the most critical unit from user's perspective. All the interaction between Bookwiser and user is done through user interface. Main job of user interface is showing the user the video stream enhanced with extra information. It is also responsible for capturing user actions such as mode selection, book selection etc.

Since the nature of Bookwiser as a visual product, the interface has to be designed to make product attractive, easy to use and functional. Functionality is exclusively important to Bookwiser can perform its assisting tasks fullfilly. In that manner, interface screen modes are designed based on user requirements. There are five basic screen classes where all screens are grouped. Five classes are created according to the similarity between screen environments. All sub-screens which belong to same class can be seen as same screen with different options. These five classes are

Log-in screen: These screen class includes initial screen modes. User connects ,or chooses not to connect, Bookwiser system from these screens. Video display is off. Four states of these screen classes are

- Log-in
- Log-out
- Register
- Unregister

Library Tour screen: After connecting the system, user is directly passed to the library tour. Video display shall be started to be shown. In these screens Bookwiser shall try to find objects in the environment and give notification to the user about these objects. A very small amount of information about context of objects will be shown to the user. Six states of these screen classes are

- Free-walk mode
- Search mode
- Librarian recognition enable
- Recognizing shelves
- Recognizing books
- Recognizing librarian

Information screen: These are the screens where different types of contextual information and some extra functionality are provided about particular object which user chooses. Five states of these screen classes are

- Request shelf info
- Book selected
- Rate the book

Detailed book info
Detailed author info

Reservation screen: This is a special class of screens which is only designed for reservation transactions. The reason these screens are apart from information screens is that the user will have a reservation profile and shall be able to edit his/her reservations easily. There is no multiple states in reservation screen in a visual manner. However, there are three actions all of which can be performed by clicking particular buttons:

- Reserve the book
- Extend a reservation
- Cancel a reservation

Help screen: This is also a special class where the users demanding for help are directed. Help topics shall be displayed and user shall be able to select a topic.

A state diagram for entire user interface is below. Sub-screens and transitions between them are also included in the diagram:

select log-out sful

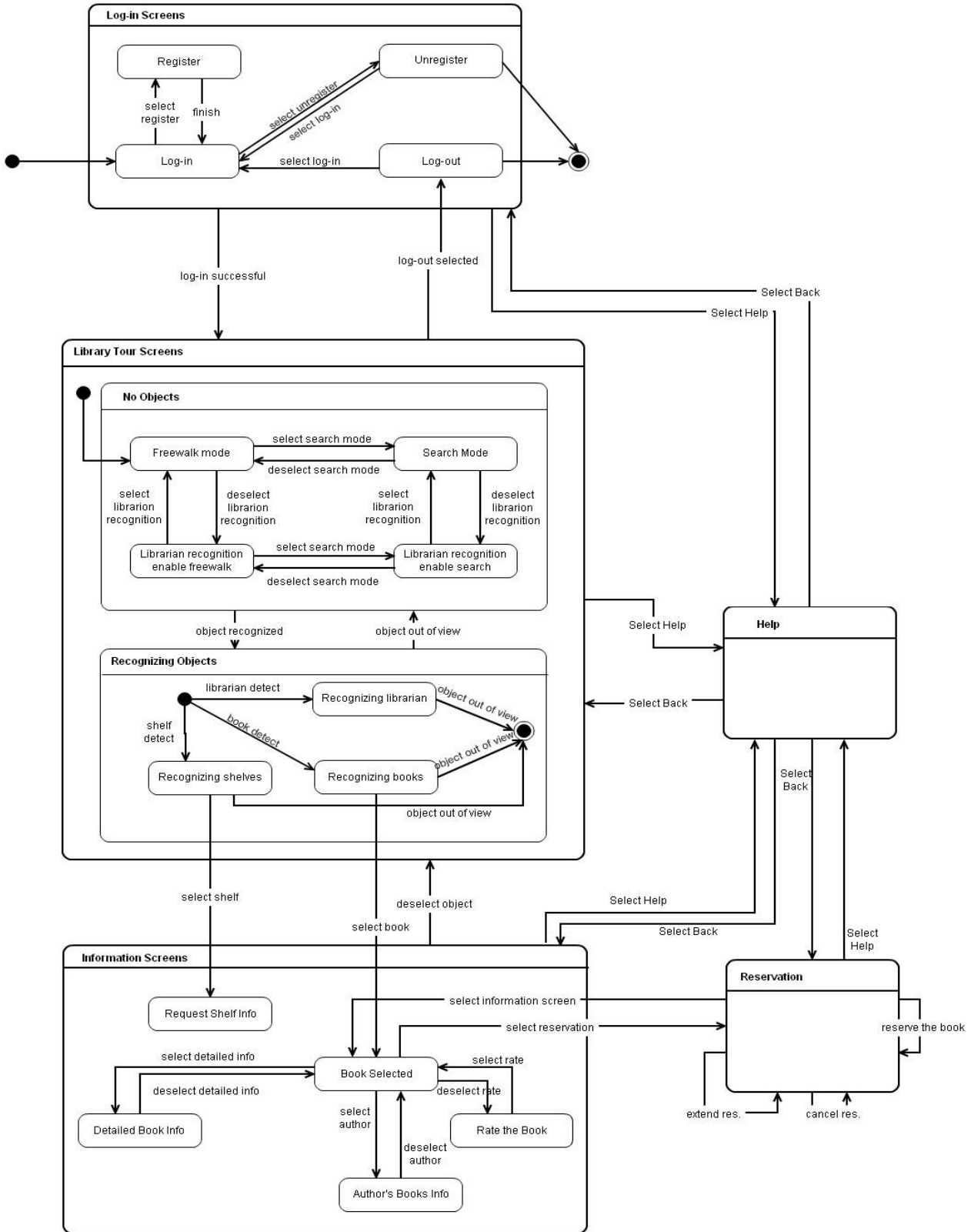


Diagram 6.1 – State Diagram for User Interface

6.2 Screen Images

Screen shots of prototype user interface are given in this section:

Log-in:

BOOKWISER

Log-in

Username

Password

Remember me

Log-out:

BOOKWISER

Log-out

Log-out is successful

Return to log-in screen

Turn off Bookwiser

Help

Register:

BOOKWISER

Registration

Please enter following information:

Name

Surname

Date of Birth

E-mail

Nationality

Address

Phone Number

Preferred Language

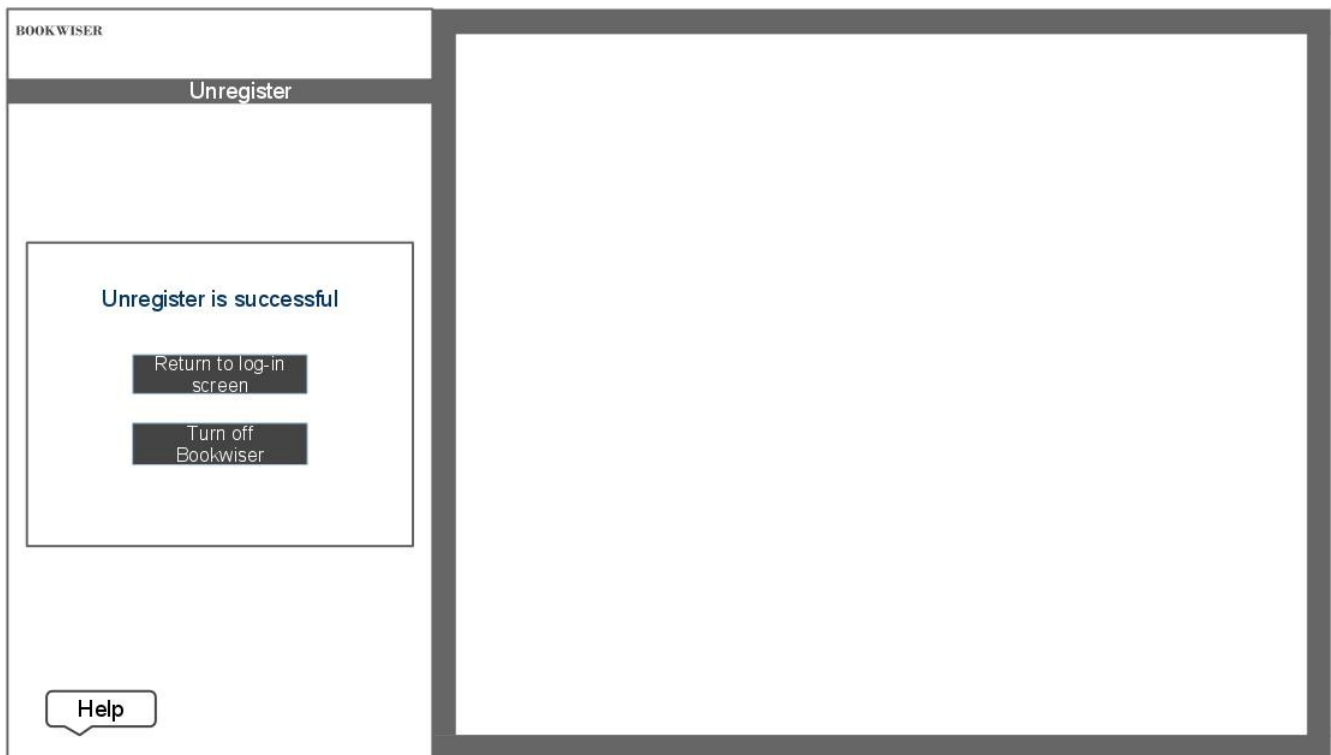
Password

Re-type Password

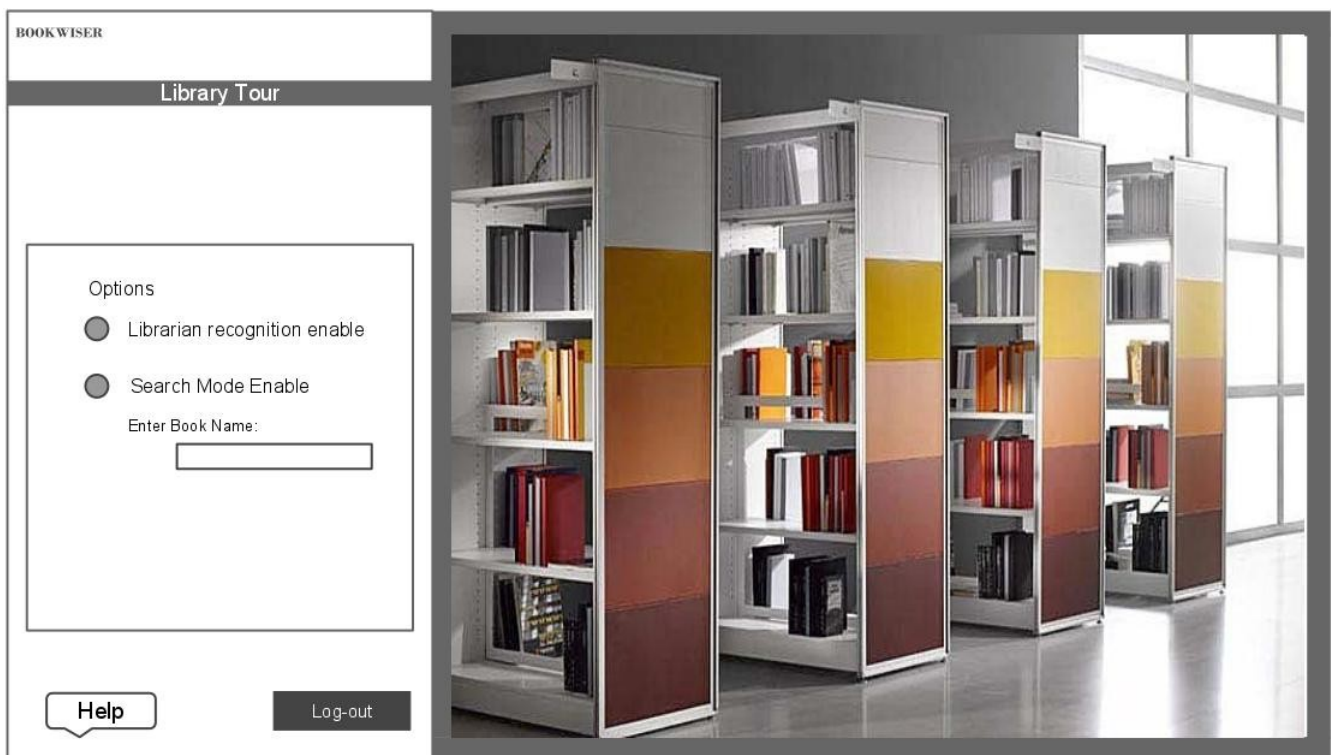
FINISH

Help

Unregister:



Freewalk Mode:



Search Mode:

BOOKWISER

Library Tour


Options

Librarian recognition enable

Search Mode Enable

Enter Book Name:

Help Log-out



Librarian Recognition Enable:

BOOKWISER

Library Tour


Options

Librarian recognition enable

Search Mode Enable

Enter Book Name:

Help Log-out



Recognizing Shelves:

BOOKWISER

Library Tour


Options

- Librarian recognition enable
- Search Mode Enable

Enter Book Name:

! Shelves are detected
Please choose one to get more information...

Help Log-out



Recognizing Books:

BOOKWISER

Library Tour


Options

- Librarian recognition enable
- Search Mode Enable

Enter Book Name:

! Books are detected
Please choose one to get more information...

Help Log-out



Recognizing Librarian:


BOOKWISER

Library Tour

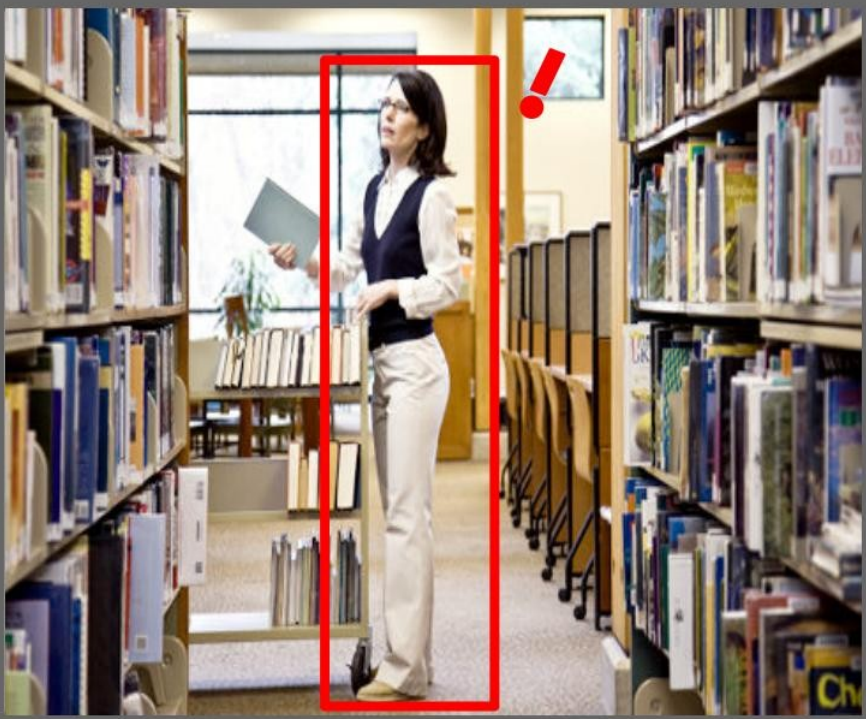
Options

- Librarian recognition enable
- Search Mode Enable

Enter Book Name:

 Librarian is detected

Help Log-out



Request Shelf Info:

BOOKWISER

Shelf Information

Shelf Information


Genre: Science Fiction

Popular Books: Foundation 1984

Number of Books: 217

Deselect

Help Log-out



Book Selected:

BOOKWISER

Book Information

★★★★★

PICASSO
by Batuhan Karagöz

Ballmer Peak Yayınevi

1967

Golden Edition


Abstract
This book of Karagöz is ...

Rate the book

Get detailed information

Reserve the book

Help Log-out



Rate the Book:

BOOKWISER

Book Information

★★★★★

PICASSO
by Batuhan Karagöz

Ballmer Peak Yayınevi

1967

Golden Edition


Abstract
This book of Karagöz is ...

Rate the book

Get detailed information

Reserve the book


Help Log-out





Detailed Book Info:


BOOKWISER

Book Information


★★★★★
PICASSO
by Batuhan Karagöz 
Ballmer Peak Yayınevi
1967
Golden Edition
Abstract
This book of Karagoz is ...

 Rate the book

 Get detailed information

 Reserve the book

[Help](#) [Log-out](#)



Detailed Book Information


Contextual information
sent from library
database...


Deselect


Detailed Author Info:


BOOKWISER

Book Information


★★★★★
PICASSO
by Batuhan Karagöz 
Ballmer Peak Yayınevi
1967
Golden Edition
Abstract
This book of Karagoz is ...

 Rate the book

 Get detailed information

 Reserve the book

[Help](#) [Log-out](#)



Other Books of the Author

List of publishings of Batuhan
Karagöz:

Yemende 4 Sene
Aborjinlerle Yaşamak
Marsın Bozkırları
....

Deselect

Reservation:

BOOKWISER

Reservation

★★★★★

PICASSO
by Batuhan Karagöz

Reserve

Current Reservations
Şafağa Geçit: 26.01.2011


Extend Cancel

Yitik Güneş: 12.01.2011

Extend Cancel

Back to the info screen

Help Log-out




Help:

BOOKWISER

Help

- General information
- Modes
 - Search mode
 - Free walk mode
- Display Features
 - Shelves
 - Shelf recognition
 - Shelf genres
 - Selecting a shelf
 - Books
 - Book recognition
 - Book ratings
 - How to rate?
 - Selecting a book
 - Request more info
 - Detailed info
 - Books of the author
 - Reservation
- Rules
- Options

Back Log-out



6.3 Screen Objects and Actions


In general there are two types of objects in Bookwiser interface: Interactive objects and visual-only objects. As the names suggest, interactive screen objects are those which user can perform an action through and visual-only objects are those which user can see in the screen but not interact. There are three states of an interactive screen object: focused, active and off. An off interactive object can be seen in the screen but it is not currently available. This is the case where the object actually belongs to the screen but because of the options it is displayed as dim. A focused object is the candidate for a possible selection for an action. It can be distinguished from other interactive objects by its highlight. An active object is an object which can be selected but not focused. User should focus it using directional keys. An example of three different states of an interactive object is shown in the figure:

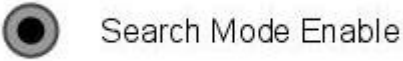


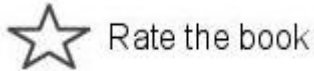
All screen actions are based on interactive screen objects and there is a general screen action mechanism in the interface. User shall be able to select the interactive objects which are not in off state. To select an object, first it should be focused. For that purpose, input device of the system needs to have special direction keys. A focused object can be selected using a special key in the input device assign to do select action. State transitions shown in the state diagram are either performed by selecting related interactive screen objects or changing the camera view.

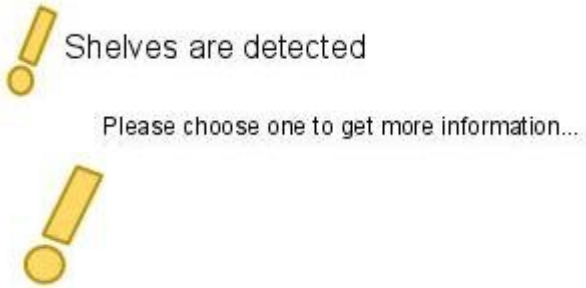
Screen objects are explained as follows:

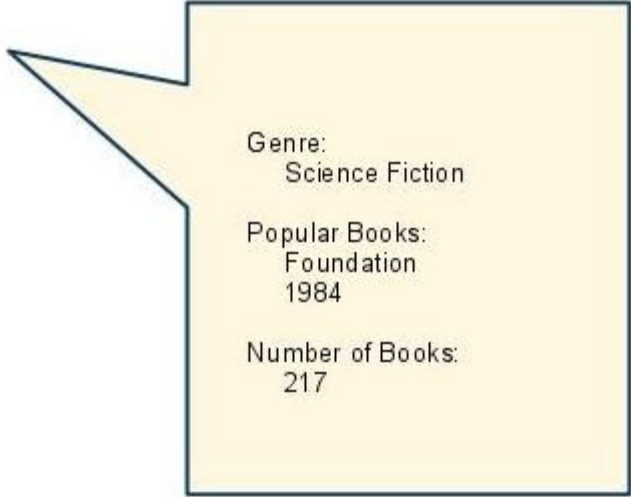
Number	1
Name	Entry Field
Type	Interactive
Description	Entry fields are used for entering textual information such as password and username.
Including Screens	Log-in,Registration,Search Mode
Instance	<div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 0 auto;">Mastering Phy...</div>


Number	2
Name	Selection Button
Type	Interactive
Description	Selection buttons are used to perform particular selection actions. Generally, they trigger a state transition between different screen classes such as logging out from library tour.
Including Screens	All screens
Instance	


Number	3
Name	Option Button
Type	Interactive
Description	Selection buttons are used to enable particular options. Differently from selection buttons they do not change the screen class.
Including Screens	All library tour screens
Instance	

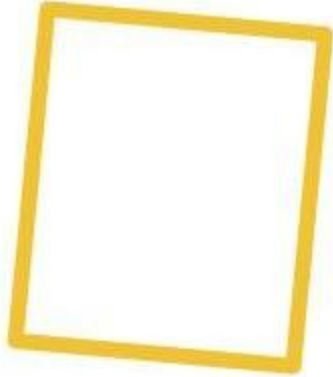
Number	4
Name	Special Selection Button
Type	Interactive
Description	Special selection buttons has a similar job to the normal selection buttons. However their mechanism is similar to option buttons. A special selection button triggers the opening of a special window for its representing action. Deselecting the button again reverse the state back.
Including Screens	All screens for help button, only information screens for other buttons
Instance	


Number	5
Name	Notifications, Notification marks
Type	Visual-only
Description	This objects are displayed to inform user about recognized objects. A notification mark is shown nearer to a object in video and a notification with texture is shown in main window.
Including Screens	Shelf detected,Book detected,Librarian detected
Instance	<p>Notification:</p>  <p style="text-align: right;">Notification Mark:</p>

Number	6
Name	Information window
Type	Visual-only
Description	Information windows are typically used for displaying contextual information about selected objects. User should request the information. These windows are transparent not to avoid user view.
Including Screens	Request shelf info,Request detailed book info,List books of the author
Instance	 <p>Genre: Science Fiction</p> <p>Popular Books: Foundation 1984</p> <p>Number of Books: 217</p>

Number	7
Name	Selecting arrows
Type	Interactive
Description	Selecting arrows are used for one purpose: To focus on different objects in the video display. One left and one right arrow is displayed when multiple objects are recognized.
Including Screens	Recognizing shelves, recognizing books, all information screens and reservation screen
Instance	

Number	8
Name	Rating Stars
Type	Visual-only
Description	A special type of objects which are used for showing ratings of the books.
Including Screens	Recognizing books, select book, rating book, request detailed book info, list books of the author, reservation
Instance	

Number	9
Name	Object Boundary
Type	Visual-only
Description	Object boundaries shows the focused recognized objects(books, shelves, librarians). For three types of objects, object boundaries are displayed with different colors.
Including Screens	Recognizing books, recognizing librarian, recognizing shelves and all information screens.
Instance	

Number	10
Name	Information boxes
Type	Visual-only
Description	Information boxes are used to combine similar screen objects in the main window. Objects inside them can be interacted but information boxes are visual-only.
Including Screens	All screens
Instance	 <p>The screenshot shows a rectangular information box with a thin black border. At the top, there are four yellow stars. Below the stars, the text 'PICASSO' is displayed in a larger font, followed by 'by Batuhan Karagöz' in a smaller font. At the bottom center of the box, there is a dark grey rectangular button with the word 'Reserve' written in white text.</p>

7 Detailed Design

7.1 Object Recognizer Component

In this section detailed design of Object Recognizer Component is provided.

7.1.1 Classification

Object Recognizer Component can be described as a subsystem of the application. Object Recognizer Component itself is not a class but contains several classes which work as data processors or data structures.

The reason behind why several data structures are not defined as simple structs but instead as classes is to make the design let the future extensions on the functionality of the component be easily implemented.

Object Recognizer component is responsible of all the object recognition procedure of Bookwiser . To achieve its goal the component mainly uses the interface provided by OpenCV library and makes all other components free from OpenCV calls, sending only the current frame captured from the camera and position and id's of recognized objects to other components. This approach makes integration of different components and also integration of OpenCV related work with interface operations done under QT platform easier.

7.1.2 Definition

As stated in the Requirements Specification Document's "Functional Requirements" section, Bookwiser application needs to recognize objects in the view of its user and give information about the objects recognized to the user. What this component accomplishes in the design is to recognize which known objects are in the view of user, so that the system can tell the user about the information it knows about these objects.

At this point, Object Recognizer Component can be described as the eye-brain channel of the Bookwiser system.

When the application starts, Object Recognizer Components starts capturing frames from the camera which can be described as the eye of the system and processes information coming from this frame.

The component , tries to find candidate objects with pre-defined methods and using its own database, tries to find what objects are on the screen. When the component is sure about the recognized objects, it informs the World Model component, the brain of the system, about detected objects.

As it can be seen, this component simply tries to recognize objects from a frame, or tracks them in the frame, and lets other components use the information it extracts from the captured image from the camera.

7.1.3 Responsibilities

In general, the main responsibility of the component is to recognize objects in the view of the user and prepare a list of recognized objects in the frame for other components to use and another responsibility is to update the position information of already recognized objects due to position changes caused by the movement of the camera by the user.

To give more detail; it can be said that there are 6 main responsibilities of this component:

- 1) Capture the current frame from the camera.
- 2) Enhance the captured image for better processing.
- 3) Track the already-recognized objects from the previous frame in the new frame and update their position information for the other components.
- 4) Find candidate areas on the frame which resembles known object types; Books, Shelves or Librarians.
- 5) Detect if candidate objects are really one of known objects or not, and state which known object it is
- 6) List the latest information of recognized objects for other components to use.

When this component finishes its job, the information of objects in the view of the user should be ready for other components to use.

It can be said that responsibilities of this component isn't directly related for the user, since the work this component does only affect other components of the system; on the other hand, actually this component is the most important part of the system because all the non-functional requirements affecting the users which are stated in the Requirements Specification, are dependent on the performance of this component.

- Accuracy of object recognition done in this component directly affects the “Reliability “ requirement stated in the SRS. If this component doesn't recognize an object correctly or recognized an object as another object, this gives false results for the user which reduces the reliability of the software.
- “Usability” of the software is dependent of the speed of this component. If the process of object recognition for a frame takes too long, the latency between capturing the frame and showing the information about the object to the user may bother the user because the user is actually moving in the real time in the library and wants to see the items in the library in real-time. If this aim is not accomplished the Bookwiser software can not be used.

7.1.4 Constraints

- The camera should give images with enough quality and should not be used by other software so that whenever this component wants it can reach the camera.
- Because a camera captures 24 frames per a second as a standard, the recognition time for a frame should be fast enough to finish processing these frames in one second, +0.3 seconds can be seen as an acceptable error time.
- It is assumed that object features which makes Bookwiser able to recognize the objects are already available in the internal object database. When a new book, shelf or librarian is added to the library, the features of the object is also added to the internal object database as in the specified formats in section 4.
- It is assumed that the library have adequate light sources around the objects so that objects can clearly seen otherwise object recognition can't be made.
- It is assumed that shelves in the library are equally spaced with a minimum of 1 meters and they are placed parallel to each other. This way a shelf can be seperated from another shelf and be recognized.
- It is assumed that books are placed on the shelves with their front side facing the camera and the user. The books are assumed to be placed with equal spaces between them, a minimum of 10 centimeters. Books should have a color that can be differentiated from the library environment and special marks on the books should be apparent and proper enough to classify them.
- Librarians should wear a standard t-shirt with the same color and a specific apparent shape on top-left of their t-shirts` front and back sides, making them recognizable.
- As an exception , if the recognition lasts too long from unforeseen circumstances the program should quit recognition process and just pass the frame to other components to make to program continue its work.
- The component should detect if theres a shelve / group of shelves in the current frame from a distance of 3-4 meters .
- The component should detect if theres a book / group of books in the current frame from a distance of 2-2,5 meters
- Bookwiser should detect if theres a librarian in the current frame from a distance of 3,5-4 meters .

7.1.5 Composition

There are 8 major subcomponents as a part of the Object Recognizer component.

1) RecognitionHandler:

RecognitionHandler works like the brain for this component. This sub-component synchronizes the work of other sub-components, analyzes and synthesizes the data coming from other components and processes resulting data to share with other components of the main system.

All the frame capturing, image filtering, object tracking, candidate area finding and object recognizing process is controlled via this sub-component.

2) FilterHandler:

This sub-component interacts with the RecognitionHandler only, gets the captured frame from the RecognitionHandler and applies specified filtering operations on the image. After the filtering operations are done the filtered and enhanced image is returned back to the RecognitionHandler:

3) BoundaryFinder:

To be able to finish the object recognition procedures in a fast way, this component is used. First some candidate areas are found, and these areas are matched from the database so processing time is decreased. This sub-component is activated by the RecognitionHandler sub-component and roughly determines the areas that are possible to be a known-object. This component finds areas that similar to known objects. Candidate areas for Librarians, shelves and books are detected with separate algorithms, built with specific properties of these objects.

After this component finishes its work it returns possible areas to the RecognitionHandler component for further processing.

4) ObjectRecognizer:

ObjectRecognizer sub-component is the sub-component that tries to match the features of the candidate objects with the known object features in the internal object database.

RecognitionHandler activates an ObjectRecognizer instance for each candidate area, and this sub-component gets features of the candidate Object and Using the FeatureMatcher sub-component matches features of the candidate object with the known object features in the database. If a successful match occurs, then pushes the recognized object to the recognized objects list.

5) FeatureMatcher:

FeatureMatcher is a low-level sub-component that used by 2 different sub-components ObjectRecognizer and ObjectTracker.

What FeatureMatcher does is gets an Object's features and tries to match the features of the object with known object features in the internal object database and share the results with the parent sub-component.

6) ObjDbaseMngr:

The main aim of this sub-component is to manage the information coming from the internal object database. It takes the information from the database in the memory and converts it to one of known data structures for other sub-components to use.

This sub-component can be used by more than one sub-component.

7) ObjectTracker:

This sub-component is again activated by the RecognitionHandler and the main aim is to track the already-recognized objects in the new frame. When some objects are detected in a frame on the next frame is highly possible that these objects are in the frame again, so there's no need to spend time on trying to recognize the same object again.

This sub-component selects some good features of the object to track from the previous frame and on the current frame watches the position changes of these features and therefore determines the new position of the object.

So this sub-component is useful to reduce the processing time of the main component and is highly important.

8) ObjectList:

This sub-component is where the recognized objects are stored. This sub-component is again managed by different sub-components . ObjectTracker uses this sub-component to see what objects are recognized in the previous frame and also this sub-component is used by other main components of the system to get information about the recognized objects.

Class diagram of the component describing relations between sub-component classes can be viewed in Diagram 5.2 in 5th section.

7.1.6 Uses/Interactions

- The component interacts with the camera using OpenCV's CvCam methods to get the current frame from the camera.
- This component is used by WorldModel component, WorldModel component uses the information about recognized objects provided by this component.
- To do this ObjectRecognizer component provides an interface called "World" , implemented by the IplImage captured from the camera and ObjectList sub-component.
- WorldModel component's Librarian, Book and Shelf sub-components inherits the Object sub-component because they extend the information provided by Object class.

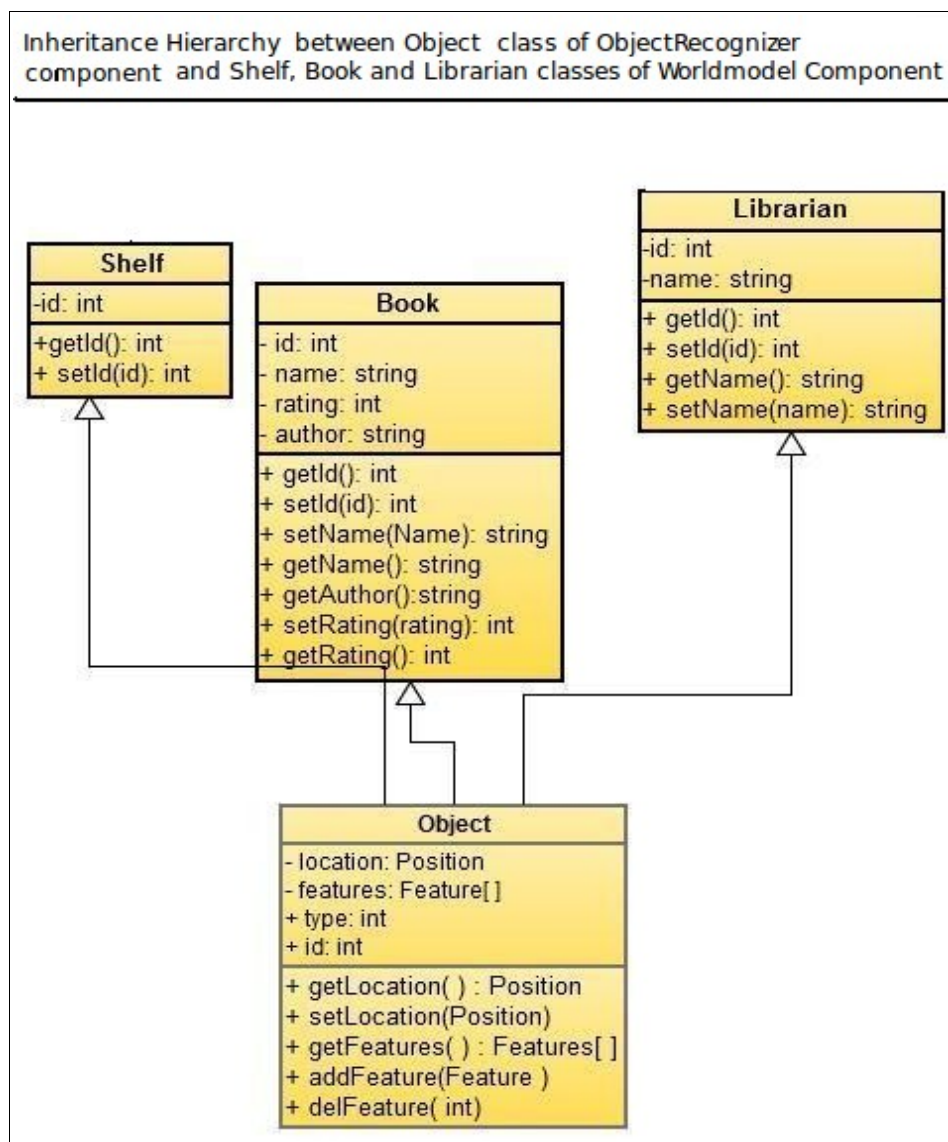


Diagram 7.1 Class Diagram

7.1.7 Resources

- The system is assumed to work on a device with a minimum of 20GB hard-disk and 1024Mb RAM. Because object features keeps a space and the component works with the images, matrices and features, the component uses memory intensively.
- The system is assumed to a work on a device with a minimum of 2.0 GHz Intel processor and a minimum of 500 core GPU, no shared memory. To make the component work fast as required, the speed of the graphics card and processor is critical for performance criteria.
- It is assumed that the system needs a minimum of 5.0 mega-pixels camera to feed the system with images with necessary quality and detail to do object recognition.

There are no race conditions or deadlock situations expected since this is the only component works with the camera and also the device the Bookwiser works on is a dedicated device and no other softwares besides the operating system works on the device.

Only memory shortage can cause problems because of the other component's uses . The situation is solved with threading and Object Recognizer component should have the highest priority for scheduling.

Several Sub-components of the component, namely FeatureMatcher and ObjDbaseMngr are also threaded so matching for different objects are done simultaneously so waiting time for an object to be recognized doesn't affect other object's recognition and if the procedure for recognition of an object passes behind the maximum time and recognition process is canceled other objects can still be recognized.

7.1.8 Processing

As explained in the 5th section main aim of the component is to recognize objects.

First the component captures a frame from the camera, filters the frame for a better quality image and operates on this image.

The component first tracks already-recognized objects in the new frame. When an object is recognized in a frame, on the next frame the object's position can be changed or the object can be completely out of the scene. The component therefore decides to update the position of the object instance on the global object list or remove the object from the recognized object list.

After that the component finds assumptions for borders of possible objects on the scene. Candidate areas are tried to match with known objects and when an object is matched object is said to be "recognized" and added to the recognized object list.

After all these procedures are done, the component checks the new frame.

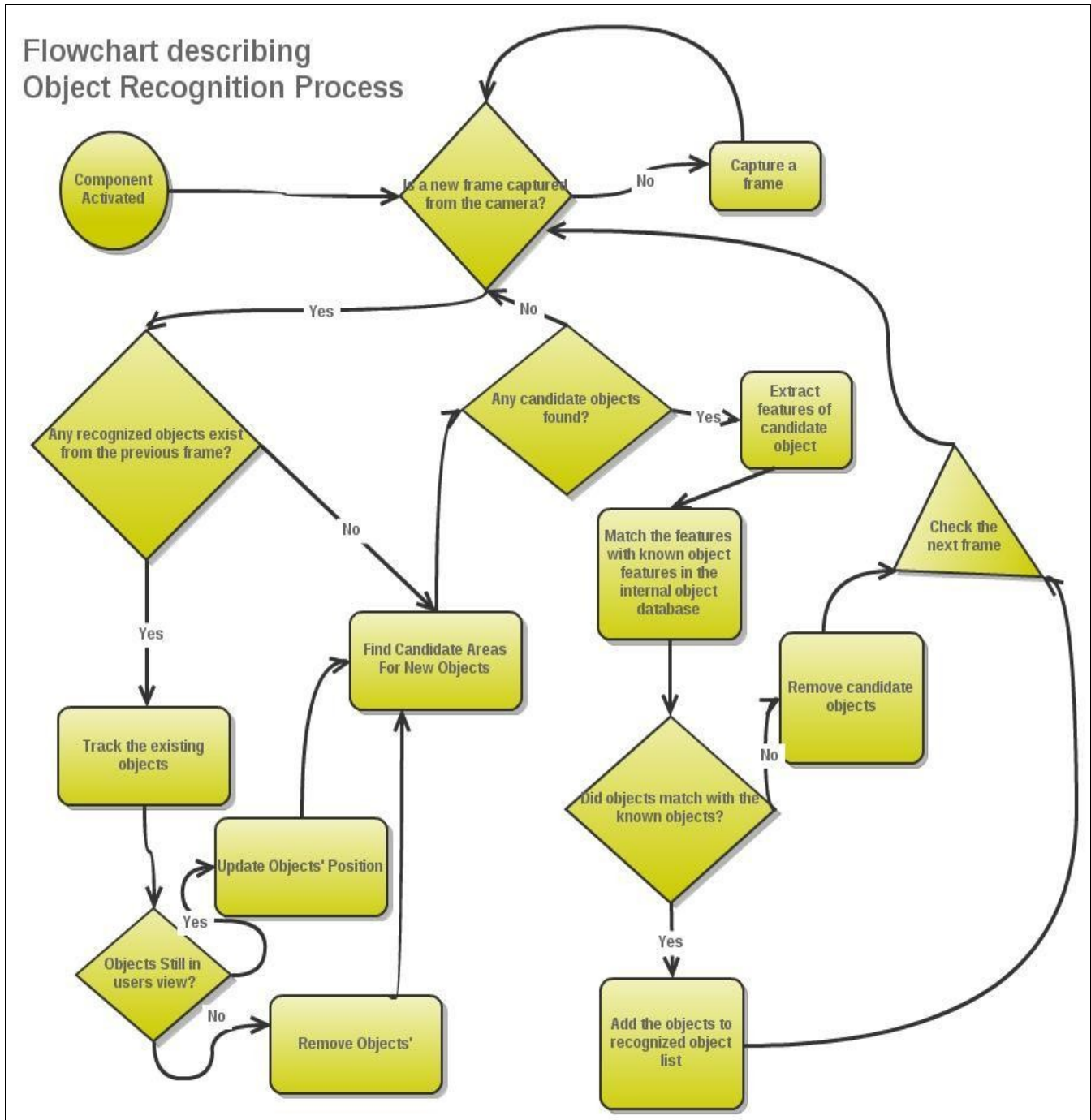


Chart 7.1 – Flow Chart

The component generally uses OpenCV library for its operations. A developer should have the information about OpenCV:

OpenCV naming conventions

- Function naming conventions:

cvActionTargetMod(...)

Action = the core functionality (e.g. set, create)

Target = the target image area (e.g. contour, polygon)

Mod = optional modifiers (e.g. argument type)

- Matrix data types:

CV_<bit_depth>(S|U|F)C<number_of_channels>

S = Signed integer

U = Unsigned integer

F = Float

E.g.: CV_8UC1 means an 8-bit unsigned single-channel matrix,

CV_32FC2 means a 32-bit float matrix with two channels.

- Image data types:

IPL_DEPTH_<bit_depth>(S|U|F)

E.g.: IPL_DEPTH_8U means an 8-bit unsigned image.

IPL_DEPTH_32F means a 32-bit float image.

- Header files:

```
#include <cv.h>
```

```
#include <cvaux.h>
```

```
#include <highgui.h>
```

Operations that are done by this component are:

1) Capturing a frame:

To capture a frame from the camera RecognitionHandler uses OpenCV's CvCam methods:

- `CvCapture* capture = cvCaptureFromCAM(0); // this method initializes the capturer of OpenCV`
- `IplImage* frame = cvQueryFrame(capture); // after the initialization to capture a frame this method should be called to capture a frame.`

2) Image Filtering:

After capturing a frame , the image is enhanced. RecognitionHandler sets the FilterHandler for a enhancement method and applies the filter on the IplImage using this sub-component.

The specific filtering methods are not given , because different filtering methods can be applied due to the quality of the camera used and lighting conditions of the library environment.

Different OpenCV filtering methods are listed on the documentation at the official OpenCV page :

“Image Filtering”:

http://opencv.willowgarage.com/documentation/cpp/image_filtering.html

The sequence diagram for Image Filtering Process can be seen in Diagram 5.3 in section 5

3) Candidate Object Boundary Finding:

After Object Tracking is done, a search for new objects is started. RecognitionHandler initializes BoundaryFinder with `init(IplImage)` call.

There are 3 methods that starts BoundaryFinder to search for candidate objects :

FindBook():

To find candidate areas that can be books is determined via several methods:

- By observation it can be easily seen that books are generally consisting of rectangles. Therefore rectangle areas can be inspected for book matching. To determine rectangles:
 - Thresholding on separate color channels of the images are used.

- By finding contours on the image via OpenCV's `cvFindContours()` method areas are detected.

- Angles between corner points are examined and if all the angles are near 90 degrees, to be more precise if the sine values of all the angles are lower than 0.3, then the area can be seen as a rectangle.

- Too small or too large rectangle areas are ignored.

- To find rectangles more easily Background/Foreground segmentation is done. To do this "cvaux.h" library is used.

`cvCreateGaussianBGModel(frame)` method of OpenCV creates a Gaussian statistics model to statistically determine background and foreground colors. When the camera moves the model is updated via `cvUpdateBGStatModel()` call and using the information of previous frame and the current frame the foreground objects are thresholded. `bgmodel->foreground IplImage` gives the foreground objects thresholded.

FindShelf():

To find candidate shelf areas same algorithm as book recognition is used except that color information is used to detect shelf areas, by color thresholding.

FindLibr():

- To find candidate librarian objects a people detection algorithm is used. Histogram of oriented gradients are used to detect human postures.

- OpenCV makes it easier to use this method with `peopleDetector()` and HOGDescriptor's are extracted from the image using `peopleDetector` of openCV with the call

```
HOGDescriptor hog;  
hog.setSVMDetector(HOGDescriptor::getDefaultPeopleDetector());
```

- And the rectangle area around the people is returned as the candidate areas.

More detailed information about HOG Descriptors are available at Wikipedia :

"Histogram of oriented gradients":

http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

As a result;

- For each found possible boundary a candidate Object is set .
- A list of found candidate objects are returned to the RecognitionHandler.

The sequence diagram for Candidate Object Boundary Finding Process can be seen in Diagram 5.5 in section 5

4) Object Recognition:

After candidate Objects are found, ObjectRecognizer is initialized by RecognitionHandler with a call to the method `init(IplImage, Object[], ObjectList)` .

`MatchSurfFeatures()` method for each candidate Object starts a recognition process.

- Candidate areas returned are set as Region of Interest with OpenCV call `cvSetImageROI()`
- On these areas Surf features are extracted with `cvExtractSURF()` method.

The extracted features are sent to FeatureMatcher and FeatureMatcher sets a database manager, ObjDbaseMngr with `dbSetConn(string)` call. This database manager gets features from each book entry and FeatureMatcher tries to match the features using NearestNeighbor Method.

More detailed information about Nearest Neighbor feature matching is available at Wikipedia :

“Scale invariant feature transform: Feature matching and indexing”:

http://en.wikipedia.org/wiki/Scale-invariant_feature_transform#Feature_matching_and_indexing

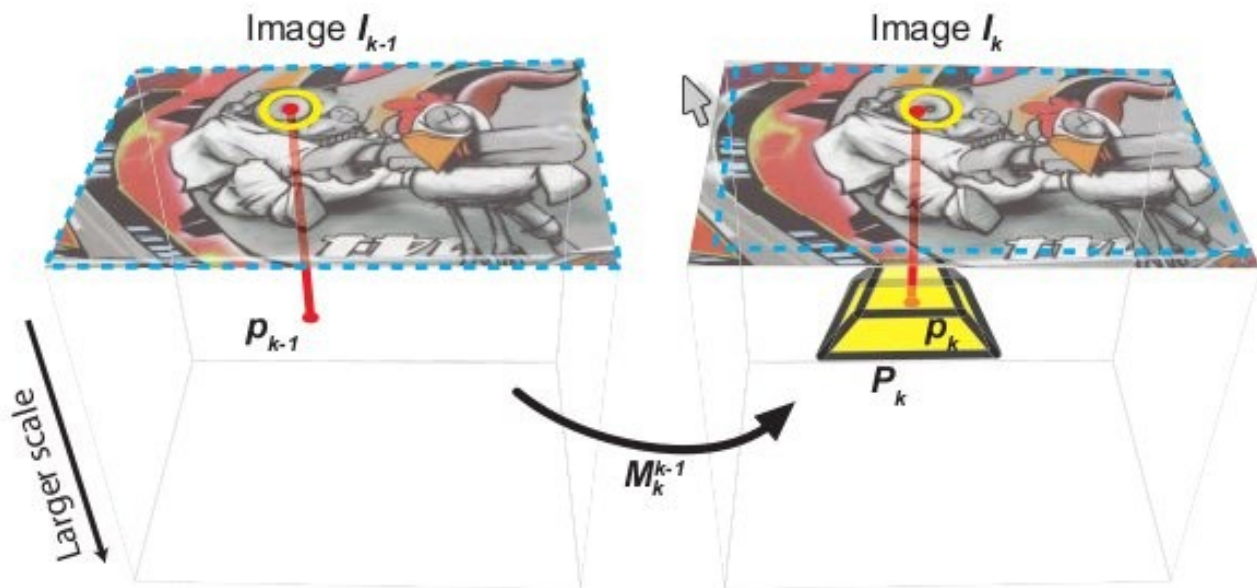
- When object features are matched correctly, Object instances are created and added to the recognized object list, ObjectList for future uses.

The sequence diagram for Object Recognizing Process can be seen in Diagram 5.6 in section 5

5) Object Tracking:

To track objects , some features of known objects are chosen by looking at their sizes and positions. Features near corner points are followed in the next frame and position changes are inspected.

RecognitionHandler , for each already-recognized object initializes an ObjectTracker instance with init(IpImage) method. After the initialization is done, startTrack(Object) starts the tracking operations. With getFeatures() method. ObjectTracker gets features from the related Object instance. And featureMatcher finds the chosen features in the image.



The change in position for different features result with the new position and orientation of the object in the image.

The sequence diagram for Object Tracking Process can be seen in Diagram 5.4 in section 5

7.1.9 Interface / Exports

The component as a result of all its duties creates an interface called World.

World interface consists of the image of the current frame and the list of recognized objects, namely ObjectList .

This interface is provided to World Model component , so that World Model component is now informed about the recognized objects and their positions on the current frame image. Therefore World Model component can update context information about the recognized objects from the library database.

7.2. World Model Component:

7.2.1. Classification

This model is a subsystem that consists of classes.

7.2.2. Definition:

The purpose of this component is mainly coordinate all of the components and make them work properly in a way of coordination. Since world model component has so many interactions with object recognizer component and graphic component, it mainly controls the information that flows in the **Bookwiser** application. So, semantic meaning of this component has a great importance among other components. For example, a major mistake in the graphic component affects only the graphic component and its operation. On the other hand, even a minor mistake in the world model component can cause a collapse of whole operation in the application.

7.2.3. Responsibilities

The main responsibilities of this component can be explained by the following steps:

- Coordination of the general classes that defined in the component, such as ObjectManager, WorldManager, LibraryDbaseManager.
- Coordination of the data classes that defined in the component, such as ObjectTable, StateChart (in the interface of WorldRepresentation), Librarian, Book, Shelf (inherited from the Object data class) and State.

The roles of this component play parts in the application related with the feature matching. So, this component interacts with the internal object database directly. For this reason, it provides finding features of the detected object in the object recognizer component, having comparisons these features with the internal object database in a sufficient fast time with a reasonably efficient algorithm and giving the results to the graphic unit. This is the basic component that “**real-time**” operations are provided to the clients of the application.

7.2.4. Constraints

Constraints for world model component can be listed as follows:

- Bookwiser needs to communicate with the library database , therefore it is assumed that the library database is open to Bookwiser and the library database is able to share information via Internet or a local network.
- Bookwiser uses the library's own database to show data about recognized books. This database, called internal object database, locates in this component. So, world model component has to have the capacity of this library.
- Since, internal library is directly interacted in this component, the feature matching algorithms should be very fast in order to display the result to the user in real-time.
- The input coming from the object recognizer component should be in a form of containing features:set of float numbers.
- The input coming from the internal object database in the form of string.
- Book, Shelf and Librarian classes should be inherited from the Object class in order

to use the same class for features and position detection in object recognizer component.

7.2.5.Composition

This component consist of 6 data classes, namely ObjectTable, StateChart, Shelf, Book, Librarian, State, and 3 general classes, namely ObjectManager, WorldManager and LibraryDbaseManager.

- Shelf class is used to hold information about detected shelf/shelves. A Shelf object is recognized by its id and the methods getId and setId are used for this purpose.
- Book class is used to hold information about detected book/books. A Book object contains number of fields such as id, name, author and rating. The set methods in the Book class are used to change the current state of the field and the get methods are used to get the value of that field.
- Librarian class is used to hold information about the detected librarian people in the library. The fields name and id are necessary for the comparison after object recognition has been completed. A Librarian object is the result of object detection in the object recognizer part because librarians are recognized by their specific costumes and id cards, no face detection algorithms are used in this part.
- Shelf, Book and State classes are inherited from the Object class. The meaning of the use of Object class is not to interact with the specific objects, instead to interact with the general objects in terms of locations and features which will be accessed directly.
- State class is used to hold the information about the current state. Only type field is necessary to identify what the state of **Bookwiser** is. State objects currently refer to the modes of the application such that select a book mode or free walk mode when the application is started and used by the user actively.
- ObjectTable class is used to create tables from the finite number of objects in the current state. The methods of this class is used to insert new objects, update this table or list the object specified in the table.
- Similarly, StateChart class is used to hold all information about the current state in a table form. Also, it is possible to add a new state, update current state or remove the state from the table.
- LibraryDbaseManager class is provided to control the elements retrieved from the library. The purpose of the retrieving those elements from the library is actually the main purpose of this component. That is to say, the features of the detected objects coming from the object recognizer component is compared with the elements/features retrieved from the library.
- The information obtained by the class of LibraryDbaseManager is brought to the other important class, which is ObjectManager. ObjectManager class controls the objects in the ObjectTable class by the information coming from LibraryDbaseManager class. “arrangeObj” method is managed to hold these kind of operations.
- The spine of world model component is the class named WorldManager. All the information about objects, modes(states) and their updates and any possible changes are gathered in this class. Transfer of information between components is managed by the send methods of this class.

7.2.6.Uses/Interactions

World Model component is the spine of the application. So that, it directly interacts with the two other components. Any side effect in object recognizer component directly affects the process on world model component because the features of the detected objects are retrieved from the object recognizer component. Moreover, after the matching process is completed in the world model component, the information is sent to the graphic component. So, any side effect coming from outside, namely database, or a corruption in the world model component directly affects the graphic component.

Bookwiser system is designed in an object oriented manner. So that, it has so many classes, which can be either superclass or subclass. The interactions between them can be stated as follows by the use of class diagram of world model component referenced by Diagram 5.7, World Model Component Class Diagram:

- Object class is the superclass of the Book, Shelf and librarian classes. Book, Shelf and Librarian classes are mainly used in world model component, however their general fields such as feature and position are used in object recognizer component.
- There are two classes that hold tables about lots of objects and states, one is stated for Object class, and the other is stated for State Class.
- ObjectTable and StateChart classes implement an interface, called WorldRepresentation in order to flow a general information between components.

7.2.7.Resources

Any and all sources that are managed, affected or needed by the world model component can be listed as follows:

- A database is the basic resource needed by this entity. This database is used for the matching process. This database includes float numbers that have a meaning with the corresponding feature and position information. As mentioned in the constraints part, there is a time limit in this relationship between internal object database and input coming from object recognizer component to the world model component. Retrieving the matched information from the internal database needs reasonably quite time. So that, efficient algorithms should be used in order to deal with this limit and those algorithms are described in the processing part.
- A software library, namely OpenCV, is needed for this component. High qualities of OpenCV and its useful relationship with Qt, which is needed in the graphic component to create the graphical user interface of the **Bookwiser**, makes the OpenCV the best choice for the application. So that, how the possible disconnections between graphic component and world model component is resolved by the giving path for the Qt in the OpenCV.

7.2.8.Processing

The algorithm is used how the world model component performs its duties to fulfill its responsibilities can be described as follows:

- The information comes from the Object Recognizer Component to the World Model Manager Component.

- After, “setConn” function is called and returned by true, Context Information coming from the Library Database is brought to the ObjectManager class in order to be combined with the information coming from the Object Recognizer Component.
- Step2 is repeated whenever the information coming from Object Recognizer Component is updated.
- Mode updates information comes from the Graphic Component to the World Model Manager Component.
- Step4 is repeated when the user changes the state of the system or select one of the modes by the input controller such as Search Mode and FreeWalk Mode.
- When a book, shelf or librarian is chosen by the user, incoming states from the Graphic Component are stored in the state bank of the component.
- Output of the component is manipulated by the changes on the states.

Relevant time for this algorithm for the world model component can be estimated as 1 second. There are 24 frames needed to process in a 1 second video frame. Since **Bookwiser** should provide real-time operations, world model component gains a great importance on time complexity. For this reason, the matching operation can be in the late at most half of the process time.

Relevant space for this component should be enough to hold the library database. Since this library database has an efficient storage of features and positions of detected objects, this is mainly not a problem for the application.

7.2.9.Interface/Interactions

All services that are provided by world model component can be listed as follows:

- **Resources:** LibraryDbaseManger is the resource of the world model component.
 - LibraryDbaseManger: This is a class entity used to retrieve the information about detected objects. It has two methods, namely setConn and getElmInfo. SetConn method is used to return a boolean value that is either a connection established between library and other methods or not. sentConn gets a string parameter which has a meaning of the element. getElmInfo method is used to return a string that corresponds the element used in ObjectManager class in the process of matching. LibraryDbaseManager I referenced by the Diagram 5.7 World Model Component Class Diagram.
- **Data:** Shelf, Book Librarian, Object, State StateChart, ObjectTable classes are the basic data classes of **Bookwiser**.
 - Shelf Book and Librarian classes are used to define and match the specific objects detected by the camera. They have some methods in common, such as getId and setId. Since each object is defined by their unique ids, an integer type, namely id is used in these methods. Id fields can be changed by the use of setId method and obtained by the use of getId method for all these 3 classes.
 - Book class has name, rating and author fields in order to specify the further information about a Book object matched in this component. They have all get and set methods that can be used change the value of the parameter and obtain the value of that parameter.

- Librarian also has name and id fields and their get set methods to specify a Librarian object.
- ObjectTable class holds a table that consists of Objects. This class implements an interface with the other table located in StateChart class. ObjectTable class has three methods each of which is used to modify the table. insertObject, updateObject and listObject all have a parameter of Object and used to respectively insert a new object to the table, update the values of an object specified by the parameter and list the information about object specified in the parameter. ObjectTable is directly interacted with Book Shelf and Librarian classes in a form of one to many relation. That means lots of Book, Shelf and Librarian objects can be created and hold in ObjectTable class.
- State class has a field named type in order to access the state object uniquely. get and set methods are used to process the access operation.
- StateChart class hold a table which consists of State objects. This class implements an interface with the other table located in ObjectTable class. StateChart class has three methods each of which is used to modify the chart. getState, setState and removeState, all have a parameter of State and used to respectively obtain the state given in the parameter from the chart, update the values of the state specified by the parameter and deletes the state from the chart specified in the parameter. StateChart is directly interacted with State class in a form of one to many relation. That means lots of State objects can be created and hold in StateChart class.

The relationships between these classes and their methods are clearly shown in the sequence diagrams of world model component, referenced in 5.8 and 5.9 sequence diagrams; World Model Component.

- **Types:** Interface (WorldRepresentation), database (string), id (int), name (string), rating (int), author (string), type (int), objects (Object []), states (State []) are the main types used in all methods and classes of world model component. These types used as parameters and return values of methods and those relationships can be seen in 5.7 class diagram of world model component.
- **Subroutines:** ObjectManager and WorldManager classes can be described as subroutines of the world model component because all the basic operations are processed using these classes and their methods. arrangeObjInfo is a method that used to establish a relationship between LibraryDbaseManager and ObjectTable.

Class Diagram - World Model Component

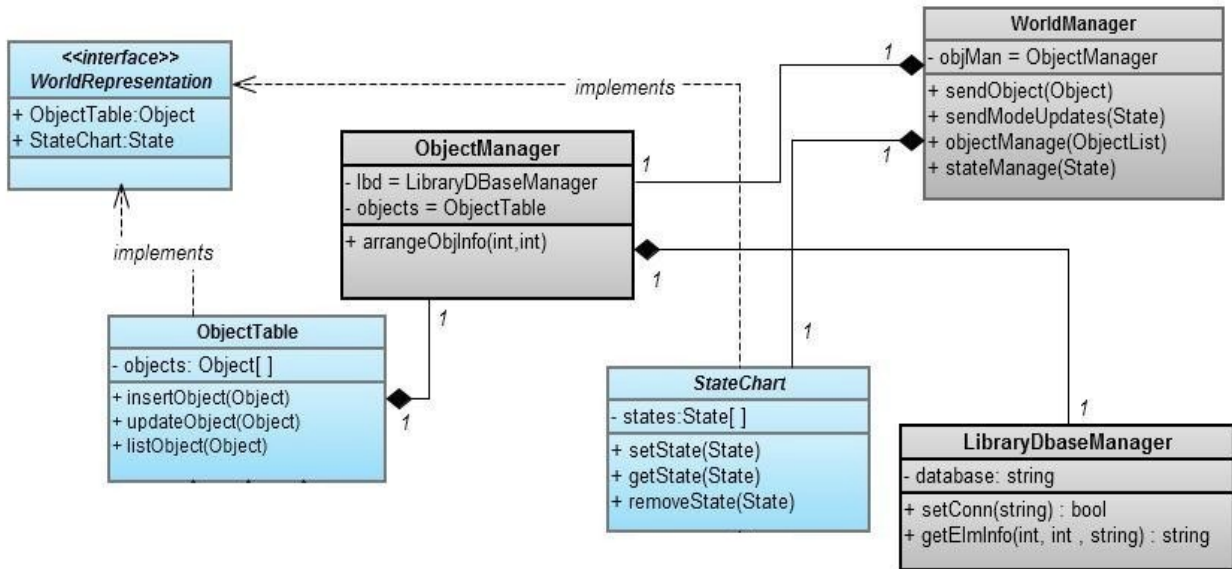


Diagram 7.2-World Model Component Class Diagram

- **Exceptions:** In this service, possible exceptions that may occur are explained. Exceptions occur when the world model component of **Bookwiser** application operates an abnormal process. For the data classes and their methods, there are possible “not found” exceptions may occur, especially in get methods. In the ObjectTable and StateChart classes, there may be “null reference” exceptions because of the allocating space for the object table and/or state chart.

7.3 Graphic Component

Class Diagram - Graphics Component

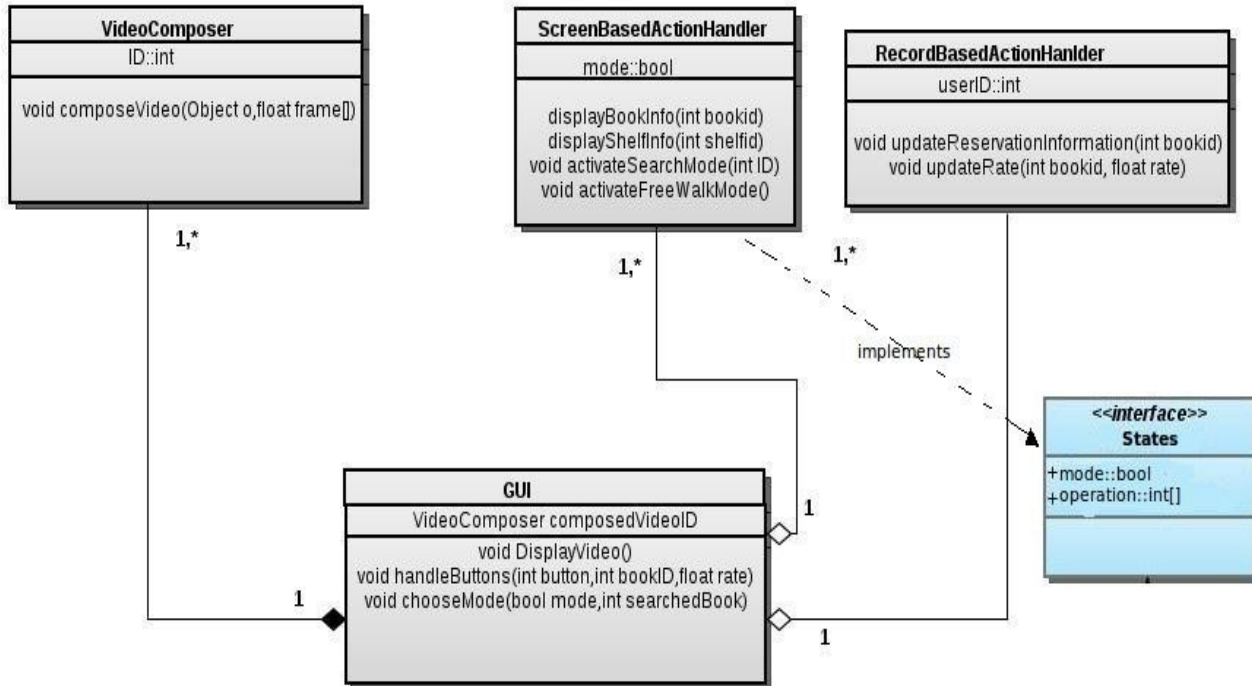


Diagram 7.3-Class Diagram of Graphic Component

7.3.1. Classification

The Graphic Unit is a subsystem of the Bookwiser System which interacts with the World Model Unit and the user himself/herself.

7.3.2. Definition

The purpose of this component is to interact directly with the user for every action. These actions are two-way. Giving information to the user and getting information from him/her. Giving information means displaying the camera view with augmented contextual information. Getting information is handled by input controller. User inputs are classified and sent to related components. (The button clicks) Note that user inputs are not processed by the graphic unit.

The button clicks were described in Bookwiser SRS document Use Case Diagram. (page 11)

7.3.3. Responsibilities

The main responsibility of this component is to compose the object information coming from World Model Unit with the objects captured in the video coming from the camera and forward the process to the World Model Unit as the user requires through the button clicks.

7.3.4.Constraints

The constraints for the Graphic Unit are that:

- The Graphic Unit must be in touch with the World Model Unit without some serious time-delay in order to get and send the information accurately.
- The Graphic Unit must be in touch with the library database without some serious time-delay in order to get the required information accurately.
- The input coming from the World Model Unit- corresponding to the object information(book, shelf, librarian information) must be given in an appropriate order as to be placed correctly on the captured video captured by the camera.
- The video captured by the camera must be received without any serious time-delay in order to place the object-information (shelf, book & librarian information) correctly on the video to display it to the user.
- The “Display Book Information” button click cannot be active when there is multiple shelves in the user's vision, in other words the selection of a book must be available.
- The “Rate the Book” option cannot be active when there is multiple shelves in the user's vision, in other words the selection of a book must be available.
- The “Reserve the Book” option cannot be active when there is multiple shelves in the user's vision, in other words the selection of a book must be available.
- If the user wants to be alerted when the book he/she is searching for is in his/her vision the “Search Mode” must have been chosen.
- The “Display Shelf Information” button click cannot be active when there is no shelf in the user's vision, in other words the selection of a shelf must be available, and it is required that the user must have at least one shelf in his/her vision.

7.3.5.Composition

The Graphic Unit has been composed by 4 subcomponents:

1-) Video Composer:

The purpose of the Video Composer is to compose the video captured by the camera with the object information, specifically shell, book, librarian etc. in order to forward the composed video to the user.

The main responsibility of this component is to compose the object information coming from World Model Unit with the objects captured in the video coming from the camera.

2-) Screen Based Action Handler

The purpose of the Screen Based Action Handler component is to handle the screen updates.

The responsibilities of the Screen Based Action Handler can be seen in two parts:

Firstly, it is responsible for the user mode changes: whether the mode is free-walk mode or the search mode. The Screen Based Action Handler has to announce the World

Model Unit about the mode change.

Secondly, it is responsible to display the shelf and book information. When the user has some shelves (one or multiple, but at least one) in his vision and require the information about one specific shelf the Screen Based Action Handler announces the World Model Unit about that requirement. In a similar way; when the user has some books (one or multiple, but at least one) in his vision and require the information about one specific book the Screen Based Action Handler announces the World Model Unit about that requirement.

3-) Record Based Action Handler

The purpose of the Record Based Action Handler component is to handle the book record updates in the library database.

The responsibilities of the Record Based Action Handler can be seen in two parts:

Firstly, when the user wants to rate a book that he is selected from his vision, the Record Based Action Handler connects to the library database and update the rate of that book according to the user's request.

Secondly, when the user wants to reserve a book, he had selected in his vision, the Record Based Action Handler connects to the library database and update the reservation information of that book according to the user's request.

4-) GUI

The purpose of the GUI component is to directly interact to the user in order to handle user's requests appropriately.

The responsibilities of the GUI component can be examined in two parts:

Firstly, it is responsible to transmit the button clicks to the regarded components in order to process as required by the user.

Secondly, it is responsible to display the information of the objects in the user's vision in a user-friendly manner.

7.3.6.Uses/Interactions

The Graphic Unit interacts mainly with three separate sections: the user himself/herself, the database library and the World Model Unit. The detailed interaction can be explained in terms of subcomponents more clearly:

1-) Video Composer:

Video Composer deals with the World Model Manager Unit subcomponent of World Model Unit & the camera capturing the video.

The Video Composer composes the object information coming from the World Model Manager Unit with the video captured by the camera.

2-) Screen Based Action Handler

Screen Based Action Handler deals with the World Model Manager Unit subcomponent

of World Model Unit.

Screen Based Action Handler informs the World Model Manager Unit about the mode updates and the requests of the users to display information about a book or about a shelf.

3-) Record Based Action Handler

Record Based Action Handler deals with the library database, only.

Record Based Action Handler updates the rate or reservation information in the database library according to the user's demands.

4-) GUI

The GUI subcomponent of the Graphic Unit is dealing only with the user himself/herself.

The GUI subcomponent displays the final video to the user and checks on the user requests via button clicks.

7.3.7.Resources

The resources that Graphic Unit needs are:

Library Database:

The Graphic Unit needs library database in order to update the information related to the rate and reservation information of the book.

Camera:

The Graphic Unit needs camera in order to put the relevant object information onto the video captured by the camera.

7.3.8.Processing

The processing can be explained in terms of components more clearly:

1-) Video Composer:

void composeVideo(Object object, float frame[]) : Overwrites the frame composing with the object information coming from World Model Unit.

2-) Screen Based Action Handler:

void displayBookInfo (int bookID) : Announces the World Model Unit that the detailed information of the book with the bookID is requested to be displayed on the screen.

void displayShelfInfo (int shelfID): Announces the World Model Unit that the

detailed information of the shelf with the shelfID is requested to be displayed on the screen.

void activateSearchMode (int bookID) : Announces the World Model Unit that the user mode is to be set up to the “Search Mode” and the book with the bookID is to be searched.

void activateFreeWalkMode() : Announces the World Model Unit that the user mode is to be set up to the “Free Walk Mode”.

3-) Record Based Action Handler:

void updateReservationInformation(int bookID) : Updates the book information given with the bookID as reserved in the library database.

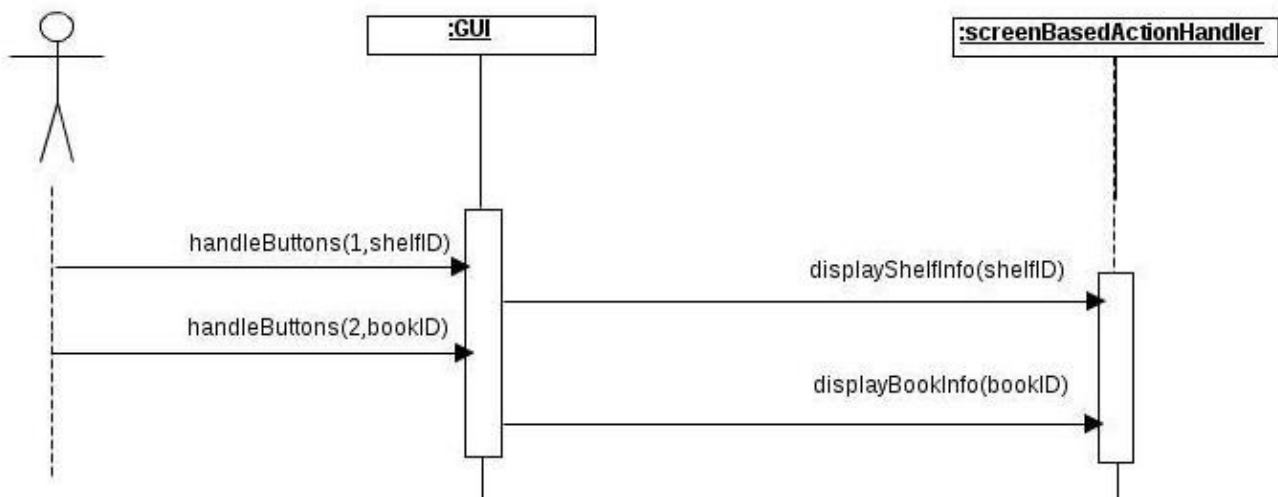
void updateRate(int bookID, float rate): Updates the book rate given with the bookID according to the given rate in the library database.

4-) GUI:

void displayVideo(): Displays the updated video frame by the Video Composer. This method is called by the Video Composer in order to warn the GUI that the video is updated and is to be displayed.

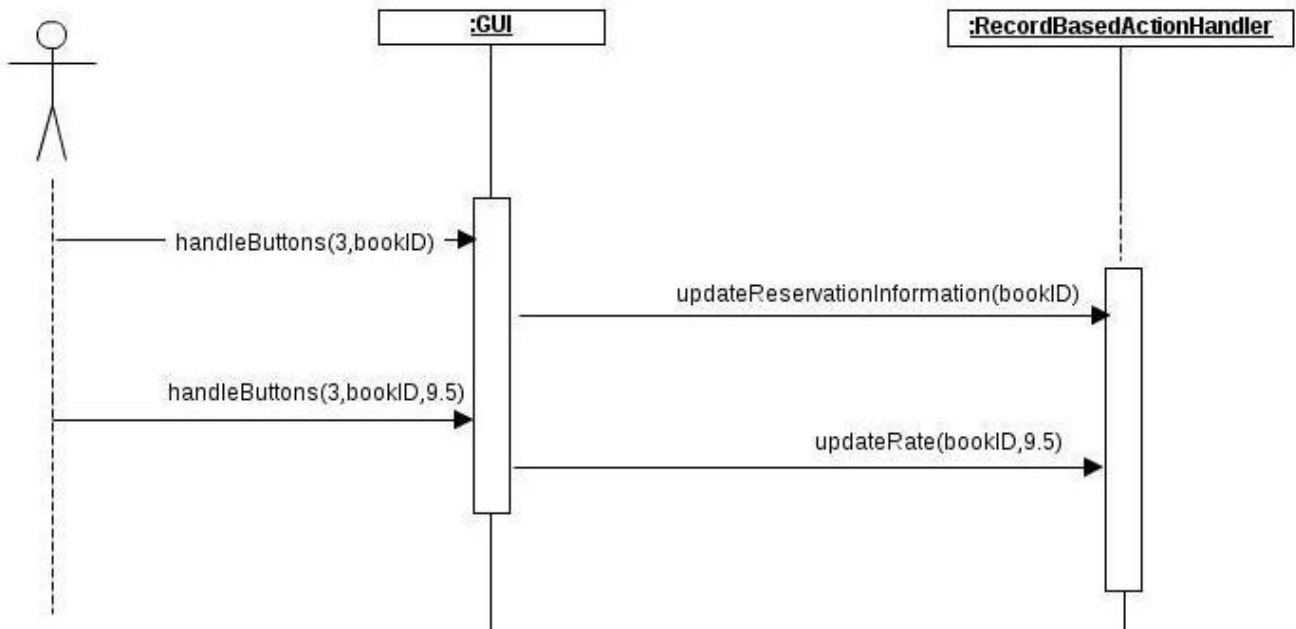
void handleButtons(int button,int bookID,float rate): Handles the button clicks. According to the given button click calls the required methods. It can be seen in the following sequence diagrams:

Sequence Diagram - Choosing Objects



Sequence Diagram 7.3.1

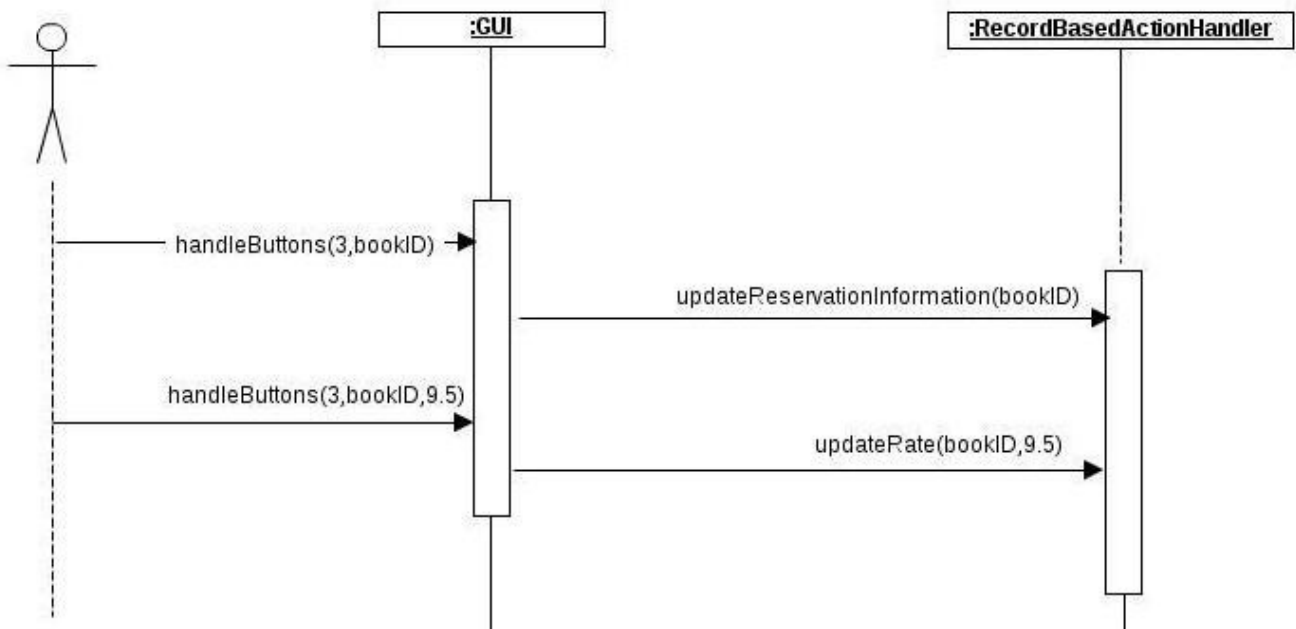
Sequence Diagram - Record Updates



Sequence Diagram 7.3.2

`void chooseMode (bool mode, int searchedBook):` Changes the mode according to the user's request. It can be seen from the following sequence diagram:

Sequence Diagram - Record Updates



Sequence Diagram 7.3.3

7.3.9.Interface/Exports

The Graphic Unit component can be seen as a subsystem of the Bookwiser interacting only with the World Model Unit, the user, the camera & the library database, basically.

The inputs that the Graphic Unit subsystem needs are that the video captured by the camera & the object information sent by the World Model Unit.

The outputs that this subsystem generates are the updates on the library database, the button click information sent to the World Model Unit & the final video frame that the user needs to see in his vision.

The exceptions that the Graphic Unit throws are in the following situations:

- If there is no book selected & the user requests a displaying detailed information of some book.
- If there is no book selected & the user requests an updating rate some book.
- If there is no book selected & the user requests a reservation of some book.
- If there is no shelf selected & the user requests a displaying detailed information of some shelf.

8 Libraries and Tools

In this section, used libraries and tools are described.

1. OpenCV:

In the Bookwiser project, OpenCV library is used for the object detection part. OpenCV is an open source computer vision library in C/C++. Since it is optimized and intended for real-time applications, it is very applicable to use in Bookwiser which is a real-time augmented reality application. Bookwiser is designed to be platform independent in a way that it will work on the mobile devices. Independence of operating system/hardware or window-manager, OpenCV is the best choice for our purpose. Moreover, it provides a generic image/video loading and both low and high level of API.

OpenCV has considerably high quality features that provides so much easiness to the developers:

1. Image and video I/O (file and camera based input, image/video file output)
2. Various dynamic data structures (lists, queues, sets, trees, graphs)
3. Basic image and video manipulation processing such as filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids.
4. Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation)
5. Camera calibration (finding and tracking calibration patterns, calibration, fundamental matrix estimation, homography estimation, stereo correspondence)

6. Motion analysis (optical flow, motion segmentation, tracking)

7. Object recognition (eigen-methods, HMM).

For all the reasons listed above, **Bookwiser** uses the OpenCV.

2. QT Tool:

For the graphical user interface of Bookwiser, QT tool is used.

QT brings many advantages as portability, predefined interface with OpenCV and useful built-in tools for GUI development.

9 Gantt Chart



10 Conclusion

Bookwiser is a valuable project that will both serve the users and the owners of the system in a positive way, being a innovative idea and aiming to create a more comfortable environment for its users.

The project should be developed in a complete understanding of the facts behind the need for the project, the importance of the results of the project and should be considered as a part of the instinct that creates the need of progress.

All the design details stated in this document is the result of a mixture containing innovative thinking, analytic approach and a proper analysis of users' needs; therefore, developers should not forget that these details have a critical importance for all elements of all levels in the project as a whole.

All the design details stated in this document should give the developers a design mainframe with some details.