**CHAPTER 1**

**INTRODUCTION**

* 1. **BRIEF DESCRIPTION ABOUT THE PROJECT**

World is contracting with the growth of mobile phone technology. With the number of users increasing day by day, facilities are also increasing. Starting with simple regular handsets which were used just for making phone calls, mobiles have changed our lives and have become part of it.

As cell phones are getting smarter everyday with features like GPS, Accelerometer, Gyroscope and processor speeds improving, we may soon be spending more time in a virtually enhanced world. This enhancement can be done through Augmented Reality.

This new technology, called Augmented Reality, blurs the line between what's real and what's computer-generated by enhancing what we see, hear, feel and smell. Augmented reality is changing the way we view the world or at least the way its users see the world. The basic idea of augmented reality is to superimpose [graphics](http://computer.howstuffworks.com/3dgraphics.htm), audio and other sensory enhancements over a real-world environment in real time.

Location based applications for a mobile augmented reality system is a natural interface to visualize spatial information such as position or direction of locations and objects for location based applications that process and present information based on the user's position in the real world. This concept can be used to provide a new dimension to college touring.

The application proposed allows students to get the real “inside scoop” on college life by taking tours around the college. On campus, students can use the “Augmented Reality” view to point their phone at buildings and pull up information.

The application utilizes the hardware configuration of mobile phones, in particular the GPS, compass, accelerometer, camera to provide enhanced views of campuses. Views of a real campus are enhanced with a wealth of information a visitor might want to know. The application allows visitors to learn about buildings on the campus easily and quickly on their own.

**CHAPTER 2**

**LITERATURE SURVEY**

Augmented reality (AR) is a relatively new and promising concept. The ability of superimposing digital elements on a physical world with means of interaction with the surrounding world is quite intriguing idea, since augmented reality introduction in 1968 by Ivan Sutherland. However, the technology by that time and for almost next 3 decades was quite limited to lab research, since the mobility nature of augmented reality, and lack of capable mobile processors.

The rapid development of mobile GPUs, CPUs and recently hybrid processors, leads to an increase in popularity of the augmented reality technology. Mobile devices play important role in augmented reality technology as they combine processor, memory, display and interaction technology into one single device. There are two main trends in augmented reality research: registration, where researchers try to solve misalignment and world tracking problems; and integration, where re-searchers are directed towards the enhancement of computer generated object integration with the surrounding environment. This work proposes an augmented reality mobile platform for enhancing integration of virtual objects in outdoor environments.

[1] Mohamed El-Zayat, ‘Augmented Reality Platform for enhancing integration of virtual objects’, 2011. This paper proposes the augmented reality mobile platform for arranging augmented reality objects with an emphasis on enhancing the integration of computer generated objects in the outdoor environment, by introducing lights and shaders to the augmented objects.

[2] Shane R. Porter\_ Michael R. Marner† Ross T. Smith‡ Joanne E. Zucco§ Bruce H. Thomas,Validating Spatial Augmented Reality for Interactive Rapid Prototyping,IEEE 2010. This paper investigates the use of Spatial Augmented Reality in the prototyping of new human-machine interfaces, such as control panels or car dashboards. The prototyping system uses projectors to present the visual appearance of controls onto a mock-up of a product. Finger tracking is employed to allow real-time interactions with the controls. This technology can be used to quickly and inexpensively create and evaluate interface prototypes for devices.

[3] T. Miyashita, P. Meier, T. Tachikawa, S. Orlic , T. Eble , V. Scholz, A. Gapel, O. Gerl, S. Arnaudov, S. Lieberknecht. ‘An Augmented Reality Museum Guide’, 2008. This paper proposes a three year project comprising six presentations, with the goal of gaining experience in innovative multimedia approaches to bring together visitors and artworks. Museum Lab is composed of three main spaces (Presentation Room, Information Space, Theater) and a Lobby. Information provided by multimedia tools surrounds the actual artworks on display, offering the visitor various approaches to these artworks.

**CHAPTER 3**

**PROBLEM DEFINITION**

High-end mobile phones have developed into capable computational devices equipped with high-quality color displays, high resolution digital cameras, and real-time hardware-accelerated 3D graphics. They can exchange and sense location using GPS. This enables a new class of augmented reality applications which use the phone camera to initiate search queries about objects in visual proximity to the user.

Many in the augmented reality community use object recognition to provide information about the recognizable objects viewed by the user. However this approach works only if the user is looking at specific known objects. It does not support augmented reality when there are no special objects in the scene. Also the current system doesn’t have the ability to discriminate between foreground and obscured buildings.

**CHAPTER 4**

**REQUIREMENT SPECIFICATIONS**

**4.1 HARDWARE REQUIREMENTS**

**4.1.1 Smartphones Running Android**

Mobile phones are no more used for making or receiving calls alone. Using smartphones we can do a myriad of things. Our project requires a smartphone running on Android 2.3.3 or more and GPS receiving capability built in.

**4.2 SOFTWARE REQUIREMENTS**

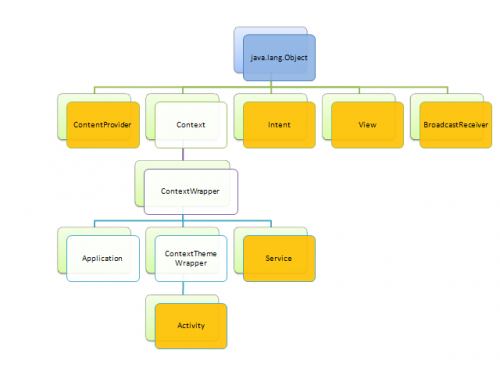
**4.2.1 Android SDK**

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android SDK enables developers to create [applications](http://www.webopedia.com/TERM/A/application.html) for the [Android platform](http://www.webopedia.com/TERM/A/Android_platform.html). The Android [SDK](http://www.webopedia.com/TERM/S/SDK.html) includes sample projects with [source code](http://www.webopedia.com/TERM/S/source_code.html), development tools, an [emulator](http://www.webopedia.com/TERM/E/emulator.html), and required libraries to build Android applications. Applications are written using the [Java](http://www.webopedia.com/TERM/J/Java.html) programming language and run on [Dalvik](http://www.webopedia.com/TERM/D/Dalvik.html), a custom [virtual machine](http://www.webopedia.com/TERM/V/virtual_machine.html) designed for embedded use which runs on top of a [Linux](http://www.webopedia.com/TERM/L/Linux.html) kernel.

**4.2.1.1 Java Classes**

The figure 4.1 represents a class hierarchy of main classes

from Android SDK a developer has to deal with while developing Android application.

**FIGURE 4.1:**  Class hierarchy of main classes from Android SDK

Content providers manage access to a structured set of data. They encapsulate the data, and provide mechanisms for defining data security.

Context is an interface to global information about an application environment. This is an abstract class whose implementation is provided by the Android system.

Intent is an abstract description of an operation to be performed. View class is a base class for all GUI widgets.

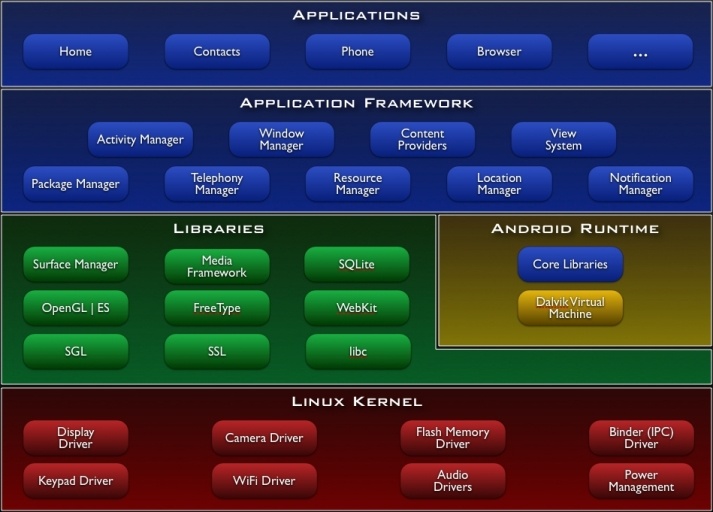
[BroadcastReceiver](http://developer.android.com/reference/android/content/BroadcastReceiver.html) class and its subclasses serve as “subscribers” in [publish/subscribe](http://en.wikipedia.org/wiki/Publish_subscribe) (or pub/sub) communication mechanism implemented by Android application architecture.

[Activity](http://developer.android.com/reference/android/app/Activity.html) class and its subclasses are the ones that provide logic behind the GUI. Actually it corresponds to ViewModel in [Model-View-ViewModel](http://en.wikipedia.org/wiki/Model_View_ViewModel) architecture pattern (MVVM).

A Service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use.

**4.2.1.2 ANDROID ARCHITECTURE**

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The architecture of android operating system is split into Applications, Application Framework, Libraries, Android Runtime and Linux Kernel. Figure 4.2 depicts the architecture of android operating system. Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.



**FIGURE 4.2: Android Architecture**

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework.

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

**4.2.1.2.1 DALVIK VIRTUAL MACHINE**

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple virtual machines efficiently. The Dalvik virtual machine executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The virtual machine is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik virtual machine relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

**4.2.2 IDE – Eclipse**

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible [plug-in](http://en.wikipedia.org/wiki/Plug-in_(computing)) system. It is written mostly in [Java](http://en.wikipedia.org/wiki/Java_(programming_language)). It can be used to develop applications in Java and, by means of various plug-ins, other [programming languages](http://en.wikipedia.org/wiki/Programming_language) including  [Ada](http://en.wikipedia.org/wiki/Ada_(programming_language)), [C](http://en.wikipedia.org/wiki/C_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B), [COBOL](http://en.wikipedia.org/wiki/COBOL), [Perl](http://en.wikipedia.org/wiki/Perl), [PHP](http://en.wikipedia.org/wiki/PHP), [Python](http://en.wikipedia.org/wiki/Python_(programming_language)), [R](http://en.wikipedia.org/wiki/R_(programming_language)), [Ruby](http://en.wikipedia.org/wiki/Ruby_(programming_language)) (including [Ruby on Rails](http://en.wikipedia.org/wiki/Ruby_on_Rails) framework), [Scala](http://en.wikipedia.org/wiki/Scala_(programming_language)), [Clojure](http://en.wikipedia.org/wiki/Clojure), [Groovy](http://en.wikipedia.org/wiki/Groovy_(programming_language)) and [Scheme](http://en.wikipedia.org/wiki/Scheme_(programming_language)). It can also be used to develop packages for the software [Mathematica](http://en.wikipedia.org/wiki/Mathematica). Development environments include the Eclipse Java development tools (JDT) for Java, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

The initial [codebase](http://en.wikipedia.org/wiki/Codebase) originated from Age. The Eclipse SDK (which includes the Java development tools) is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Android Development Tools (ADT) is a plug-in for the Eclipse IDE that is designed to give users a powerful, integrated environment in which to build Android applications.

ADT extends the capabilities of Eclipse to let users quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute applications.

Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides tools integration, custom XML editors and debug output pane. ADT gives users an incredible boost in developing Android applications.

**4.2.3 Advantages of Android SDK**

* + 1. Ability to collect exact information at a fast rate.
    2. Significantly concentrated cycle.
    3. Uses of tools are very simple.
    4. Browser based on WebKit engine Integration.
    5. Memory and performance.
    6. Rich development environment.
    7. Supports audio, video and image formats.
    8. Support for extra-large screen sizes and resolutions
    9. JVM requires lot of memory and in mobile resources are very limited; therefore it is not possible to have all classes and methods which are present in core [java](http://www.javaranch.com/) used in Mobile applications. DVM outshines JVM here.
    10. Android SDK is platform independent.

4.3 FUNCTIONAL REQUIREMENTS

1. Effective filtering mechanism to provide the best possible POIs to users.
2. Efficient overlaying of information about POIs.
3. Simple and intuitive user interface.

## 4.4 PERFORMANCE REQUIREMENTS

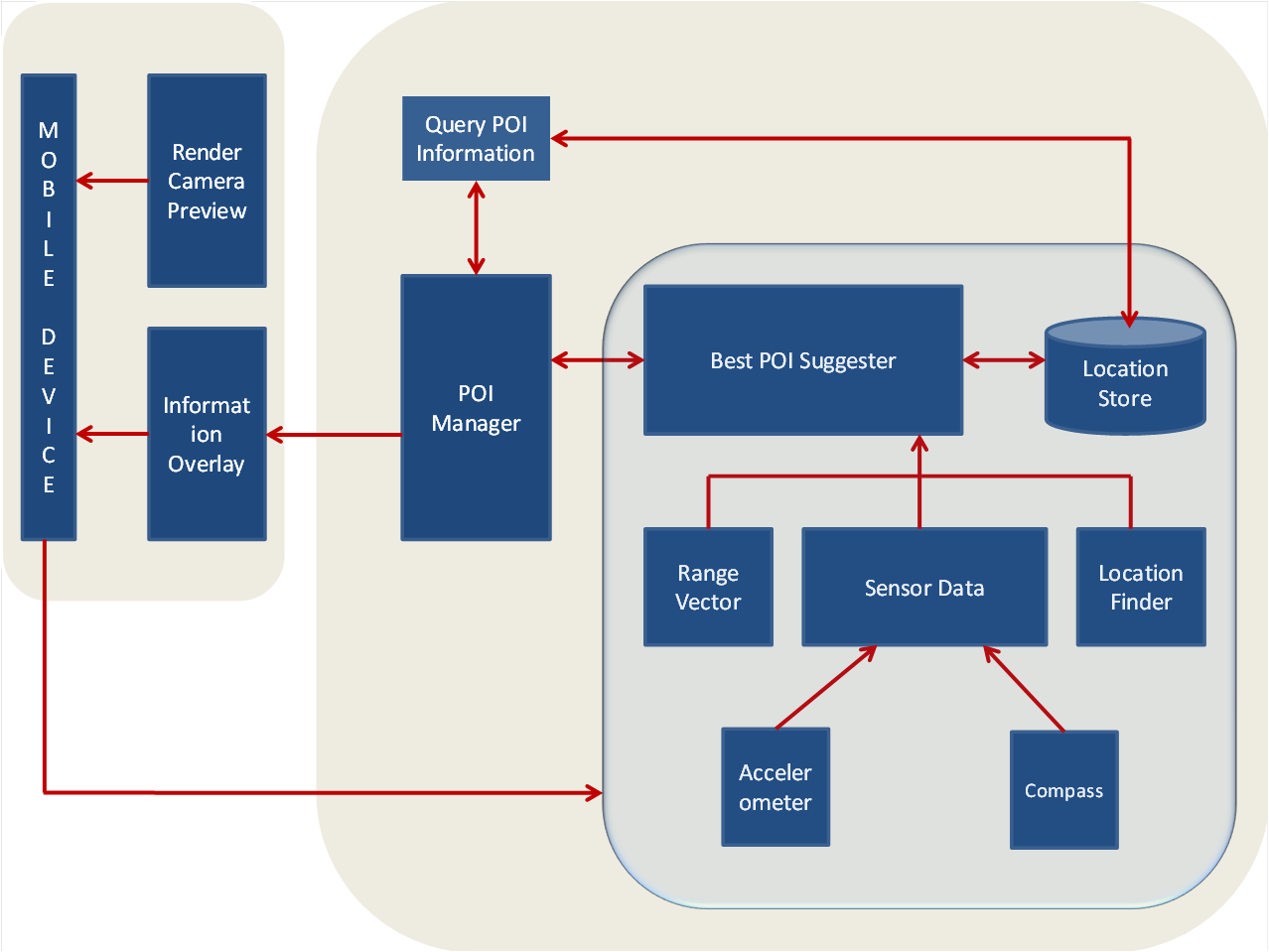
1. Once the GPS is active the Points of Interests should be found and displayed quickly without any delay.
2. Transition from one POI to another must be fast and smooth.
3. Proper rendering of camera without any non-aligned overlaying of information.
4. Application should not take up excessive memory space in the mobile device.

**CHAPTER 5**

**DESIGN PROCESS**

**5.1 ARCHITECTURE DIAGRAM**

The basic architectural diagram of the application is as follows



**FIGURE 5.1:** Architecture Diagram

When a user opens the application, data like location, sensor values, range are obtained. Current location of the user is obtained through GPS and the sensor values like the direction the phone faces, angle, orientation is obtained. Now the application has the necessary data to track the user and find his/her Point of Interest. The obtained data are sent to Best POI Suggester which in turn uses the Location Store to get the best possible location(s) for the information gathered by querying it. Now the best location(s) found is sent to the POI Manager which in turn will query the result obtained to get information about the found location(s) from the Location Store. Once this information is obtained the whole result is to be presented to the user. The information acquired is overlaid on the camera of the device after rendering it.

**5.2** **MODULES**

There are totally 5 modules for this project. They are Tracker, Best POI Suggester, POI Manager, AR Delivery and Deployment, Interaction module.

**5.2.1 TRACKER**

This subsystem is responsible for responding to changes in the user’s location and orientation so that digital objects can be superimposed on the reality view in a way that convinces the user that the digital object is a part of the natural environment. For Smartphone AR application, tracking is typically based on location sensors in the device such as GPS, compass and accelerometer allowing 6 degrees of freedom in displaying a digital object. The GPS data is collected along with sensor data where we get the direction faced by the mobile phone.

* **INPUT:**
* Sensor data, Location data.
* **OUTPUT:**
* Normalized Geo Coordinates.

**Flow chart**

Start

GET the sensor data from compass and accelerometer.

Find the geo co-ordinates via GPS.

Determine the location.

Report the tracker results.

Stop

**5.2.2 BEST POI SUGGESTER**

This sub system considers the sensor data and geo information, deals with a list of possible Points of Interests. The subsystem then performs a thorough analysis on the available data and produces the best possible Points of Interests (POIs).

Once the location and sensor data are obtained, Best POI Suggester compares the results with the data in Location Store. On comparison with the data stored, this module gives the best possible Points of Interests.

The Best Possible POIs are found by analyzing the Geo Coordinates that is the latitude and longitude and the direction obtained from the sensor. The filtering is done based on the direction the device is facing and the location found.

Once the Location and direction are found the Best POI Suggester can zero in on the most apt Points of Interests.

* **INPUT:**
* Geo Coordinates based on Sensor and Location data.
* **OUTPUT:**
* Best Possible Points of Interests.

**Flow chart**

Start

Get the results from the TRACKER module

Compare the data with the LBS STORE to fetch the BEST POI

If the POI is BEST

NO

YES

Feed the Result to the POI manager

Stop

**5.2.3 POI MANAGER**

POI Manager does the task of getting the best Point of Interest from the Best POI suggester. Once the best POI is obtained, it is queried into a store. The metadata of the POI is obtained from the store. The metadata contains information about the chosen Point of Interest.

* **INPUT:**
* Best Possible Points of Interests.
* **OUTPUT:**
* Metadata of the given Points of Interests.

**Flow chart**

Start

GET the best POI through best poi suggester

Retrieve information about the best POI.

Stop

**5.2.4 AR Delivery and Deployment**

This subsystem is responsible for all output, especially text and image outputs. Output depends on the Point of Interest and its metadata. The metadata is augmented to the real world thus enhancing it. This module helps in overlaying information on the camera thus augmenting the real world.

* **INPUT:**
* POI and its metadata..
* **OUTPUT:**
* Overlaid information.

**Flow chart**

Start

GET the metadata of the BEST POI

Overlay the augmented information on the mobile device

Stop

**5.2.5 Interaction**

This subsystem provides access to augmented representation of the real world, including points of interests, objects and metadata. The user interacts with the world augmented with objects and has the option to view information about a particular Point of Interest within a range.

This module allows user to define his/her preferences on how he/she wants the augmented information to be and the number of POIs to be considered.

**Flow chart**

Start

Display

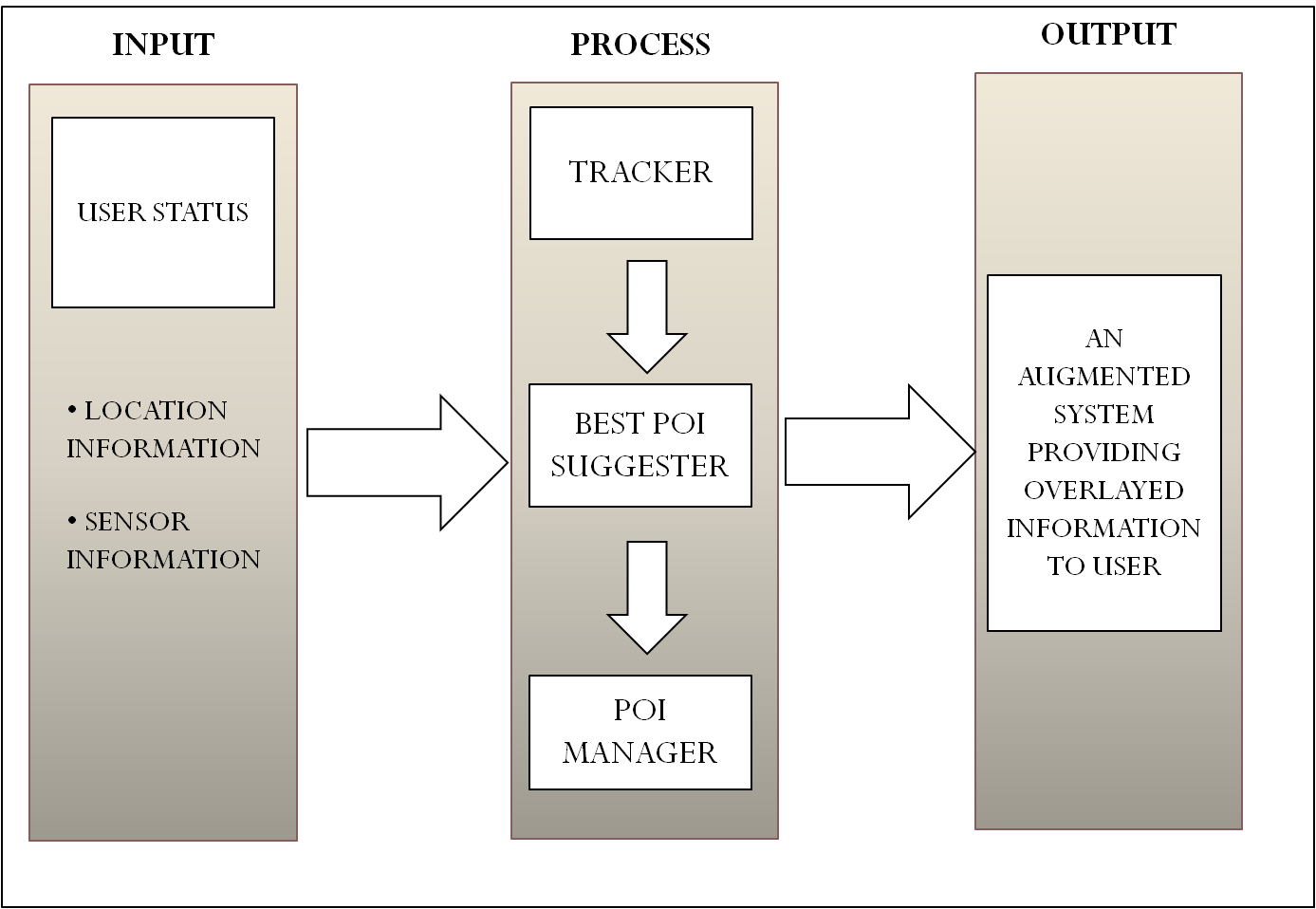
the preferences

Provide Location Preferences

Provide AR Preferences

Stop

**5.3 DATA FLOW DIAGRAM**

****

The above diagram shows the Data flow of the application. The Location and Sensor information are obtained through GPS and built in sensors of the mobile device. This data is used to track the current location of the user. Tracker is responsible to collect these data and is processed by Best POI Suggester in turn to get best results.

 The POI Manager then uses the result of POI sugester to get information about the results. Once the final results are obtained it is overlaid on the camera to give user a realistic view

**5.4 OVERVIEW OF USER INTERFACE**

When the application is started, the phone camera opens enabling the user to point to the place he/she wishes to pull up information about. On receiving the required data via GPS and sensors, the user is provided with the best possible POIs which are overlaid on the camera preview thus giving a realistic view. The POIs are placed at the centre of the screen pleasing the user.

To get information about the POI, the user has to click on the desired POI displayed. Once clicked the information about the place is displayed at the bottom of the screen. The view containing the information has a close button which helps the user to close the information view when not required.

Apart from these, when the menu button is pressed, user gets two options

* Settings
* Help

Settings when clicked provides user with two list boxes where users can set location radius and angle to find POIs within a particular range.This provides users the flexibility to distinguish between foreground and obscured buildings.

Help contains information about how to use the application and a few tips to help new users.

**CHAPTER 6**

**IMPLEMENTATION and RESULTS**

**6.1 IMPLEMENTATION**

The Android application developed can be implemented on all mobile phones running on Android 2.3.3 and above with GPS enabled. The executable file (.apk) of the project can be directly loaded into the mobile phone, installed and the application can be executed.

The camera is quickly enabled as soon as the application is launched which then proceeds with the tracking function.

Firstly accurate tracking is important during implementation especially for a moving user, the system would have to constantly determine the position within the environment of the user surrounding the virtual object, this because the computer generated object should appear to be fixed.

The location of the user is found using GPS to track where a mobile device is situated. Then internal sensors fixed to mobile devices are used to determine the direction the device faces. Thus it has made tracking the location of the user, as well as the direction that the device is pointing a possibility, and thereby superimposing the augmented information with respect to location in real time interaction.

**6.2 RESULTS**

**6.2.1 Augmented Reality**

The markers are placed at the centre of the screen and there is no overlapping when the number of markers increases. The information to be overlaid lies within the screen and the description about places does not overlap with the markers or place name. The application was tested on various Android devices with different screen resolutions such as Sony Ericsson Xperia Arc, Sony Ericsson Xperia Mini, Samsung Galaxy Y and Samsung Galaxy Ace and augmented reality was found to be working smoothly.

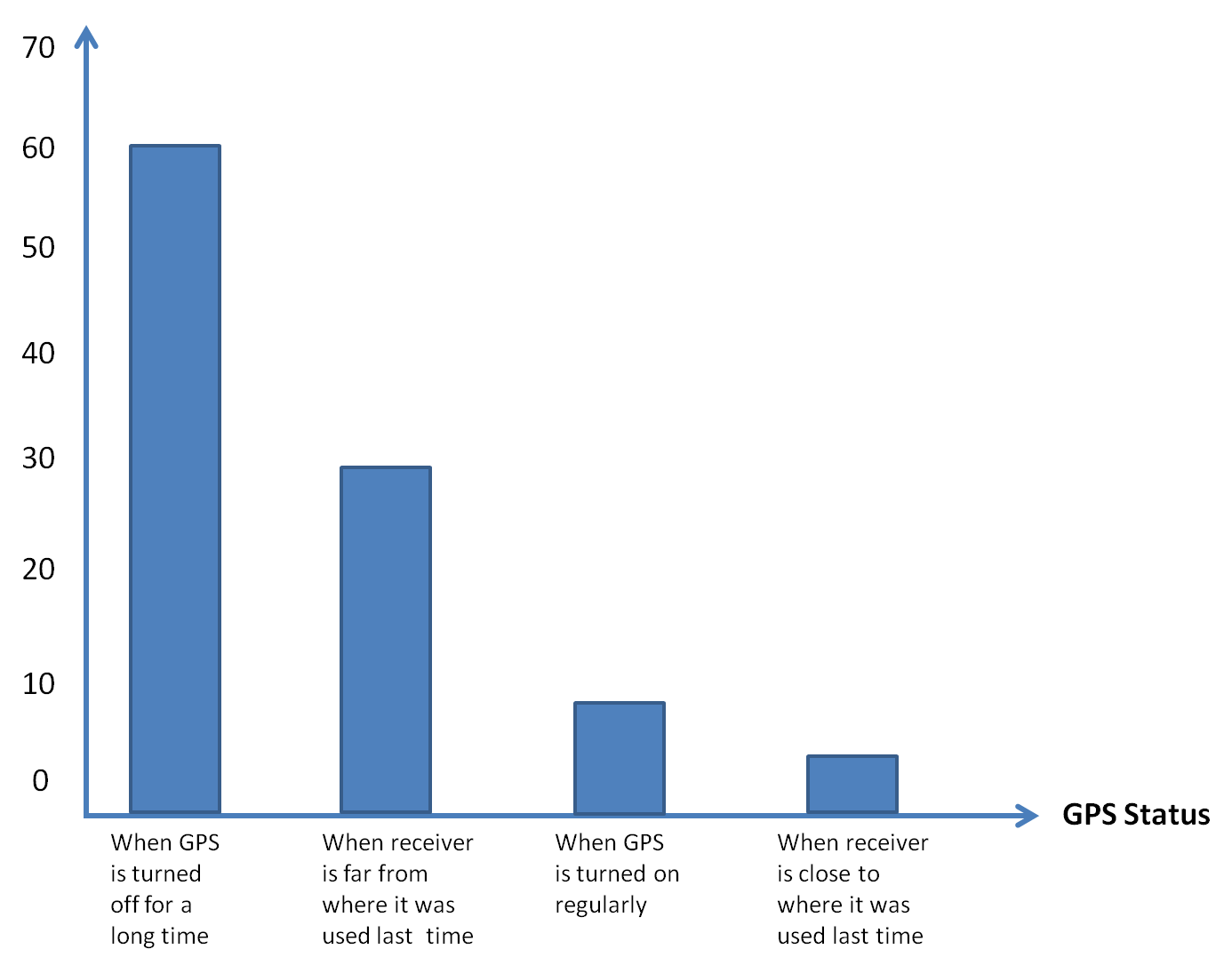




**6.2.2 Localization**

This application finds the user’s current location by determining the geo coordinates and the direction the device is facing. Figure shows the time taken to track user’s location under different circumstances.

There is time variation under different circumstances because when a GPS receiver is switched on, it needs to gather data from at least three or four GPS satellites before it can report its position. This usually takes about a minute, but it might take longer if the receiver is far from where it was last used or it's been off for a very long time.

****

**Time in seconds**

**6.2.3 Interest Point Detection**

Once the location and sensor data are obtained the POIs are detected spontaneously. On analysis it is found that the application takes a maximum of 3 to 5 seconds to find best possible POIs.

The location radius and angle preferences can be used to produce better results. Figure shows the ability of the application to discriminate between foreground and obscured buildings.



****

**CHAPTER 7**

**PERFORMANCE AND CONTRIBUTIONS**

**7.1 METRICS USED**

* **Range**:

It refers to the specific area until which the Points of Interests are listed, beyond which all available POIs are neglected. Users are provided the option to set their preferred value through settings menu.

* **Compass**:

It measures directions in a frame of reference that is stationary relative to the surface of the earth.  The frame of reference defines the four [cardinal directions](http://en.wikipedia.org/wiki/Cardinal_direction) (or points) [north](http://en.wikipedia.org/wiki/North), [south](http://en.wikipedia.org/wiki/South), [east](http://en.wikipedia.org/wiki/East), and [west](http://en.wikipedia.org/wiki/West). Intermediate directions are also defined. This helps users find the direction they face and also the direction at which Points of Interests are located.

* **Angle:**

It refers to the specific angle at which Points of Interests are to be determined from the current location and the direction faced by the device. Users are provided the option to set their preferred value through settings menu.

**7.2 CONTRIBUTION OF WORK**

* Our application works on any android mobile device/ tablet (GPS enabled) that runs on Android 2.3.3 and above.
* Primary emphasis of our system is mostly concentrated on providing aid through visual enhancement.
* The augmented description about the location / objects solves the user’s uncertainty on the location/object.

**CHAPTER 8**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**8.1** **CONCLUSION**

Our application takes college touring to a new level using Augmented Reality, which is getting popular every day. Simply pointing your camera at objects around you and ﬁnding information about them provides a very intuitive user interface for accessing information around one’s current location. Such a system provides a bridge between the digital and physical worlds. By providing Location Radius to filter out places, this application solves some of the common problems faced by location based augmented reality applications for mobile.

**8.2 FUTURE ENHANCEMENTS**

Adding a map to this application and enabling user to view and pull up information about buildings off campus can be a worthy extension to the project. This framework can be developed for other leading mobile operating systems too.

**ANNEXURE**

**A.1 SOURCE CODE**

**DbHelper.java**

package wa.places.data;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import android.content.Context;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteException;

import android.database.sqlite.SQLiteOpenHelper;

public class DbHelper extends SQLiteOpenHelper implements IDBConstants{

private SQLiteDatabase myDataBase;

private final Context myContext;

public DbHelper(Context context) {

super(context, DBNAME, null, 1);

this.myContext = context;

}

public void createDataBase() throws IOException{

boolean dbExist = checkDataBase();

if(dbExist){

}else{

this.getWritableDatabase();

try {

copyDataBase();

} catch (IOException e) {

throw new Error("Error copying database");

}

}

}

public void overWriteDataBase() throws IOException{

this.getWritableDatabase();

try {

copyDataBase();

} catch (IOException e) {

throw new Error("Error copying database");

}

}

private boolean checkDataBase(){

SQLiteDatabase checkDB = null;

try{

String myPath = DBPATH + DBNAME;

checkDB = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN\_READWRITE);

}catch(SQLiteException e){

}

if(checkDB != null){

checkDB.close();

}

return checkDB != null ? true : false;

}

private void copyDataBase() throws IOException{

InputStream myInput = myContext.getAssets().open(DBNAME);

String outFileName = DBPATH + DBNAME;

OutputStream myOutput = new FileOutputStream(outFileName);

byte[] buffer = new byte[1024];

int length;

while ((length = myInput.read(buffer))>0){

myOutput.write(buffer, 0, length);

}

myOutput.flush();

myOutput.close();

myInput.close();

}

public SQLiteDatabase openDataBase() throws SQLException, IOException{

createDataBase();

String myPath = DBPATH + DBNAME;

myDataBase = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN\_READWRITE);

return myDataBase;

}

@Override

public synchronized void close() {

if(myDataBase != null)

myDataBase.close();

super.close();

}

@Override

public void onCreate(SQLiteDatabase db) {

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

}

}

**DirectionData.java**

package wa.places.data;

import android.content.Context;

import android.hardware.SensorManager;

public class DirectionData {

private double azimuth;

private double pitch;

private double roll;

private boolean directionFound = false;

private Context ctx;

public DirectionData()

{

}

public void findPhoneOrientation(float[] accelerometerVals, float[] magneticVals)

{

float inR[] = new float[9];

float outR[] = new float[9];

float I[] = new float[9];

float orientation[] = new float[3];

SensorManager.getRotationMatrix(inR, I, accelerometerVals, magneticVals);

SensorManager.remapCoordinateSystem(inR, SensorManager.AXIS\_X, SensorManager.AXIS\_Z, outR);

SensorManager.getOrientation(outR, orientation);

azimuth = Math.round(Math.toDegrees(orientation[0]));

azimuth = (azimuth + 360)%360;

pitch = Math.round(Math.toDegrees(orientation[1]));

roll = Math.round(Math.toDegrees(orientation[2]));

directionFound = true;

}

public double getAzimuth()

{

return azimuth;

}

public double getPitch()

{ return pitch;

}

public double getRoll()

{

return roll;

}

public boolean isDirectionFound()

{

return directionFound;

}

}

**IDBConstants.java**

package wa.places.data;

public interface IDBConstants {

public static final String DBNAME = "watsarnd";

public static final String PLACETABLE = "PLACE";

public static final int DATABASE\_VERSION = 1;

public static final String INTEGERQUERYQUALIFIER = "integer primary key autoincrement";

public static final String TEXTNONULLQUERYQUALIFIER = "text not null";

public static final String TEXTQUERYQUALIFIER = "text";

public static final String COMMA = ",";

public static final String DBPATH = "/data/data/wa.places.main/databases/";

}

**IGetPlacesCallBack.java**

package wa.places.data;

import java.util.List;

import android.location.Location;

public interface IGetPlacesCallBack {

public void update(List<Location> places);

}

**ILocationConstants.java**

package wa.places.data;

public interface ILocationConstants {

public static String \_id = "\_id";

public static String name = "name";

public static String desc = "description";

public static String types = "types";

public static String vicinity = "vicinity";

public static String lat = "lattitude";

public static String lng = "longitude";

public static String accuracy = "accuracy";

public static String bearing = "bearing";

public static String altitude = "altitude";

public static String speed = "speed";

public static String time = "time";

}

**PlaceManager.java**

package wa.places.data;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.SQLException;

import android.database.sqlite.SQLiteDatabase;

import android.location.Location;

import android.location.LocationManager;

import android.os.Bundle;

public class PlaceManager implements ILocationConstants{

private SQLiteDatabase database;

private DbHelper dbHelper;

private String allColumns[] = new String[]{\_id, name, desc, types, vicinity,lat,lng,accuracy,bearing,altitude,speed,time};

public PlaceManager(Context context) {

dbHelper = new DbHelper(context);

}

public void open() throws SQLException, IOException {

database = dbHelper.openDataBase();

}

public void close() {

dbHelper.close();

}

public Collection<Location> getPlaces(double direction, Location currentLocation, float radius)

{

List<Location> places = getPlaces();

double currentLat = currentLocation.getLatitude();

double currentLng = currentLocation.getLongitude();

float result[] = new float[3];

Map<String, Location> filteredPlacesMap = new HashMap<String, Location>();

for(Location loc: places)

{

Location.distanceBetween(currentLat, currentLng, loc.getLatitude(), loc.getLongitude(), result);

double locAzimuth = (result[1] + 360)%360;

String key = loc.getExtras().getString(ILocationConstants.name);

if(!((locAzimuth < (direction -20)) || (locAzimuth > (direction + 20))) && (result[0]<=radius))

{

filteredPlacesMap.put(key, loc);

}

}

return filteredPlacesMap.values();

}

public List<Location> getPlaces()

{

List<Location> places = new ArrayList<Location>();

Cursor placesCursor = database.query(IDBConstants.PLACETABLE, allColumns, null, null, null, null, null);

int rowCount = placesCursor.getCount();

for(int i=0; i<rowCount; i++)

{

placesCursor.moveToNext();

places.add(cursorToLocation(placesCursor));

}

placesCursor.close();

return places;

}

private Location cursorToLocation(Cursor c)

{

Location loc = new Location(LocationManager.GPS\_PROVIDER);

loc.setExtras(new Bundle());

loc.getExtras().putString(vicinity, c.getString(c.getColumnIndex(vicinity)));

loc.getExtras().putString(types, c.getString(c.getColumnIndex(types)));

loc.getExtras().putString(name, c.getString(c.getColumnIndex(name)).trim());

loc.getExtras().putString(desc, c.getString(c.getColumnIndex(desc)));

loc.getExtras().putString(\_id, c.getString(c.getColumnIndex(\_id)));

loc.setLatitude(Double.valueOf(c.getString(c.getColumnIndex(lat))));

loc.setLongitude(Double.valueOf(c.getString(c.getColumnIndex(lng))));

loc.setAccuracy(Float.valueOf(c.getString(c.getColumnIndex(accuracy))));

loc.setTime(Long.parseLong(c.getString(c.getColumnIndex(time))));

loc.setAltitude(Double.valueOf(c.getString(c.getColumnIndex(altitude))));

loc.setBearing(Float.valueOf(c.getString(c.getColumnIndex(bearing)))); loc.setSpeed(Float.valueOf(c.getString(c.getColumnIndex(speed))));

return loc;

}

public String addPlace(Location place)

{

long insertId = database.insert(IDBConstants.PLACETABLE, null, locationToContentValues(place));

return String.valueOf(insertId);

}

private ContentValues locationToContentValues(Location place)

{

ContentValues values = new ContentValues();

values.put(lat, place.getLatitude());

values.put(lng, place.getLongitude());

values.put(accuracy, place.getAccuracy());

values.put(time, place.getTime());

values.put(altitude, place.getAltitude());

values.put(bearing, place.getBearing());

values.put(speed, place.getSpeed());

values.put(vicinity, place.getExtras().getString(vicinity));

values.put(types, place.getExtras().getString(types));

values.put(name, place.getExtras().getString(name));

values.put(desc, place.getExtras().getString(desc));

return values;

}

}

**MainActivity.java**

package wa.places.main;

import java.io.IOException;

import java.util.Collection;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

import wa.places.data.DirectionData;

import wa.places.data.ILocationConstants;

import wa.places.data.PlaceManager;

import wa.places.main.cameraview.CameraPreview;

import wa.places.main.subviews.HelpActivity;

import wa.places.main.subviews.SettingsActivity;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.database.SQLException;

import android.graphics.Color;

import android.graphics.Typeface;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorManager;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.os.Bundle;

import android.preference.PreferenceManager;

import android.util.DisplayMetrics;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.RelativeLayout;

import android.widget.RelativeLayout.LayoutParams;

import android.widget.TextView;

import android.widget.Toast;

public class MainActivity extends Activity implements LocationListener, SensorEventListener, OnClickListener{

private Location lastKnownLocation;

private LocationManager locManager;

private int locationUpdateCount = 0;

private DirectionData direction;

private SensorManager sensorManager;

private float[] accelerometerVals;

private float[] magneticVals;

private RelativeLayout placeView;

private RelativeLayout infoView;

private Map<String, Location> loc\_idMap = new HashMap<String, Location>();

private LayoutInflater inflater;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.mainview);

locManager = (LocationManager) getSystemService(Context.LOCATION\_SERVICE);

sensorManager = (SensorManager) getSystemService(Context.SENSOR\_SERVICE);

RelativeLayout rootLayout = (RelativeLayout)findViewById(R.id.mainview);

rootLayout.addView(new CameraPreview(this));

inflater = (LayoutInflater) this.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

View azimuthView = inflater.inflate(R.layout.azimuthview, null);

azimuthView.bringToFront();

rootLayout.addView(azimuthView);

placeView = (RelativeLayout)inflater.inflate(R.layout.placelayout, null);

placeView.bringToFront();

LayoutParams placeLayout = new LayoutParams(LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

placeLayout.addRule(RelativeLayout.CENTER\_IN\_PARENT);

placeView.setLayoutParams(placeLayout);

rootLayout.addView(placeView);

infoView=(RelativeLayout)inflater.inflate(R.layout.infolayout, null);

infoView.bringToFront();

LayoutParams infolayout = new LayoutParams(LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

infolayout.addRule(RelativeLayout.ALIGN\_PARENT\_BOTTOM);

infoView.setLayoutParams(infolayout);

rootLayout.addView(infoView);

requestLocUpdate();

}

@Override

public void onBackPressed() {

super.onBackPressed();

locManager.removeUpdates(this);

unregisterSensors();

}

@Override

protected void onDestroy() {

super.onDestroy();

locManager.removeUpdates(this);

unregisterSensors();

}

@Override

protected void onPause() {

super.onPause();

locManager.removeUpdates(this);

unregisterSensors();

}

@Override

protected void onResume() {

super.onResume();

if(checkGPSEnable())

{

locManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 0, 0, this);

registerSensors();

locationUpdateCount = 0;

}

placeView.removeAllViews();

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.placesmenu, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.help:

Intent helpintent = new Intent(this, HelpActivity.class);

startActivity(helpintent);

break;

case R.id.settings:

Intent intent = new Intent(this, SettingsActivity.class);

startActivity(intent);

break;

default:

break;

}

return true;

}

public void onLocationChanged(Location location) {

++locationUpdateCount;

if(locationUpdateCount<3)

{

if(lastKnownLocation == null || lastKnownLocation.getAccuracy()>location.getAccuracy())

{

lastKnownLocation = location;

}

}

else

{

if(direction!=null && direction.isDirectionFound())

{ locationUpdateCount=0;

SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);

String radius = prefs.getString("listPref", "100");

PlaceManager manager = null;

try {

manager = new PlaceManager(this);

manager.open();

Collection<Location> locs = manager.getPlaces(direction.getAzimuth(), lastKnownLocation, Float.valueOf(radius));

manager.close();

int prevTextId = 0;

placeView.removeAllViews();

loc\_idMap.clear();

for(Location loc:locs)

{

int currentId = Integer.parseInt(loc.getExtras().getString(ILocationConstants.\_id));

TextView placeTextView = new TextView(this);

placeTextView.setTypeface(null,Typeface.BOLD);

placeTextView.setTextColor(Color.WHITE);

placeTextView.setTextSize((float) 14);

placeTextView.setId(currentId);

placeTextView.setCompoundDrawablesWithIntrinsicBounds(0, getResources().getIdentifier("marker", "drawable", getPackageName()), 0, 0);

String name = loc.getExtras().getString(ILocationConstants.name);

placeTextView.setText(name);

LayoutParams textLayoutParams = new LayoutParams(LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

placeTextView.setLayoutParams(textLayoutParams);

if(prevTextId>0){

textLayoutParams.addRule(RelativeLayout.RIGHT\_OF, prevTextId);

textLayoutParams.leftMargin = 10;

placeTextView.requestLayout();

}

prevTextId = currentId;

placeView.addView(placeTextView);

placeTextView.setOnClickListener(this);

loc\_idMap.put(String.valueOf(currentId), loc);

}

} catch (SQLException e) {

Toast.makeText(this,e.getMessage(), Toast.LENGTH\_LONG).show();

if(manager!=null) manager.close();

} catch (IOException e) {

Toast.makeText(this, e.getMessage(), Toast.LENGTH\_LONG).show();

if(manager!=null) manager.close();

}

}

else

{

Toast.makeText(this, "Direction not found please try again", Toast.LENGTH\_LONG).show();

}

}

}

public void onProviderDisabled(String provider) {

Toast.makeText(this, "Provider " + provider + "disabled ", Toast.LENGTH\_LONG).show();

}

public void onProviderEnabled(String provider) {

Toast.makeText(this, "Provider " + provider + "turned on", Toast.LENGTH\_LONG).show();

}

public void onStatusChanged(String provider, int status, Bundle extras) {

}

private boolean checkGPSEnable()

{

boolean enable = locManager.isProviderEnabled(LocationManager.GPS\_PROVIDER);

if(!enable)

{

Toast.makeText(this, "Enable GPS", Toast.LENGTH\_LONG).show();

}

return enable;

}

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}

public void onSensorChanged(SensorEvent event) {

switch (event.sensor.getType()) {

case Sensor.TYPE\_ACCELEROMETER:

accelerometerVals = event.values.clone();

break;

case Sensor.TYPE\_MAGNETIC\_FIELD:

magneticVals = event.values.clone();

break;

default:

break;

}

if(accelerometerVals!=null && magneticVals!=null)

{

direction = new DirectionData();

direction.findPhoneOrientation(accelerometerVals, magneticVals);

TextView azimuthTextView = (TextView)findViewById(R.id.azimuthtext);

if(direction.getAzimuth()>0 && direction.getAzimuth()<=30)

{

azimuthTextView.setText("N");

}

if(direction.getAzimuth()>30 && direction.getAzimuth()<=60)

{

azimuthTextView.setText("NE");

}

if(direction.getAzimuth()>60 && direction.getAzimuth()<=115)

{

azimuthTextView.setText("E");

}

if(direction.getAzimuth()>115 && direction.getAzimuth()<=150)

{

azimuthTextView.setText("SE");

}

if(direction.getAzimuth()>150 && direction.getAzimuth()<=205)

{

azimuthTextView.setText("S");

}

if(direction.getAzimuth()>205 && direction.getAzimuth()<=240)

{

azimuthTextView.setText("SW");

}

if(direction.getAzimuth()>240 && direction.getAzimuth()<=290)

{

azimuthTextView.setText("W");

}

if(direction.getAzimuth()>290 && direction.getAzimuth()<=330)

{

azimuthTextView.setText("NW");

}

accelerometerVals = null;

magneticVals = null;

}

}

private void unregisterSensors()

{

sensorManager.unregisterListener(this, sensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER));

sensorManager.unregisterListener(this, sensorManager.getDefaultSensor(Sensor.TYPE\_ORIENTATION));

}

private void registerSensors()

{

sensorManager.registerListener(this, sensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER), SensorManager.SENSOR\_DELAY\_NORMAL);

sensorManager.registerListener(this, sensorManager.getDefaultSensor(Sensor.TYPE\_MAGNETIC\_FIELD), SensorManager.SENSOR\_DELAY\_NORMAL);

}

private void requestLocUpdate()

{

if(checkGPSEnable())

{

locManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 0, 0, this);

registerSensors();

locationUpdateCount = 0;

}

}

public void onClick(View v) {

Button infoButton=new Button(this);

ImageView close=new ImageView(this);

ImageView testImage=new ImageView(this);

close.setBackgroundResource(getResources().getIdentifier("closebutton", "drawable", getPackageName()));

Location currentLoc = loc\_idMap.get(String.valueOf(v.getId()));

infoButton.setText(currentLoc.getExtras().getString(ILocationConstants.desc));

isplayMetrics metrics = new DisplayMetrics();

getWindowManager().getDefaultDisplay().getMetrics(metrics);

infoButton.setWidth(metrics.widthPixels);

infoButton.setHeight((metrics.heightPixels)/3);

infoButton.setBackgroundColor(Color.WHITE);

LayoutParams infoLayoutParams = new LayoutParams(LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

infoButton.setLayoutParams(infoLayoutParams);

infoButton.requestLayout();

infoView.setVisibility(View.VISIBLE);

infoView.addView(infoButton);

infoButton.setTypeface(null,Typeface.BOLD);

infoView.addView(close);

locManager.removeUpdates(this);

unregisterSensors();

close.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

infoView.setVisibility(4);

requestLocUpdate();

}

}); }}

**CameraPreview.java**

package wa.places.main.cameraview;

import java.io.IOException;

import android.content.Context;

import android.hardware.Camera;

import android.util.Log;

import android.view.SurfaceHolder;

import android.view.SurfaceView;

public class CameraPreview extends SurfaceView implements

SurfaceHolder.Callback {

private SurfaceHolder mHolder=null;

private Camera mCamera;

public CameraPreview(Context context) {

super(context);

mHolder = getHolder();

mHolder.addCallback(this);

mHolder.setType(SurfaceHolder.SURFACE\_TYPE\_PUSH\_BUFFERS);

}

public void surfaceChanged(SurfaceHolder holder, int format, int width,

int height) {

if (mHolder.getSurface() == null) {  
 return;

}

try {

mCamera.stopPreview();

} catch (Exception e) {

}

try {

Camera.Parameters params = mCamera.getParameters();

params.setPreviewSize(width, height); mCamera.setPreviewDisplay(mHolder);

mCamera.startPreview();

} catch (IOException e) {

Log.d(this.toString(),

"Error starting camera preview: " + e.getMessage());

}

}

public void surfaceCreated(SurfaceHolder holder) {

try {

mCamera = Camera.open();

mCamera.setPreviewDisplay(mHolder);

mCamera.startPreview();

} catch (IOException e) {

Log.d(this.toString(),

"Error setting camera preview: " + e.getMessage());

}

}

public void surfaceDestroyed(SurfaceHolder holder) {

if (mCamera != null) {

try {

mCamera.stopPreview();

} catch (Exception ignore) {

}

try {

mCamera.release();

} catch (Exception ignore) {

}

mCamera = null; }

}

}

**PlacesOverlay.java**

package wa.places.main.cameraview;

import java.util.List;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.PixelFormat;

import android.location.Location;

import android.view.SurfaceHolder;

import android.view.SurfaceView;

import android.widget.Toast;

public class PlacesOverlay extends SurfaceView {

private SurfaceHolder mHolder;

private List<Location> places;

private Context ctx;

public PlacesOverlay(Context context, List<Location> places) {

super(context);

this.ctx = context;

mHolder = getHolder();

mHolder.setFormat(PixelFormat.TRANSLUCENT);

this.places=places;

}

@Override

public void draw(Canvas canvas) {

super.draw(canvas);

Toast.makeText(ctx, places.size(), Toast.LENGTH\_LONG).show();

}

}

**HelpActivity.java**

package wa.places.main.subviews;

import wa.places.main.R;

import android.app.Activity;

import android.content.pm.ActivityInfo;

import android.graphics.Typeface;

import android.os.Bundle;

import android.widget.TextView;

public class HelpActivity extends Activity {

TextView helpText;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_LANDSCAPE);

setContentView(R.layout.helpview);

helpText = (TextView) findViewById(R.id.textView1);

helpText.setText("CresAR is an Augmented Reality driven College guide developed exclusively for Crescent Engineering College.\n\nCresAr needs GPS to be enabled to work and CresAr does not use your data pack.\n\nThe first time you open the application it takes some time as the device has to communicate with GPS Satellites.\n\nHold the phone still pointing the place you want to get information about for atleast 3 to 5 seconds.\n\nClick on the places displayed to get description about them.\n\nClick menu->settings and you can change Location Radius to get better results.\n\n Thanks for using CresAR!!");

helpText.setTextSize((float) 14);

helpText.setTypeface(null,Typeface.BOLD);

}

public void onBackPressed() {

super.onBackPressed();

}

protected void onDestroy() {

super.onDestroy();

}}

**SettingsActivity.java**

package wa.places.main.subviews;

import wa.places.main.R;

import android.content.pm.ActivityInfo;

import android.graphics.drawable.GradientDrawable.Orientation;

import android.os.Bundle;

import android.preference.Preference;

import android.preference.Preference.OnPreferenceChangeListener;

import android.preference.Preference.OnPreferenceClickListener;

import android.preference.PreferenceActivity;

public class SettingsActivity extends PreferenceActivity implements OnPreferenceClickListener, OnPreferenceChangeListener{

private Preference listPref;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

addPreferencesFromResource(R.xml.preferences);

setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_LANDSCAPE);

listPref = (Preference) findPreference("listPref");

listPref.setOnPreferenceChangeListener(this);

}

@Override

protected void onDestroy() {

super.onDestroy();

}

public boolean onPreferenceChange(Preference preference, Object newValue) {

return true;

}

public boolean onPreferenceClick(Preference preference) {

return true;

}

}

**A.2 SCREEN SHOTS**



**Screen 1: When the application is started.**



**Screen 2: Best POI is displayed.**



**Screen 3: Information about the POI is displayed when it is clicked.**



**Screen 4: When menu is pressed.**

****

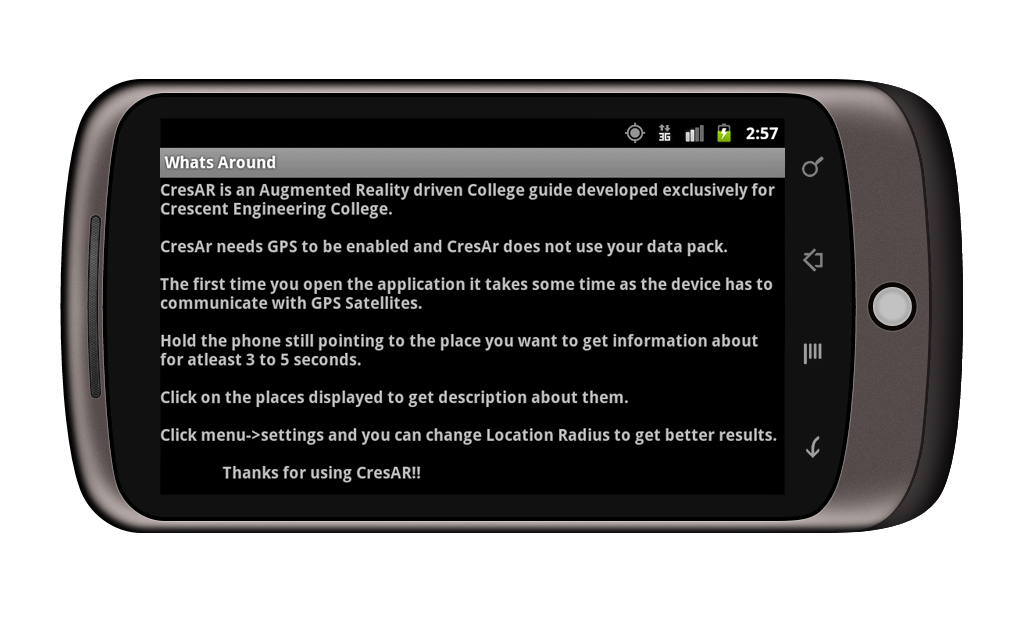
**Screen 5: When settings option is selected.**



**Screen 6: Option to select or set preferred Radius.**



**Screen 7: Option to select or set preferred Angle.**



**Screen 8: When help option is selected.**

**REFERENCES**

.

1. Mohamed El-Zayat, ‘**Augmented Reality Platform for enhancing integration of virtual objects’,** 2011.
2. Steffen Gauglitz Tobias Höllerer Matthew Turk **‘Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking‘,**Published online: 31 March 2011.
3. Gudrun Klinker, Tobias Ho¨ llerer, ‘**Special Section on the IEEE International Symposium on Mixed and Augmented Reality ( ISMAR)’,**2011
4. Shane R. Porter\_ Michael R. Marner† Ross T. Smith‡ Joanne E. Zucco§ Bruce H. Thomas,**Validating Spatial Augmented Reality for Interactive Rapid Prototyping,**IEEE 2010.
5. T. Miyashita, P. Meier, T. Tachikawa, S. Orlic , T. Eble , V. Scholz, A. Gapel, O. Gerl, S. Arnaudov, S. Lieberknecht. **‘An Augmented Reality Museum Guide’ ,** 2008.
6. Gabriel Takacs, **‘Outdoors Augmented Reality on Mobile Phone using Loxel-Based Visual Feature Organization’,** 2008.