Ain Shams University
Faculty of Computer & Information Sciences
Computer Science Department

Camera Measure

By:

Muhammed Ahmed AbdEl-Gawad [CS]
Moataz Muhammed AbdEl-Fattah [CS]
Marwan Muhammed Hasan Salem[CS]
Muhammed Abd El_Nasser Salah El_Deen[CS]
Mustafa El-Refaay Muhammed El-Sharkawy[CS]

Under Supervision of:

Prof.Dr.Mohammed Ismail Roshdy
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Dr.Salma Hamdy
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

TA.Nora Youssef
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Abstract

Traditional length measurement methods are difficult, tedious and can't be relied on when measuring things taller than human like trees, buildings and others. Camera Measure aims to facilitate this process by making it simpler and faster than traditional methods.

The app is developed using Java on Android Studio and uses smartphones cameras and sensors.

The project divided into two phases: Firstly the app is used to measure length of objects using a reference object that is known to the user or a standard reference object (ex: Coin, A4). Secondly the app is used to measure height of things taller, shorter or on eye level of the user.

Input:-

- In offline mode the user is required to input the length of the known reference object and choose the boundaries of the object.
- While in online mode the user is required to input the smartphone height and to adjust the phone at different angles according to the object required to be measured.

Output:-

- In offline mode display the object's length.
- In online mode display distance between person and object and the length of the object.

Main features:-

- Easy to use.
- Accurate by 98%.

Table of Contents

Abstract i

List of Figures iii

1- Introduction 1

1.1 Motivation 1

1.2 Problem Definition 1

1.3 Objective 1

1.4 Time Plan 2

1.5 Document Organization 2

2- Background 4

3- Analysis and Design 5

3.1 System Overview 5

3.1.1 System Architecture 5

3.1.2 System Users 9

3.2 System Analysis & Design 10

3.2.1 Use Case Diagram 10

3.2.2 Class Diagram 12

3.2.3 Sequence Diagram 13

4- Implementation and Testing 16

5- User Manual 50

6- Conclusion and Future Work 75

6.1 Conclusion 75

6.2 Future Work 75

References 76

List of Figures

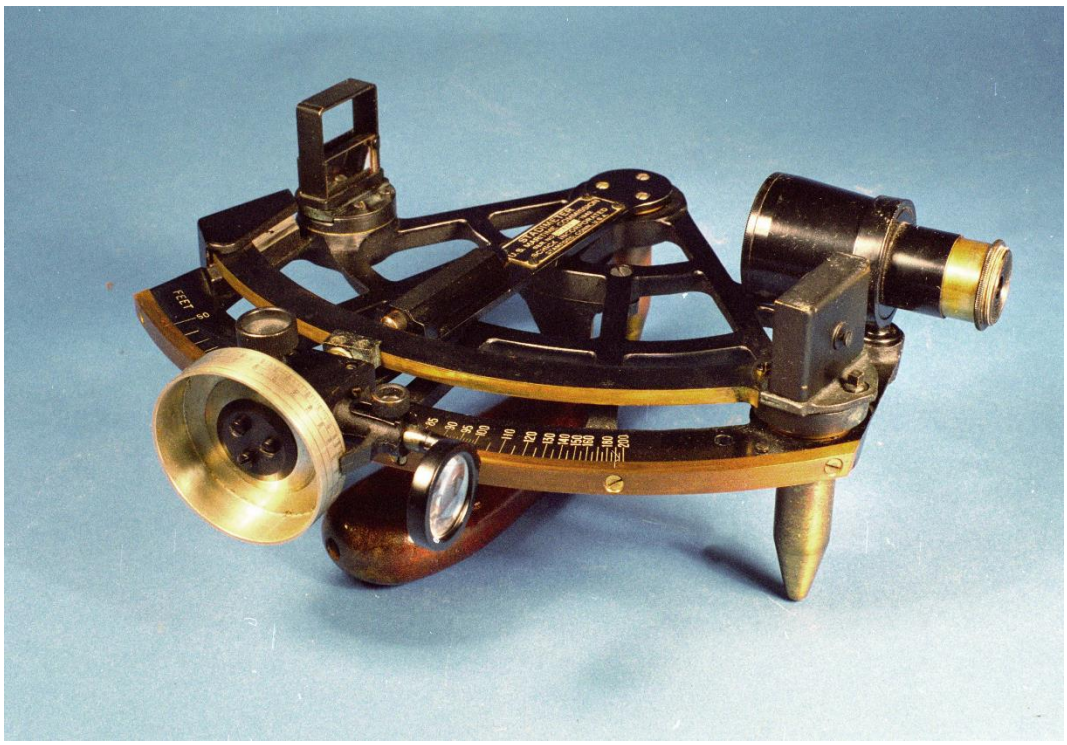


FIGURE 1- STADIMETER

FIGURE 2- LASER RANGEFINDER

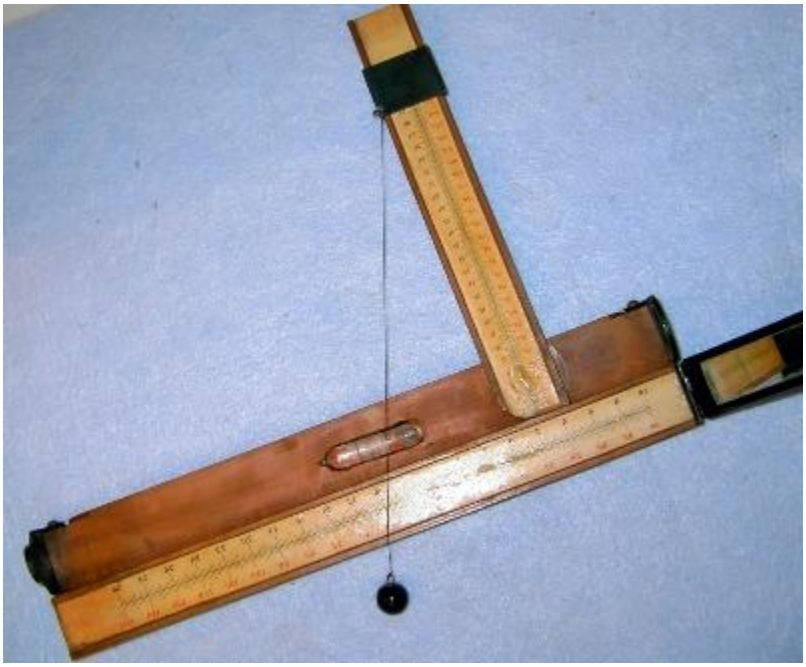


FIGURE 3- SCALE HYPSOMETER

1- Introduction

1.1 Motivation

- Facilitating the process of length measurements using an affordable and handy solution.
- Saving effort and time by making the process of length measurement much easier than using traditional methods.

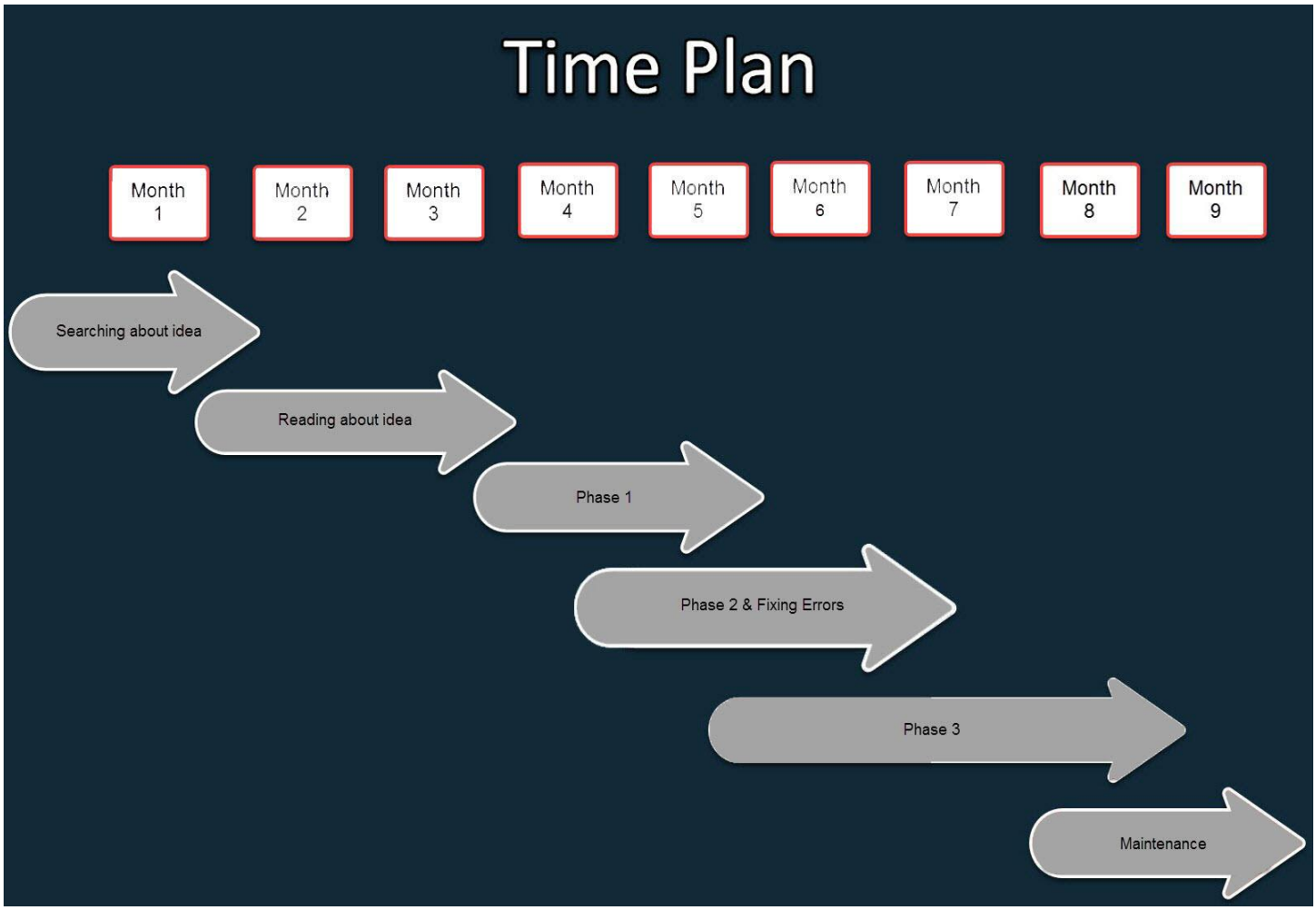
1.2 Problem Definition

- Traditional measuring methods can be really inconvenient and tiresome.
- Some measurements require more than one person.
- Some measurements can't be measured using traditional methods
  Ex: Buildings.

1.3 Objective

- Developing an app for smartphones using Android Studio and java which allows measurement of objects.
- Using smartphones' cameras for determining the objects to be measured.
- Using smartphones' sensors for determining angles required for length measurement.

1.4 Time Plan



1.5 Document Organization

Chapter 2:

This chapter introduces the following:

- A detailed description of the field of the project.
- All the scientific background related to the project.
- A survey of the work done in the field.
- Description of existing similar systems.
- Description of any technology used: Bluetooth, GPS… etc.

Chapter 3:

This chapter introduces the following:

- System overview diagrams.
- System analysis & design diagrams.

Chapter 4:

This chapter includes:

- A detailed description of all the functions in the system.
- A detailed description of all the techniques and algorithms implemented.
- Description of any new technologies used in implementation.
- UI Design and Wireframes
- Testing procedures and levels used

Chapter 5:

This chapter should describe in details how to operate the project along with screenshots of the project representing all steps.

Chapter 6:

This chapter includes:

- A complete summary of the whole project along with the results obtained.
- What can be done in the future to improve the performance of the project and what additional functions could be added?

2- Background

- The project uses Image processing Algorithms like Region Growing for object selection in offline mode and uses Trigonometric calculations in online mode.

- Scientific backgrounds include: Image Processing course, Data Structure course and Trigonometry.

- Work has been done on the measurement using smart phones but it wasn't compatible with all types of length measurement.
  Ex: Smart Measure App, Ruler App.

- **Existing similar systems are:**
  - Stadimeter: an optical device for estimating the range to an object of known height by measuring the angle between the top and bottom of the object as observed at the device (see Fig. 1):.

  - Laser rangefinder: rangefinder which uses a laser beam to determine the distance to an object. The most common form of laser rangefinder operates on the time of flight principle by sending a laser pulse in a narrow beam towards the object and measuring the time taken by the pulse to be reflected off the target and returned to the sender (see Fig. 2).

  - Scale Hypsometer: uses a combination of laser rangefinder and clinometer to measure distances to the top and bottom of objects, and the angle between the lines from the observer to each to calculate height (see Fig. 3).

- Technologies used are smartphone's camera, accelerometer sensor, gravity sensor and gyroscope sensor.

3- Analysis and Design

3.1 System Overview

3.1.1 System Architecture

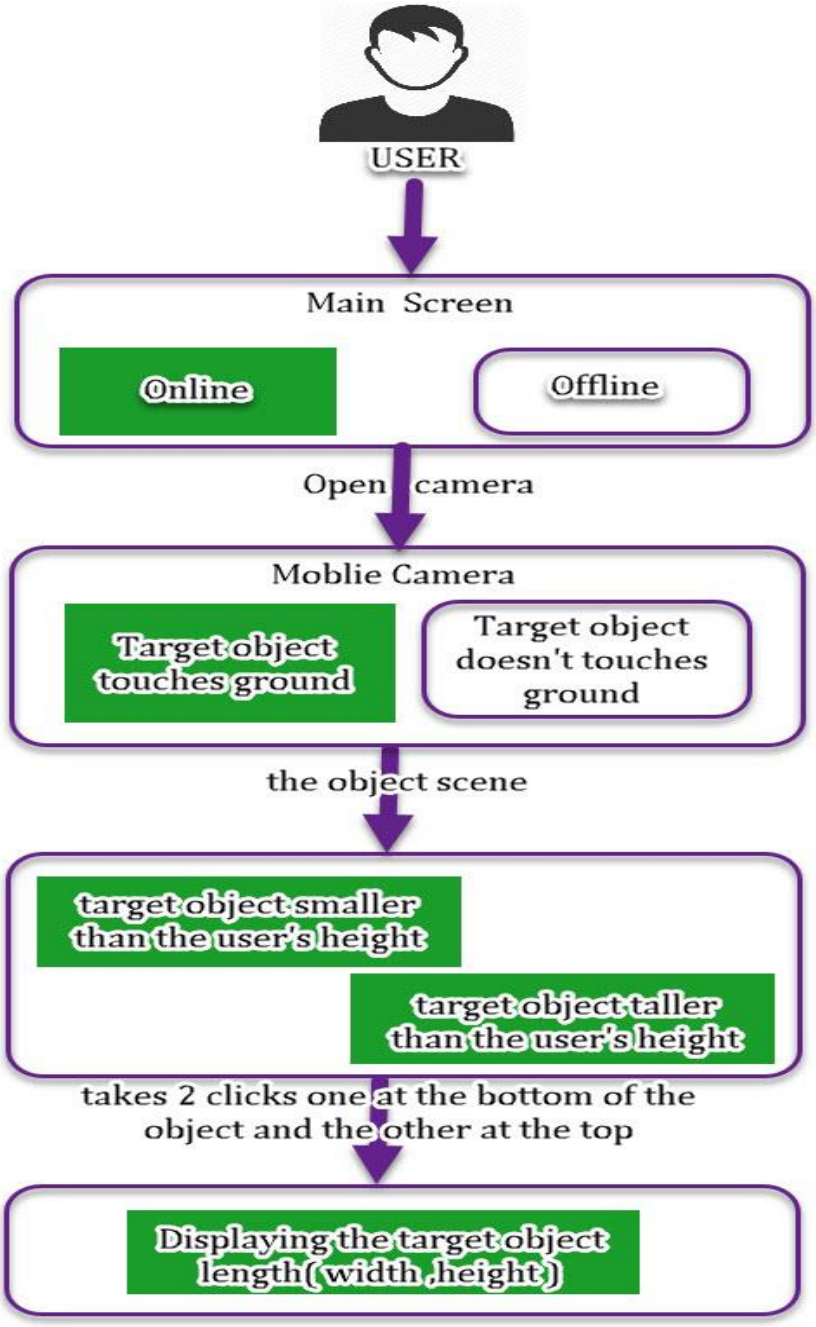


FIGURE 2 : ON-LINE MODE (TARGET OBJECT TOUCHES GROUND)

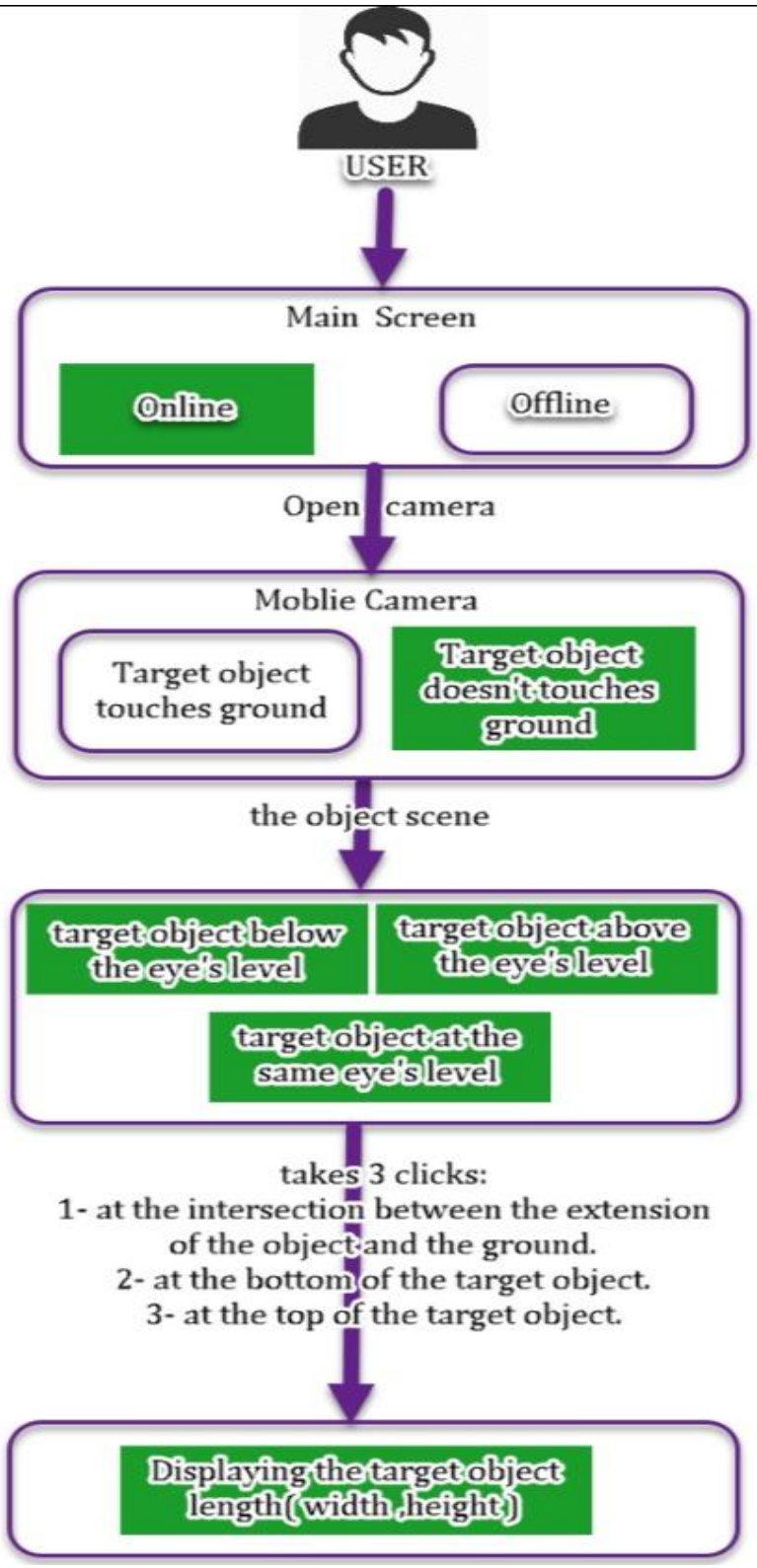


FIGURE 3 : ON-LINE MODE (TARGET OBJECT DOESN'T TOUCHES GROUND)

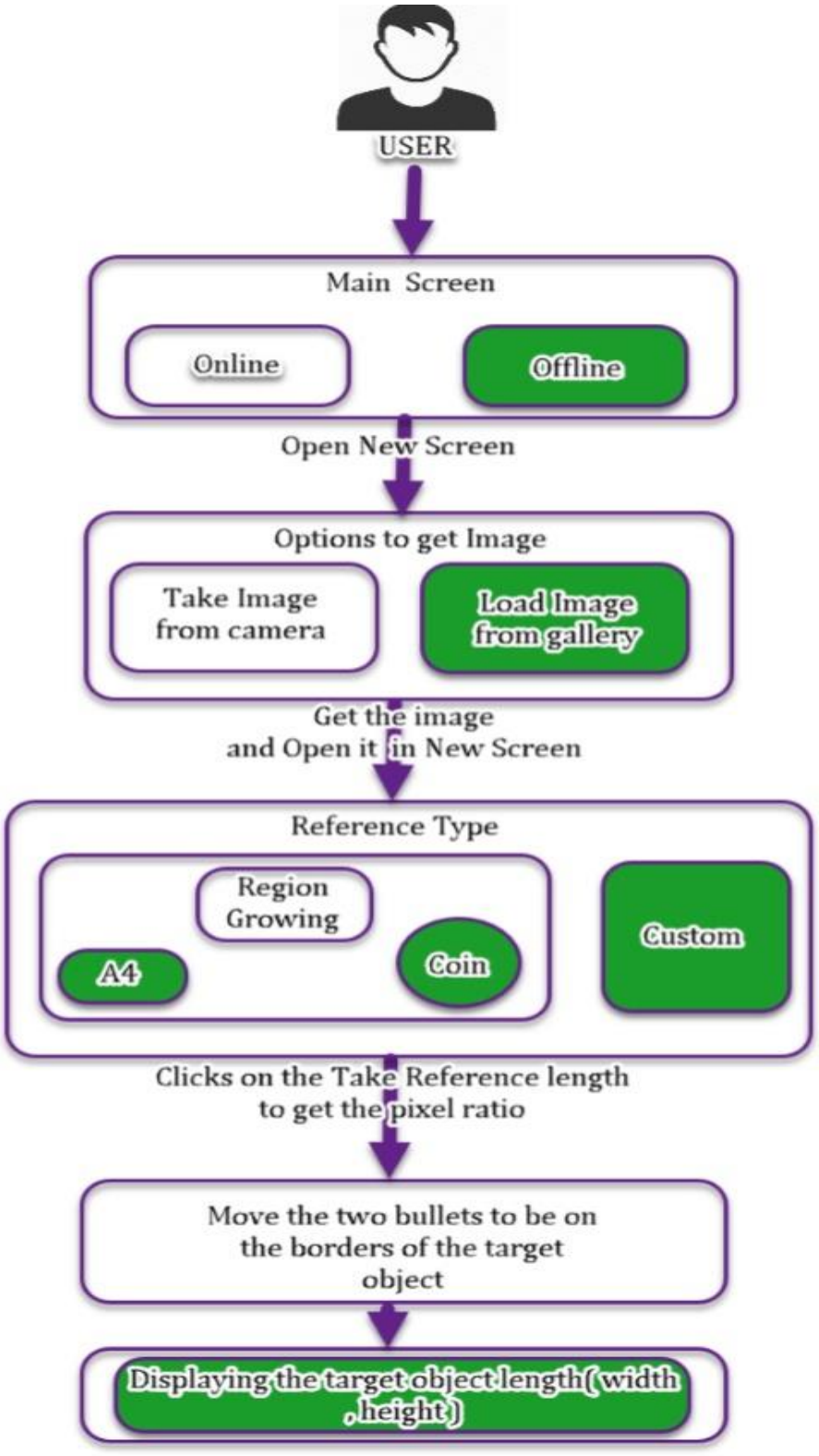


FIGURE 4: OFF-LINE MODE

Functional Requirements:

1. Application shall provide online & offline methods for measuring.

2. Offline method
    a. Application shall provide choose image form gallery and capture image from camera.
    b. Application shall provide automatic boundary selection of known reference objects (coin & A4).
    c. Application shall provide free selection of user's known length reference object.
    d. Application shall provide different measuring units (CM&INCH)
    e. Application shall provide saving image with measures.

3. Online method
    a. Application shall provide capability of measuring object's base that not start from ground.
        i. Application shall provide height of object from ground in these cases:
    b. Application shall provide length of object.
    c. Application provide measures in these cases:
        i. At eye's level.
        ii. Above eye's level.
        iii. Below eye's level.
    d. Application shall provide distance from user to object.

Non-Functional Requirements:

- **Product Requirements:**
  - The system shall be easy to use and help the user to get what he need.
  - The System shall be maintained and reliable.

- **Organizational Requirements:**
  - The system can be used by android users.
  - The system develops traditional methods of measuring.

- **External Requirements:**
  - This System Can Available for android Operating Systems.
  - The system shall be free.

3.1.2 System Users

A. Intended Users:

- Civil engineers can measure height of various buildings
- Any person who is concerned with measuring objects length.
- Application is built for android users.

B. User Characteristics

No experience or skills are required from the users all you need is the height of smartphone from ground to be able to operate the application effectively.

3.2 System Analysis & Design

3.2.1 Use Case Diagram

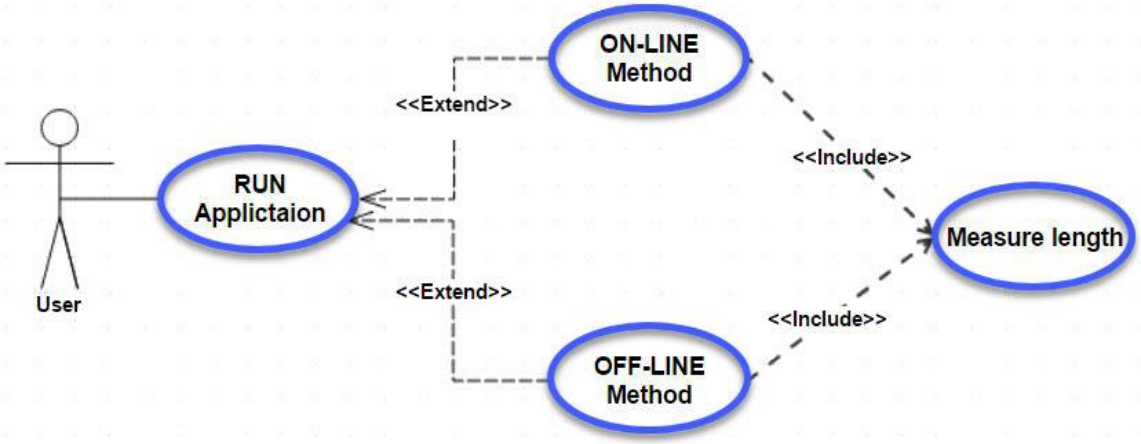


FIGURE 5 : DESCRIBING THE WHOLE APPLICATION FUNCTIONALITY

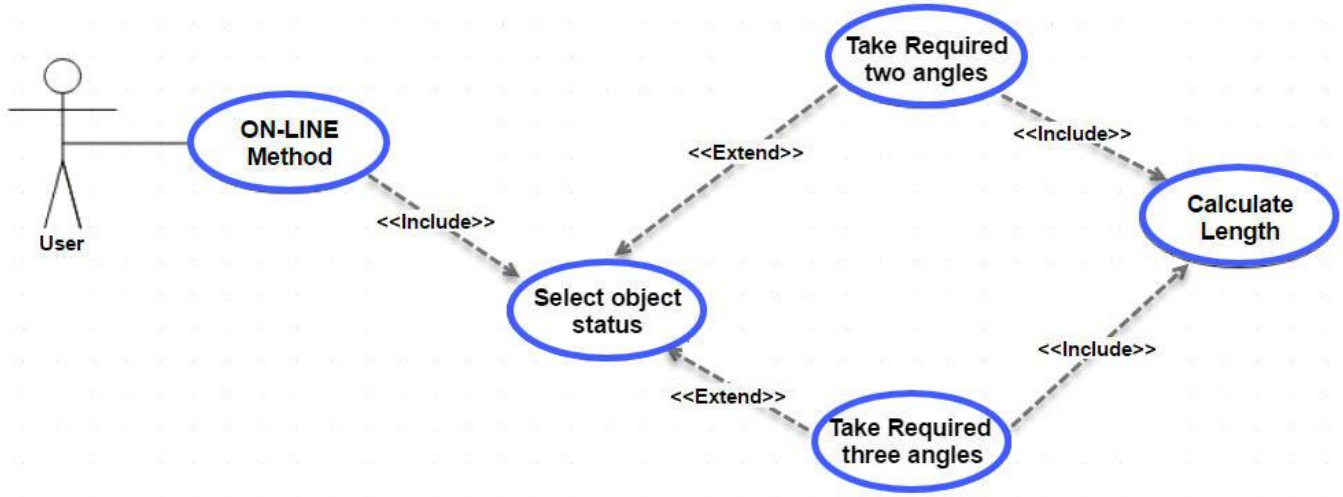


FIGURE 6 : DESCRIBING HOW ON-LINE METHOD WORK

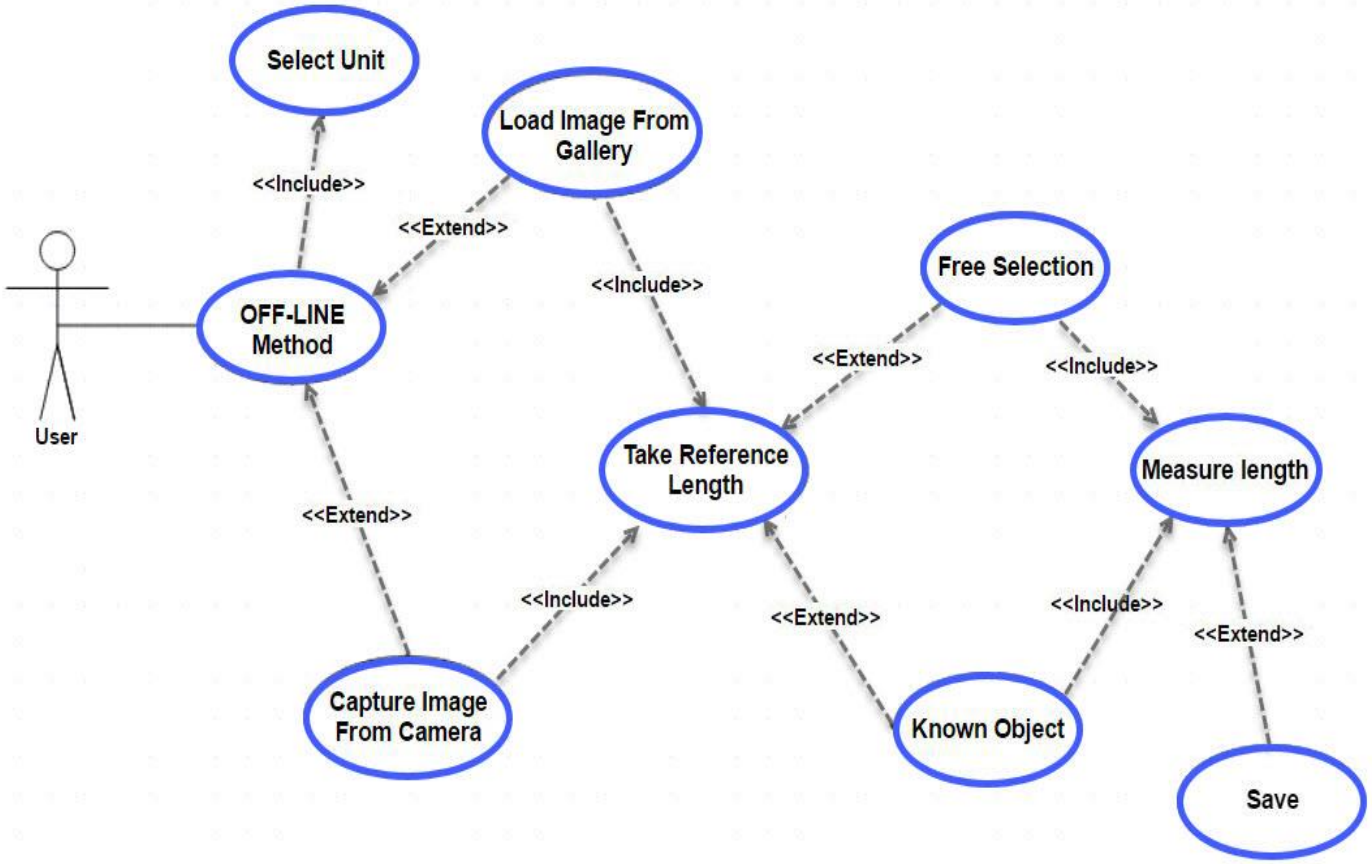


FIGURE 7 : DESCRIBING HOW OFF-LINE METHOD WORK

3.2.2 Class Diagram

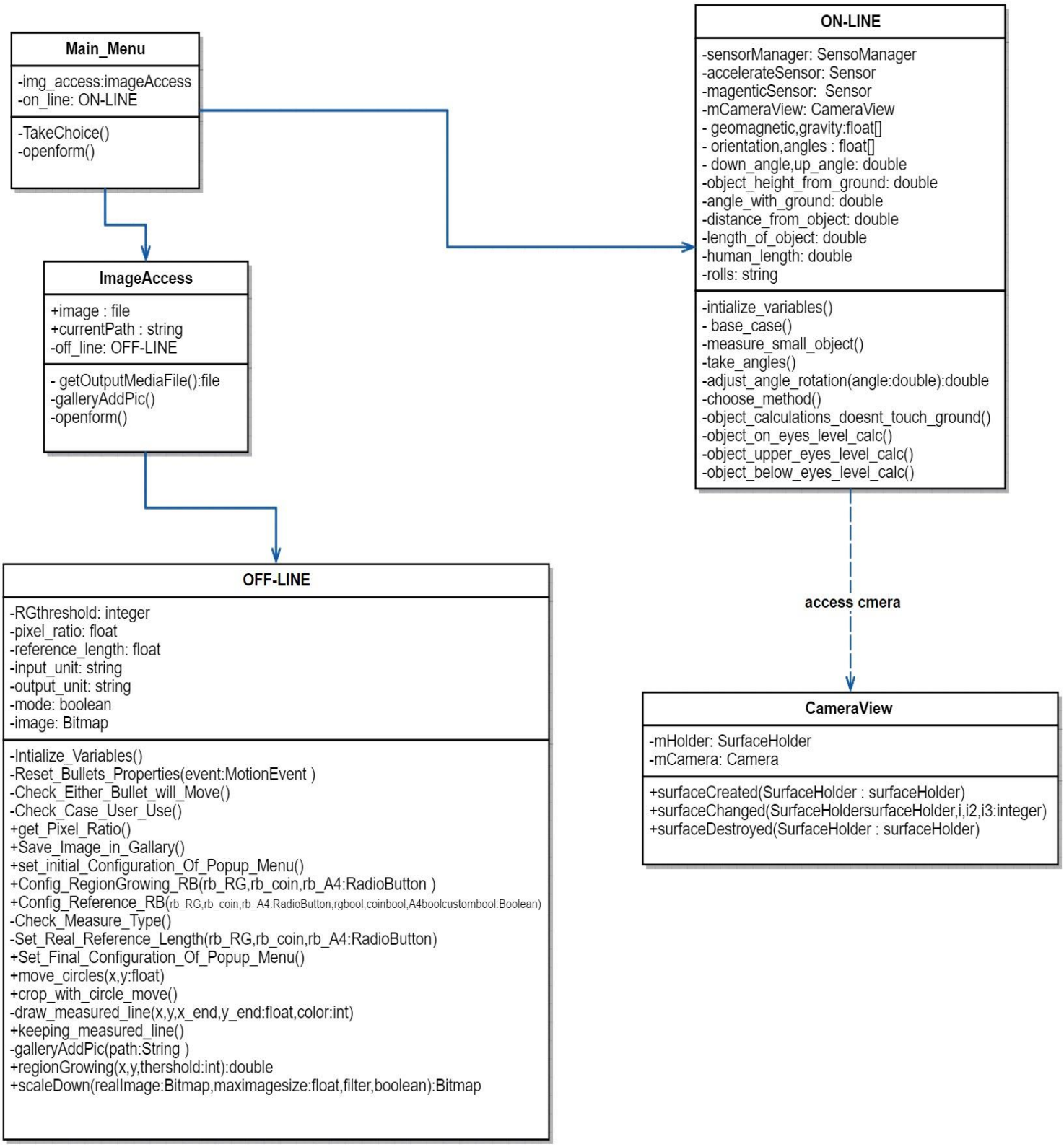


FIGURE 8 : APPLICATION CLASS DIAGRAM

3.2.3 Sequence Diagram

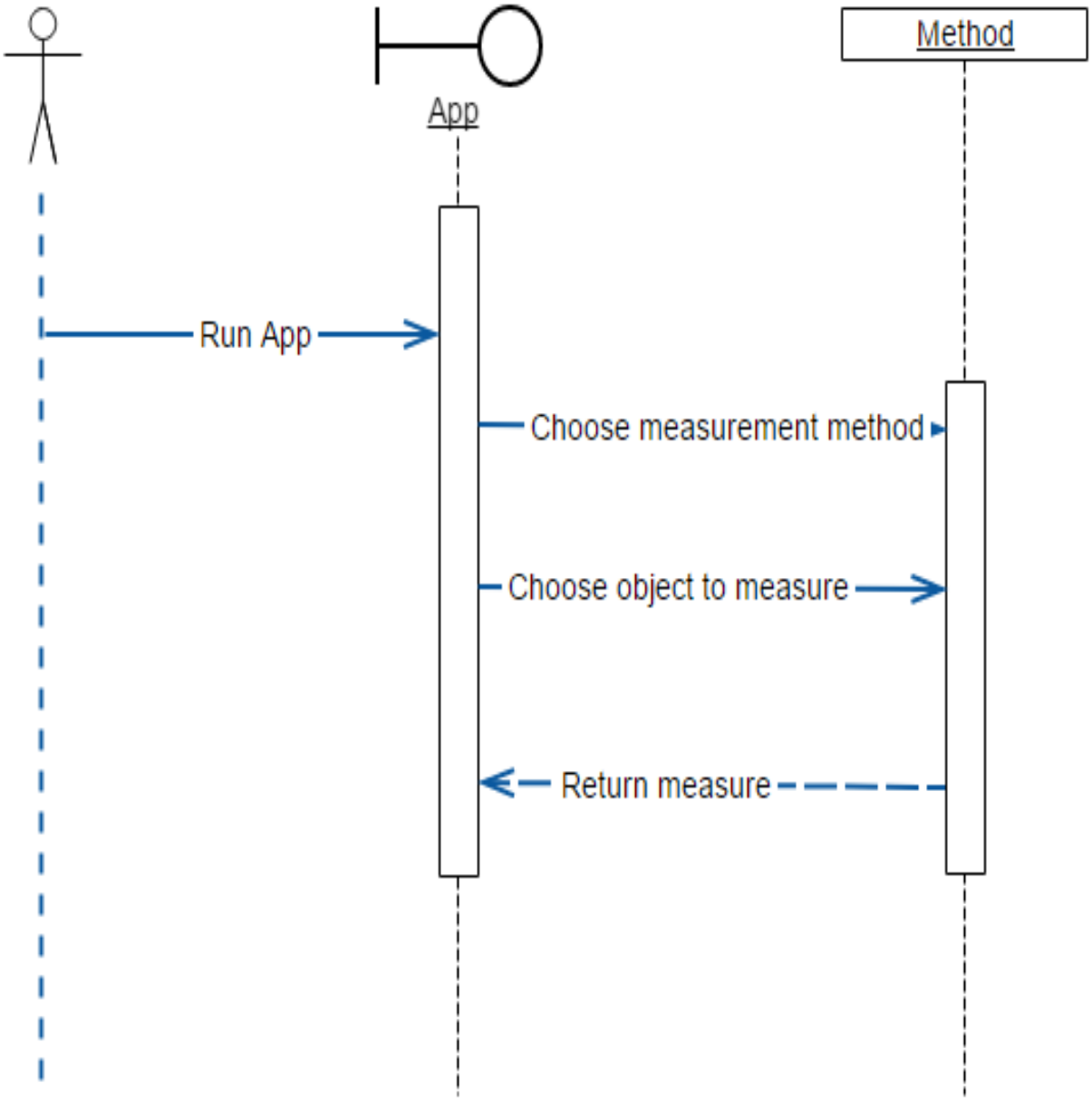


FIGURE 9 APPLICATION SEQUENCE DIAGRAM

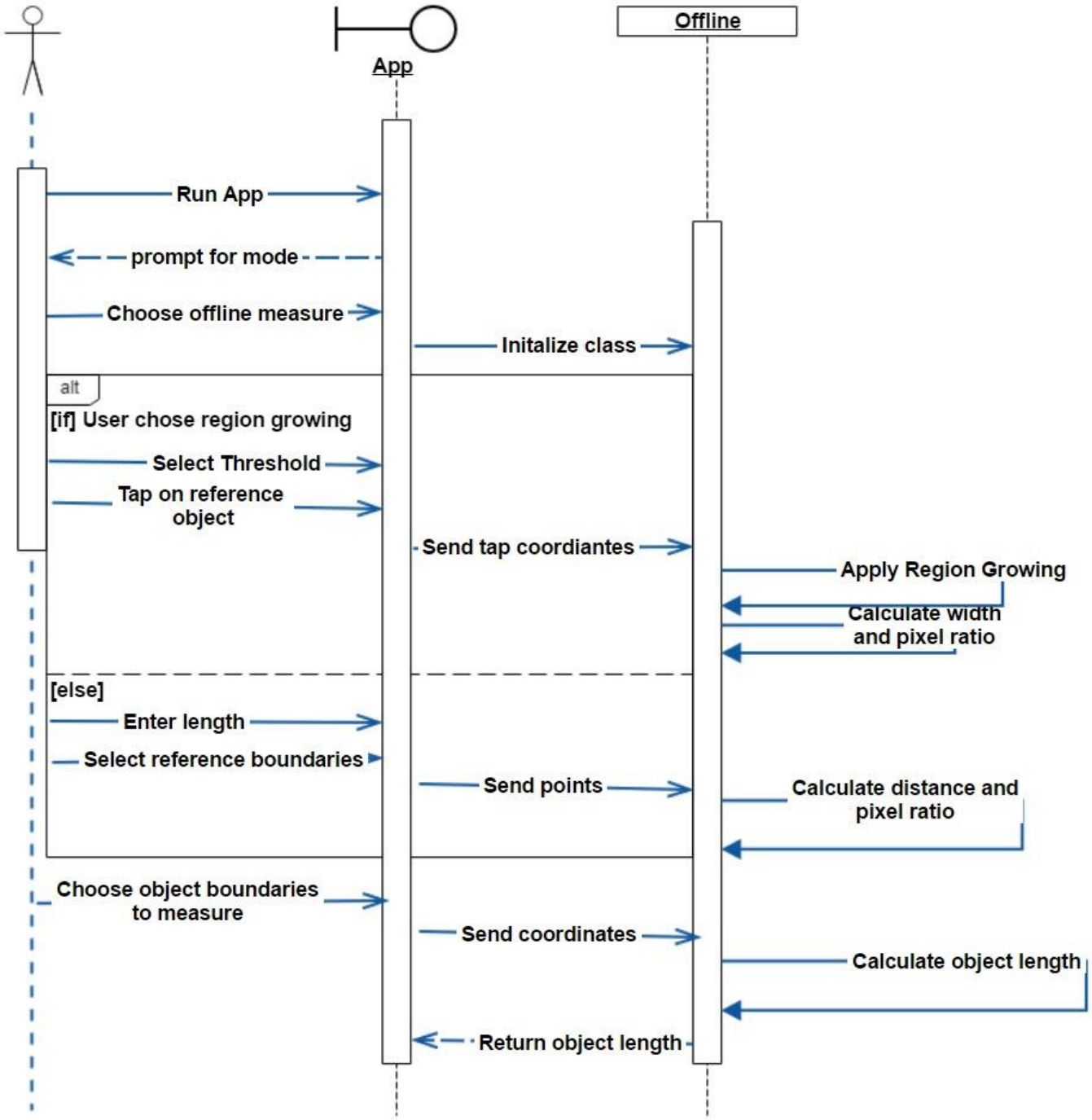


FIGURE 10 : OFF-LINE MODE SEQUENCE DIAGRAM

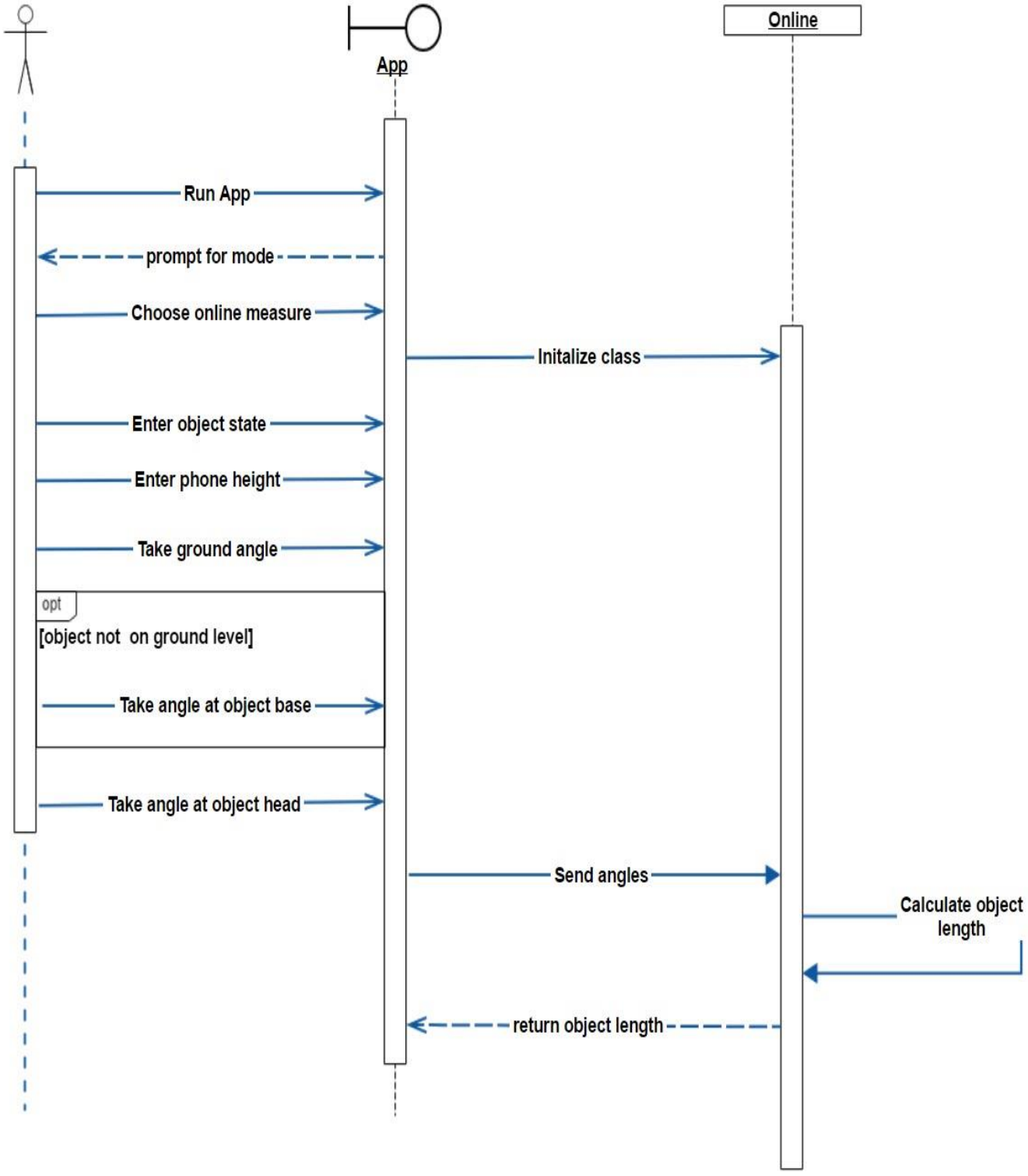


FIGURE 11 :ON-LINE MODE SEQUENCE DIAGRAM

4- Implementation and Testing

OFFLINE Mode Functionalities

1- Crop_with_circle_move Functionality

Used to crop the region around the point pressed on the screen and show it in the zooming scene imageview so that the user can see the object boundaries beneath his fingers.

```
    /*summary
    I-crop area around touched point.
end summary*/void crop_with_circle_move() {
        int x_s = (int) x - (crop_img_w / 2);
        int y_s = (int) y - (crop_img_h / 2);
        int x_e = crop_img_w, y_e = crop_img_h;
        if (x_s < 0) {
            x_e = crop_img_w + x_s;
            x_s = 0;
        }
        if (y_s < 0) {
            y_e = crop_img_h + y_s;
            y_s = 0;
        }
        if (x_s + crop_img_w > touch_original.getWidth()) {
            x_e = touch_original.getWidth() - x_s;
        }
        if (y_s + crop_img_h > touch_original.getHeight()) {
            y_e = touch_original.getHeight() - y_s;
        }

        touch_tmp.setDrawingCacheEnabled(true);
        touch_tmp.buildDrawingCache(true);
        Bitmap bitmap = touch_tmp.getDrawingCache();

        Bitmap tmp = Bitmap.createBitmap(bitmap, x_s, y_s, x_e, y_e);
        cropimage.setImageBitmap(tmp);

    }
```

2- **scaleDown** Functionality

Used to resize the image's bitmap with respect to the image width and height.

```
    /*summary
    I-scale the bitmap down.
end summary*/static Bitmap scaleDown(Bitmap realImage, float maxImageSize, boolean filter) {
        float ratio = Math.min(
                (float) maxImageSize / realImage.getWidth(),
                (float) maxImageSize / realImage.getHeight());
        int width = Math.round((float) ratio * realImage.getWidth());
        int height = Math.round((float) ratio * realImage.getHeight());

        Bitmap newBitmap = Bitmap.createScaledBitmap(realImage, width,
                height, filter);
        return newBitmap;
    }
```

3- **Check_Case_User_use** Functionality

It checks the case (custom, region growing) that the user will use. Then calls Get The_Pixel_Ratio function to get the pixel ratio and calls the Keeping_measured_line function to update the image with the line that carries the length.

```
/*summary
I-check the case (custom,region growing) that user will use it.
II-get pixel ratio.
III-update image with measured lines.
end summary*/void Check_Case_User_Use() {
    //custom case.
    if (!mode) {
        float CoorRefDistance = (float) Math.sqrt(Math.pow(circle_1.getX() - circle_2.getX(), 2)
                + Math.pow(circle_1.getY() - circle_2.getY(), 2));
        get_Pixel_Ratio(CoorRefDistance);
    }
    //region growing case.
    else if (mode) {
        bmp = Bitmap.createBitmap(touch_original.getWidth(), touch_original.getHeight(), Bitmap.Config.ARGB_8888);
        c = new Canvas(bmp);
        touch_tmp.draw(c);
        float CoorRefDistance = (float) regionGrowing((int) x, (int) y, threshold);
        get_Pixel_Ratio(CoorRefDistance);
        touch_original.setImageBitmap(bmp);
        circle_2.setVisibility(View.VISIBLE);
        mode = false;
    }
    keeping_measured_line();
    btnref.setText("Keep This Measured Line");
    refrence = true;
}
```

4- **Get_The_Pixel_Ratio** Functionality

It calculates the pixel ratio between the real world reference length and the in-image pixel length.

```
/*summary
I-calculate pixel ratio about real reference length and coordination reference length.
 end summary*/void get_Pixel_Ratio(float CoorRefDistance) {
    pixel_ratio = ref_len / CoorRefDistance;
}
```

5- **Keeping _measured_line** Functionality

it updates the bitmap with the measured line, that carries the length.

```
    /*summary
    I-save measured line.
end summary*/void keeping_measured_line() {
        circle_1.setX(touch_original.getWidth() / 2);
        circle_1.setY(0);
        circle_2.setX(touch_original.getWidth() / 2);
        circle_2.setY(touch_original.getHeight() / 2);
        touch_tmp.setImageBitmap(bmp);
    }
```

6- **CrearPunto** Functionality

It is responsible for drawing the line that carries the length about using the Canvas, it uses the positions of the touch at the borders of the object and drawing the line bounded by two small circles.

```
    /*summary
    I-dary measured line.
end summary*/void crearPunto(float x, float y, float xend, float yend, int color) {
        bmp = Bitmap.createBitmap(touch_original.getWidth(), touch_original.getHeight(), Bitmap.Config.ARGB_8888);
        c = new Canvas(bmp);
        touch_tmp.draw(c);
        Paint p = new Paint();
        Paint p1 = new Paint();
        p1.setStrokeWidth(5);
        p1.setColor(Color.BLUE);
        p.setStrokeWidth(5);
        p.setColor(color);
        c.drawCircle(x, y, 10, p1);
        c.drawCircle(xend, yend, 10, p1);
        c.drawLine(x, y, xend, yend, p);
        if (refrence) {
            float actual_dist = (float) Math.sqrt(Math.pow(circle_1.getX() - circle_2.getX(), 2)
                    + Math.pow(circle_1.getY() - circle_2.getY(), 2));
            actual_dist *= pixel_ratio;
            if (input_unit == "cm" && out_unit == "in")
                actual_dist *= 0.393701;
            else if (input_unit == "in" && out_unit == "cm")
                actual_dist /= 0.393701;
            Paint text_paint = new Paint();
            text_paint.setColor(Color.BLUE);
            text_paint.setStyle(Paint.Style.FILL_AND_STROKE);
            text_paint.setTextSize(30);
            text_paint.setStrokeWidth(3);
            float text_x = Math.abs((circle_1.getX() - circle_2.getX()) / 2) + Math.min(circle_1.getX(), circle_2.getX());
            float text_y = Math.abs((circle_1.getY() - circle_2.getY()) / 2) + Math.min(circle_1.getY(), circle_2.getY());
            c.drawText(String.valueOf(String.format("%.2f", actual_dist) + " " + out_unit), text_x, text_y, text_paint);
        }touch_original.setImageBitmap(bmp);
    }
```

7- Set_Initial_Configuration_of_Popup_Menu Functionality

used to set the configurations of the Popup Menu Modes (Custom, RegionGrowing).

```
/*summary
I-adjust the view of popup menu modes.
end summary*/void Set_initial_Configuration_Of_Popup_Menu() {
    mBuilder = new AlertDialog.Builder(MainActivity.this);
    mView = getLayoutInflater().inflate(R.layout.mode_dialog, null);

    rb_cutom = (RadioButton) mView.findViewById(R.id.rb_custom);
    spinner = (Spinner) mView.findViewById(R.id.spinner);
    editText = (EditText) mView.findViewById(R.id.editText);

}
```

8- Set_Configuration_Of_Custom_Radio_Button Functionality

Used to Set the configurations of the custom Radio Button Option.

```
/*summary
I-adjust the view of custom radio button.
end summary*/void Set_Configuration_Of_Custom_Radio_Button(RadioButton rb_region_growing, RadioButton rb_coin, RadioButton rb_a4) {
    rb_region_growing.setChecked(false);
    rb_coin.setChecked(false);
    rb_a4.setChecked(false);
    rb_cutom.setChecked(true);

    rb_coin.setVisibility(View.INVISIBLE);
    rb_a4.setVisibility(View.INVISIBLE);
    spinner.setVisibility(View.VISIBLE);
    editText.setVisibility(View.VISIBLE);
    rb_cutom.setY(rb_region_growing.getBottom());

}
```

9- Set_Real_Reference_Length Functionality

Used to set the reference real length.

```
/*summary
I-set the real reference value.
end summary*/void Set_Real_Reference_Length(RadioButton rb_coin, RadioButton rb_a4) {
    if (rb_coin.isChecked())
        ref_len = 2.2f;//length
    if (rb_a4.isChecked())
        ref_len = 36.37f;//diameter

}
```

10- Set_Configuration_Of_RegionGrowing_Radio_Button Functionality

Used to set the configurations of the Region Growing Radio Button that contains the two options (A4, Coin).

```
/*summary
I-adjust the view of RegionGrowing radio button.
end summary*/void Set_Configuration_Of_RegionGrowing_Radio_Button(RadioButton rb_region_growing, RadioButton rb_coin, RadioButton rb
    rb_region_growing.setChecked(true);
    rb_coin.setChecked(false);
    rb_a4.setChecked(false);
    rb_cutom.setChecked(false);

    rb_cutom.setVisibility(View.VISIBLE);
    rb_coin.setVisibility(View.VISIBLE);
    rb_a4.setVisibility(View.VISIBLE);
    spinner.setVisibility(View.INVISIBLE);
    editText.setVisibility(View.INVISIBLE);
    rb_cutom.setY(spinner.getY());
}
```

11- Set_Reference_Configurations_Of_Radio_Button

Functionality

Used to set the configurations of the radio buttons inside the spinner (Modes) according to user's selection.

```
/*summary
I-update configurations of radio button according to reference's type.
end summary*/void Set_Reference_Configurations_Of_Radio_Button(RadioButton rb_region_growing, RadioButton rb_coin, RadioButton rb_a4
    rb_region_growing.setChecked(rgbool);
    rb_coin.setChecked(coinbool);
    rb_a4.setChecked(a4bool);
    rb_cutom.setChecked(custombool);
}
```

12- Set_Final_Configuration_Of_Popup_Menu Function

Used to finalize the configurations of the Pop Up Menu(Modes) before displaying it.

```
/*summary
I-show the dialog.
end summary*/void Set_Final_Configuration_Of_Popup_Menu() {
    mBuilder.setView(mView);
    AlertDialog alertDialog = mBuilder.create();
    alertDialog.setTitle("Modes");
    alertDialog.show();
    btn_mode.setVisibility(View.INVISIBLE);
}
```

13-**Move_Circles** Functionality

Used to check if the position of the touch on the screen belongs to any circle.

```
    /*summary
    I-choose which circle will be move.
end summary*/int move_circles(float x_t, float y_t) {


        small_local_pos_x = (int) circle_1.getX();
        small_local_pos_y = (int) circle_1.getY();
        if (x_t > small_local_pos_x && x_t < (small_local_pos_x + circles_w))
            if (y_t > small_local_pos_y && y_t < (small_local_pos_y + circles_h))
                return 1;
        small_local_pos_x = (int) circle_2.getX();
        small_local_pos_y = (int) circle_2.getY();
        if (x_t > small_local_pos_x && x_t < (small_local_pos_x + circles_w))
            if (y_t > small_local_pos_y && y_t < (small_local_pos_y + circles_h))
                return 2;
        return 0;
    }
```

14- **Check_Either_Bullet_will_Move** Functionality

-Calls Move_Circles function to check the touch x,y positions within the region of any circle to move then updates the zooming scene around the area the user taps.

-Calls CrearPunto function to draw the measured line.

```
/*summary
I-Check the touch X,Y within the region of any circle to move.
II-Update zooming scene around the area that user touched on it.
end summary*/void Check_Either_Bullet_will_Move() {
    if (move_circles(x, y) == 1 || hold == true) {
        hold = true;
        circle_1.setX(x - (circles_w / 2));
        circle_1.setY(y - (circles_h / 2));
        circle_1.setAlpha(0.8f);
        circle_2.setAlpha(0.3f);
        crop_with_circle_move();

        if (!mode)
            crearPunto((circle_1.getX() + (circles_w / 2)), (circle_1.getY() + (circles_h / 2)),
                    (circle_2.getX() + (circles_w / 2)), (circle_2.getY() + (circles_h / 2)), Color.YELLOW);
    } else if (move_circles(x, y) == 2 || hold2 == true) {
        hold2 = true;
        circle_2.setX(x - (circles_w / 2));
        circle_2.setY(y - (circles_h / 2));
        circle_1.setAlpha(0.3f);
        circle_2.setAlpha(0.8f);
        crop_with_circle_move();

        crearPunto((circle_1.getX() + (circles_w / 2)), (circle_1.getY() + (circles_h / 2))
                , (circle_2.getX() + (circles_w / 2))
                , (circle_2.getY() + (circles_h / 2)), Color.YELLOW);
    }
}
```

15- **Update_Bullets_Dimensions** Functionality

Get the positions x, y of the circle from the screen coordinates. Then get the width and height of the circle and the cropped scene.

```
/*summary
I-get the positions X,Y of the circle from screen coordinates.
II-get width and height of circle and crop scene.
end summary*/void Update_Bullets_Dimensions(MotionEvent event) {
    x = event.getX();
    y = event.getY();
    circles_w = circle_1.getWidth();
    circles_h = circle_1.getHeight();
    crop_img_w = cropimage.getWidth();
    crop_img_h = cropimage.getHeight();
}
```

16- **region Growing** Functionality

Get the Dimensions of the reference object by examining neighboring pixels of initial seed points and determines whether the pixel neighbors should be added to the region according to a threshold specified by user.

```
    /*summary
    I-get the dimensions of object automatically using region growing
end summary*/double regionGrowing(int x, int y, int Threshold) {
        TreeSet<Integer> open = new TreeSet<~>(), closed = new TreeSet<~>();
        int count = 0;
        //scale image and x,y
        Bitmap image = scaleDown(bmp, 800, true);
        x = (int) (x * (float) image.getWidth() / (float) bmp.getWidth());
        y = (int) (y * (float) image.getHeight() / (float) bmp.getHeight());
        //get Seed Node RGB values
        int color = image.getPixel(x, y);
        int A = Color.alpha(color), R = Color.red(color), G = Color.green(color), B = Color.blue(color);
        int Xmin = x, Xmax = x, Ymin = y, Ymax = y, Ymin2 = y, Ymax2 = y;
        int seedNode = x * image.getWidth() + y;
        open.add(seedNode);
        while (!open.isEmpty()) {
            // remove current node from open list and add to closed list
            int currentNode = open.first();
            closed.add(currentNode);
            open.remove(currentNode);
            //get current node coordinates
            int xCurr = currentNode / image.getWidth(), yCurr = currentNode % image.getWidth();
            int upColor = 0, downColor = 0, leftColor = 0, rightColor = 0;
            if (xCurr - 1 > 0) upColor = image.getPixel(xCurr - 1, yCurr);
            if (xCurr + 1 < image.getWidth()) downColor = image.getPixel(xCurr + 1, yCurr);
            if (yCurr - 1 > 0) leftColor = image.getPixel(xCurr, yCurr - 1);
            if (yCurr + 1 < image.getHeight()) rightColor = image.getPixel(xCurr, yCurr + 1);
            //Check upper Pixel
            if (xCurr - 1 > 0 && Math.abs(R - Color.red((upColor))) <= Threshold
                    && Math.abs(B - Color.blue((upColor))) <= Threshold && Math.abs(G - Color.green((upColor))) <= Threshold
                    && Math.abs(A - Color.alpha(upColor)) <= Threshold &&
                    !closed.contains(currentNode - image.getWidth()) && !open.contains(currentNode - image.getWidth()))
                open.add((xCurr - 1) * image.getWidth() + yCurr);
            //Check lower Pixel
            if (xCurr + 1 < image.getWidth() && Math.abs(R - Color.red((downColor))) <= Threshold
                    && Math.abs(B - Color.blue((downColor))) <= Threshold
                    && Math.abs(G - Color.green((downColor))) <= Threshold
                    && Math.abs(A - Color.alpha(downColor)) <= Threshold
                    && !closed.contains(currentNode + image.getWidth()) && !open.contains(currentNode + image.getWidth()))
                open.add((xCurr + 1) * image.getWidth() + yCurr);
```

```
    //Check left Pixel
    if (yCurr - 1 > 0 && Math.abs(R - Color.red((leftColor))) <= Threshold
            && Math.abs(B - Color.blue((leftColor))) <= Threshold && Math.abs(G - Color.green((leftColor))) <= Threshold
            && Math.abs(A - Color.alpha(leftColor)) <= Threshold && !closed.contains(currentNode - 1)
            && !open.contains(currentNode - 1))
        open.add((xCurr) * image.getWidth() + yCurr - 1);
    //Check right Pixel
    if (yCurr + 1 < image.getHeight() && Math.abs(R - Color.red((rightColor))) <= Threshold
            && Math.abs(B - Color.blue((rightColor))) <= Threshold
            && Math.abs(G - Color.green((rightColor))) <= Threshold
            && Math.abs(A - Color.alpha(rightColor)) <= Threshold && !closed.contains(currentNode + 1)
            && !open.contains(currentNode + 1))
        open.add((xCurr) * image.getWidth() + yCurr + 1);

    image.setPixel(xCurr, yCurr, Color.RED);
    //check for min & max x,y
    if (xCurr < Xmin) {
        Xmin = xCurr;
        Ymin = Ymin2 = yCurr;
    } else if (xCurr == Xmin) {
        Ymin = Math.min(Ymin, yCurr);
        Ymin2 = Math.max(Ymin2, yCurr);
    }
    if (xCurr > Xmax) {
        Xmax = xCurr;
        Ymax = Ymax2 = yCurr;
    } else if (xCurr == Xmax) {
        Ymax = Math.max(Ymax, yCurr);
        Ymax2 = Math.min(Ymax2, yCurr);
    }
}
image.setPixel(Xmin, Ymin, Color.BLUE);
image.setPixel(Xmax, Ymax, Color.BLUE);
image.setPixel(Xmin, Ymin2, Color.GREEN);
image.setPixel(Xmax, Ymax2, Color.GREEN);
Xmax = (int) (Xmax * ((float) bmp.getWidth() / (float) image.getWidth()));
Xmin = (int) (Xmin * ((float) bmp.getWidth() / (float) image.getWidth()));
Ymax = (int) (Ymax * ((float) bmp.getHeight() / (float) image.getHeight()));
Ymin = (int) (Ymin * ((float) bmp.getHeight() / (float) image.getHeight()));
Ymax2 = (int) (Ymax2 * ((float) bmp.getHeight() / (float) image.getHeight()));
Ymin2 = (int) (Ymin2 * ((float) bmp.getHeight() / (float) image.getHeight()));
double dist1 = Math.sqrt(Math.pow(Xmax - Xmin, 2) + Math.pow(Ymax - Ymin, 2));
double dist2 = Math.sqrt(Math.pow(Xmax - Xmin, 2) + Math.pow(Ymax2 - Ymin2, 2));
bmp = scaleDown(image, Math.max(bmp.getWidth(), bmp.getHeight()), true);
return Math.max(dist1, dist2);
```

17- **Reset_Bullets_Properties** Functionality

Reset the Bullets Properties (color, hold).

```
/*summary
I-reset configurations's bullets.
end summary*/void Reset_Bullets_Properties_When_Action_Up(MotionEvent event) {
    if (event.getAction() == android.view.MotionEvent.ACTION_UP) {
        circle_1.setAlpha(0.3f);
        circle_2.setAlpha(0.3f);
        hold = false;
        hold2 = false;
    }
}
```

18-**Check_Measure_Type** Functionality

Check the measure type(CM ,INCH).

```
/*summary
I-check measure type(cm,inch).
end summary*/void Check_Measure_Type() {
    if (spinner.getSelectedItem().toString().equalsIgnoreCase("CM"))
        input_unit = "cm";
    else
        input_unit = "in";
}
```

19-**galleryAddPic** Functionality Add the image to gallery path.

```
    /*summary
     I-add the photo to gallary.
end summary*/void galleryAddPic(String path) {
        Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
        File f = new File(path);
        Uri contentUri = Uri.fromFile(f);
        mediaScanIntent.setData(contentUri);
        this.sendBroadcast(mediaScanIntent);
    }
```

20-**Save_Image_in_Gallary** Functionality

Save the image to the gallery.

```
/*summary
I-save image in gallary.
end summary*/void Save_Image_in_Gallary() {
    OutputStream outStream = null;
    File file = new File(recived_image_path);
    File image_path = new File(Environment.getExternalStoragePublicDirectory(""), "Camera Measure/" + file.getName());
    try {
        outStream = new FileOutputStream(image_path);
        bmp.compress(Bitmap.CompressFormat.JPEG, 100, outStream);
        outStream.flush();
        outStream.close();
        galleryAddPic(image_path.getAbsolutePath());

    } catch (Exception e) {
    }
    Toast.makeText(MainActivity.this, "Image Saved", Toast.LENGTH_SHORT).show();
}
```

ONLINE Mode:

1-adjust_angle_rotation Functionality

Adjust angle rotation (with respect to the camera direction).

```
        /*summary
I-adjust angle rotation.
 end summary*/double adjust_angle_rotation(double angle) {
        double temp;
        temp=angle;
           if(temp>90) {
               temp=180-temp;
           }
        return temp;
    }
```

2-**object_on_eyes_level_calc** Functionality

Calculates the length of the object which its position is at eye's level by using the down, up angle, angle with ground, user's Height and the distance between the user and the target object.

```
        /*summary
 I-measure object started above ground and on eye's level.
 } end summary*/void object_on_eyes_level_calc(){
        down_angle=Math.abs(down_angle);
        up_angle=Math.abs(up_angle);
        angle_with_ground=90-angle_with_ground;
        distance_from_object=human_length*Math.tan(Math.toRadians(angle_with_ground));
        double part_down=distance_from_object*Math.tan(Math.toRadians(down_angle));
        double part_up=distance_from_object*Math.tan(Math.toRadians(up_angle));
        length_of_object=part_down+part_up;
        object_height_from_ground=human_length-part_down;
        ORI.setText("length_of_object :\n" +String.valueOf(String.format("%.2f",(length_of_object/100)))
                +" M"+ "\n" + "distance_from_object :\n" + String.valueOf(String.format("%.2f",(distance_from_object/100)))+
                " M" + "\n"+"height_from_ground :\n" + String.valueOf(String.format("%.2f",(object_height_from_ground/100)))+" M");


    }
```

3-**object_upper_eyes_level_calc** Functionality

Calculate the length of the object which its position is above eye's level about using the down, up angle, angle with ground, user's Height and the distance between the user and the target object.

```
    /*summary
I-measure object started above ground and upper than eye's level.
end summary*/void  object_upper_eyes_level_calc(){
        down_angle=Math.abs(down_angle);
        up_angle=Math.abs(up_angle);

        angle_with_ground=90-angle_with_ground;
            distance_from_object=human_length*Math.tan(Math.toRadians(angle_with_ground));
            double part=distance_from_object*Math.tan(Math.toRadians(down_angle));
            double all=distance_from_object*Math.tan(Math.toRadians(up_angle));
            length_of_object=all-part;
            object_height_from_ground=human_length+part;
        ORI.setText("length_of_object :\n" +String.valueOf(String.format("%.2f",(length_of_object/100))) +
                " M"+ "\n" + "distance_from_object :\n" + String.valueOf(String.format("%.2f",(distance_from_object/100)))+
                " M" + "\n"+"height_from_ground :\n" + String.valueOf(String.format("%.2f",(object_height_from_ground/100)))+" M");

    }
```

4-**object_below_eyes_level_calc** Functionality

Calculate the length of the object which its position is below eye's level about using the down, up Angle, Angle with ground, user's Height and the distance between the user and the target object.

```
    /*summary
I-measure object started above ground and shorter than eye's level.
end summary*/void  object_below_eyes_level_calc(){
        down_angle=Math.abs(down_angle);
        up_angle=Math.abs(up_angle);
        angle_with_ground=90-angle_with_ground;
            distance_from_object=human_length*Math.tan(Math.toRadians(angle_with_ground));
            double all=distance_from_object*Math.tan(Math.toRadians(down_angle));
            double part=distance_from_object*Math.tan(Math.toRadians(up_angle));
            length_of_object=all-part;
            object_height_from_ground=human_length-all;
        ORI.setText("length_of_object :\n" +String.valueOf(String.format("%.2f",(length_of_object/100))) +
                " M"+ "\n" + "distance_from_object :\n" + String.valueOf(String.format("%.2f",(distance_from_object/100)))+
                " M" + "\n"+"height_from_ground :\n" + String.valueOf(String.format("%.2f",(object_height_from_ground/100)))+" M");

    }
```

5-**object_calculations_doesnt_touch_ground** Functionality

Automatically determines the height level of the object with respect to the user's eye level by using object_on_eyes_level_calc, object_upper_eyes_level_calc, and object_below_eyes_level_calc Functions.

```
        /*summary
 I-choose automatically which method will be execute (above ground).
 end summary*/void object_calculations_doesnt_touch_ground(){
        if(angle_with_ground>0&&down_angle>0&&up_angle<0)
            object_on_eyes_level_calc();
        else if(angle_with_ground>0&&down_angle<0&&up_angle<0)
            object_upper_eyes_level_calc();
        else if(angle_with_ground>0&&down_angle>0&&up_angle>0)
            object_below_eyes_level_calc();
    }

        /*summary
```

6-**base_case** Functionality

Measures objects that starts from the ground and is taller than the user's height using the down, up Angle, user's Height and the distance between the user and the target object.

```
    /*summary
I-measure object started from ground and taller than human.
 end summary*/void base_case() {
        down_angle=Math.abs(down_angle);
        up_angle=Math.abs(up_angle);
        angles.setText("up angle="+up_angle+"\n"+"down angle="+down_angle+"roll"+"\n"+rolls);
            distance_from_object = human_length / Math.tan(Math.toRadians(down_angle));
            length_of_object = human_length + Math.tan(Math.toRadians(up_angle)) * distance_from_object;
        ORI.setText("length_of_object :\n" +String.valueOf(String.format("%.2f",(length_of_object/100)) +" M"+ "\n"
                + "distance_from_object :\n" + String.valueOf(String.format("%.2f",(distance_from_object/100)))+" M"));


    }
```

7-**Measure_Small_Object** Functionality

Measures objects that starts from ground and is smaller than the user's height using the down, up Angle, user's Height and the distance between the user and the target object.

```
    /*summary
I-measure object started from ground and shorter than human.
 end summary*/ void measure_small_object() {
        down_angle=Math.abs(down_angle);
        up_angle=Math.abs(up_angle);
        angles.setText("up angle="+up_angle+"\n"+"down angle="+down_angle+"\n"+"roll"+rolls);
        double distance_angle=90-down_angle;
         distance_from_object = human_length*Math.tan(Math.toRadians(distance_angle));
         double part_of_my_tall=distance_from_object*Math.tan(Math.toRadians(up_angle));
         length_of_object=human_length-part_of_my_tall;
        ORI.setText("length_of_object :\n" +String.valueOf(String.format("%.2f",(length_of_object/100)) +" M"+
                "\n" + "distance_from_object :\n" + String.valueOf(String.format("%.2f",(distance_from_object/100)))+" M"));


    }
```

8- **object_calculations_touches_ground** Functionality

Automatically determines the height level of the object (smaller than the user's height OR taller than the user's height).

```
        /*summary
 I-choose automatically which method will be execute (on ground).
 end summary*/void object_calculations_touch_ground(){
    if(down_angle<0&&up_angle>0)//base case
    {
        double temp=up_angle;
        up_angle=down_angle;
        down_angle=temp;
        base_case();
    }
    else if((down_angle>0&&up_angle>0)&&(down_angle<up_angle))//smaller object
    {
        double temp=up_angle;
        up_angle=down_angle;
        down_angle=temp;
        measure_small_object();
    }
        else if(up_angle<0&&down_angle>0)//base case
        base_case();
        else //smaller object
        measure_small_object();
}
```

9-**take_angles** Functionality

take the angles received from orientation matrix using adjust_angle_rotation function and calculate the object height using object_calculations_touches_ground function OR object_calculations_touches_ground function.

```
        /*summary
 I-take angles from sensors.
 end summary*/void take_angles() {

        rolls+=String.valueOf((Math.toDegrees(orientation[2]))%360+90)+"\n";
       if(!touch_ground_switch.isChecked()&&angle_with_ground==0)
           angle_with_ground=adjust_angle_rotation(Math.toDegrees(orientation[2]) % 360 + 90);
    else if (down_angle == 0)
        down_angle=adjust_angle_rotation(Math.toDegrees(orientation[2]) % 360 + 90);
    else if(up_angle==0)
    {
        up_angle=adjust_angle_rotation(Math.toDegrees(orientation[2]) % 360 + 90);
        touch_ground_switch.setVisibility(View.INVISIBLE);
        if(!touch_ground_switch.isChecked())
            object_calculations_doesnt_touch_ground();
        else
            object_calculations_touch_ground();
    }
}
```

Techniques and Algorithms

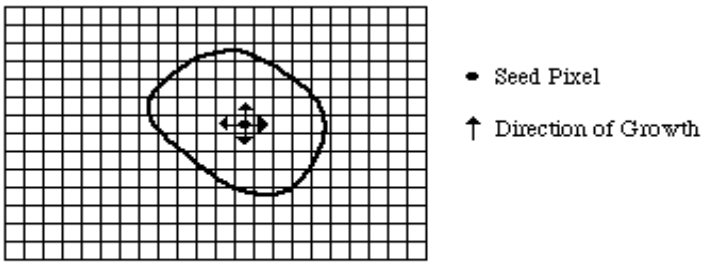
OFF-LINE Mode:

Region Growing

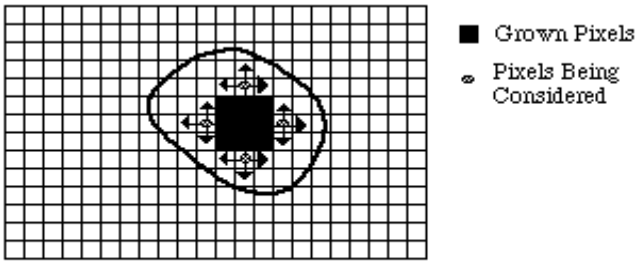
Region growing approach is the opposite of the split and merge approach:

- An initial set of small areas are iteratively merged according to similarity constraints.
- Start by choosing an arbitrary *seed pixel* and compare it with neighboring pixels.
- Region is grown from the seed pixel by adding in neighboring pixels that are similar, increasing the size of the region.
- When the growth of one region stops we simply choose another seed pixel which does not yet belong to any region and start again.
- This whole process is continued until all pixels belong to some region.

- Region growing methods often give very good segmentations that correspond well to the observed edges.



(a) Start of Growing a Region



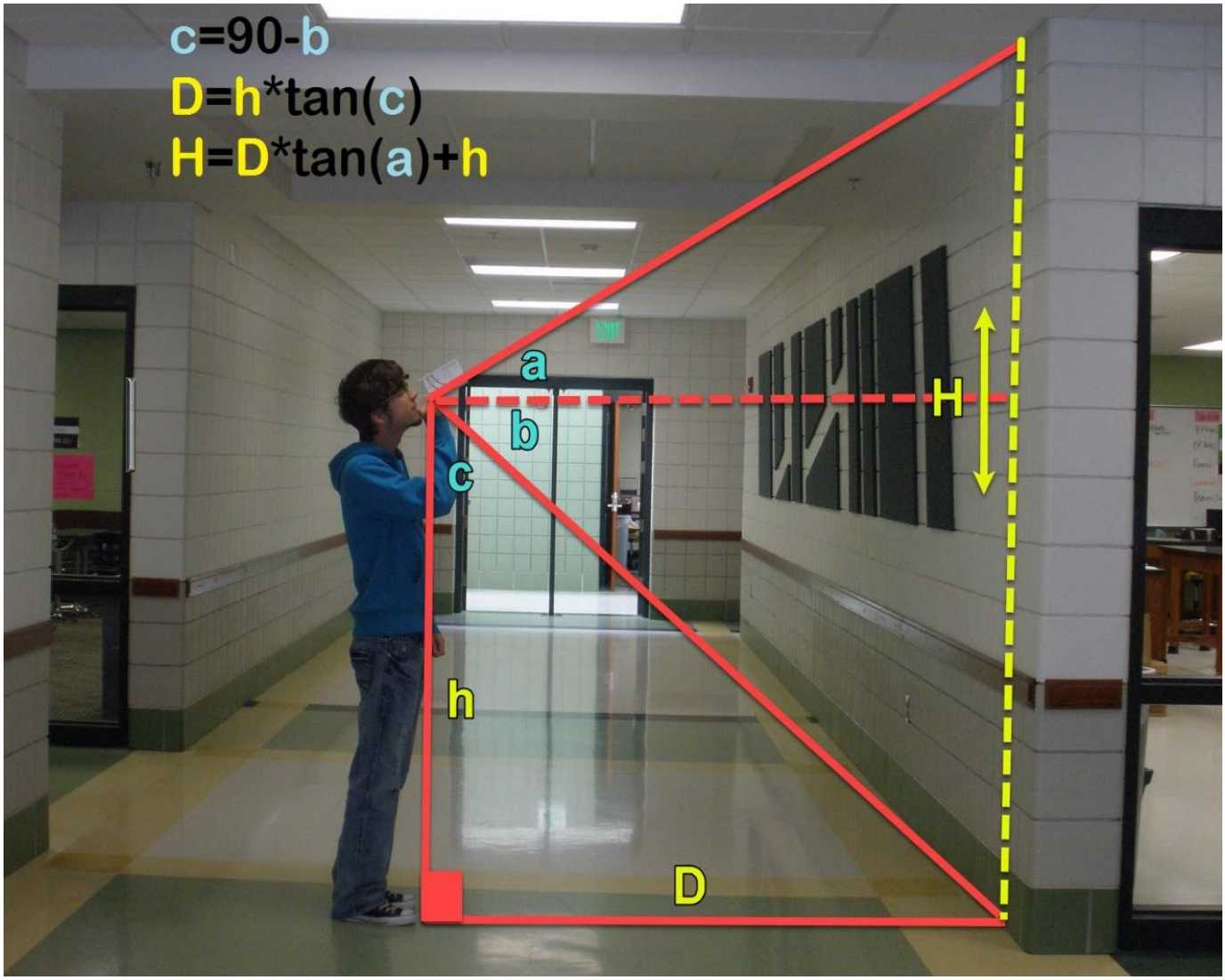
(b) Growing Process After a Few Iterations

ONLINE Mode:

There are techniques used depends on some trigonometrics.

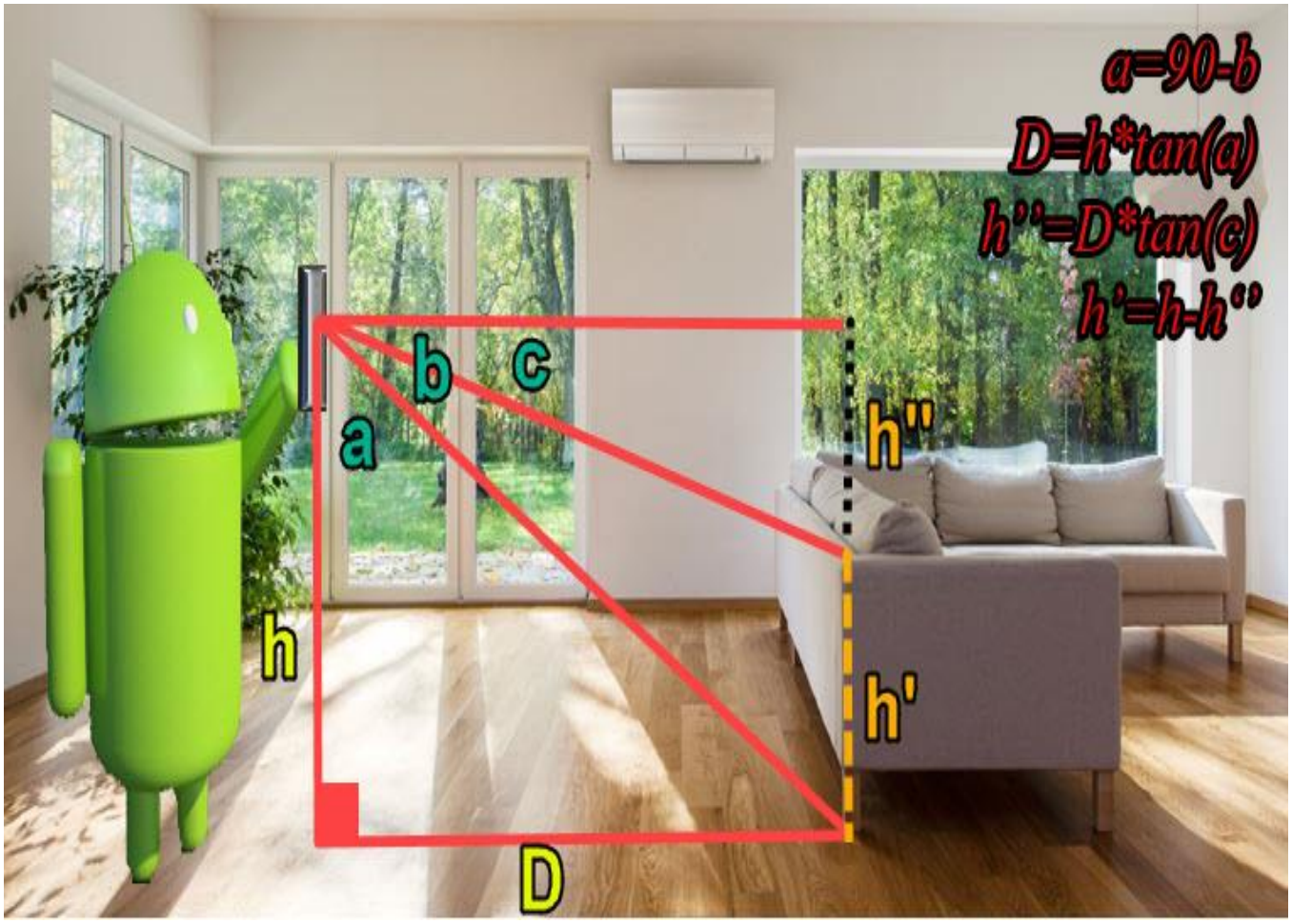
CASE1:

Related to the object that touches the ground and taller than the user's height.



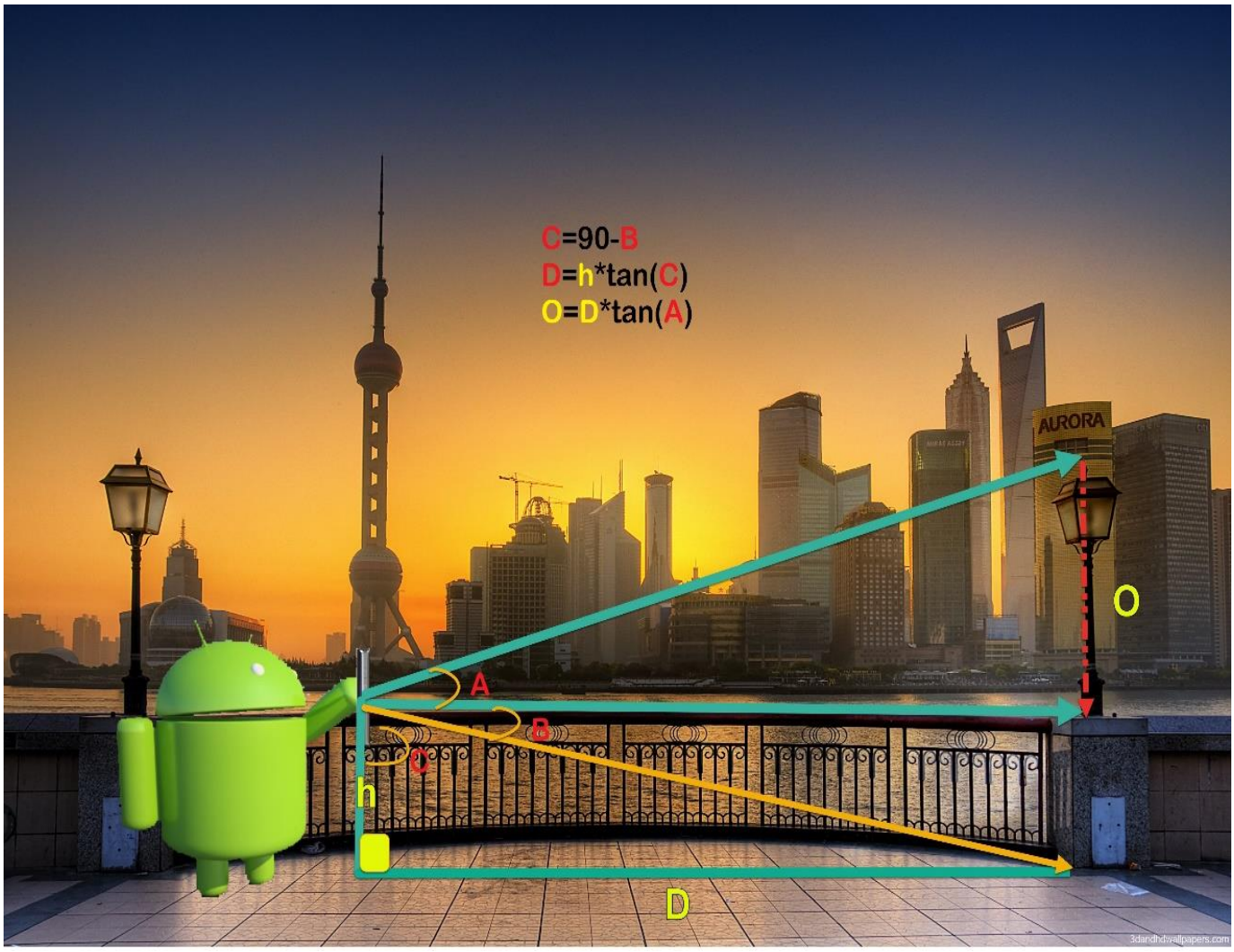
CASE2:

Related to the object that touches the ground and smaller than the user's height.



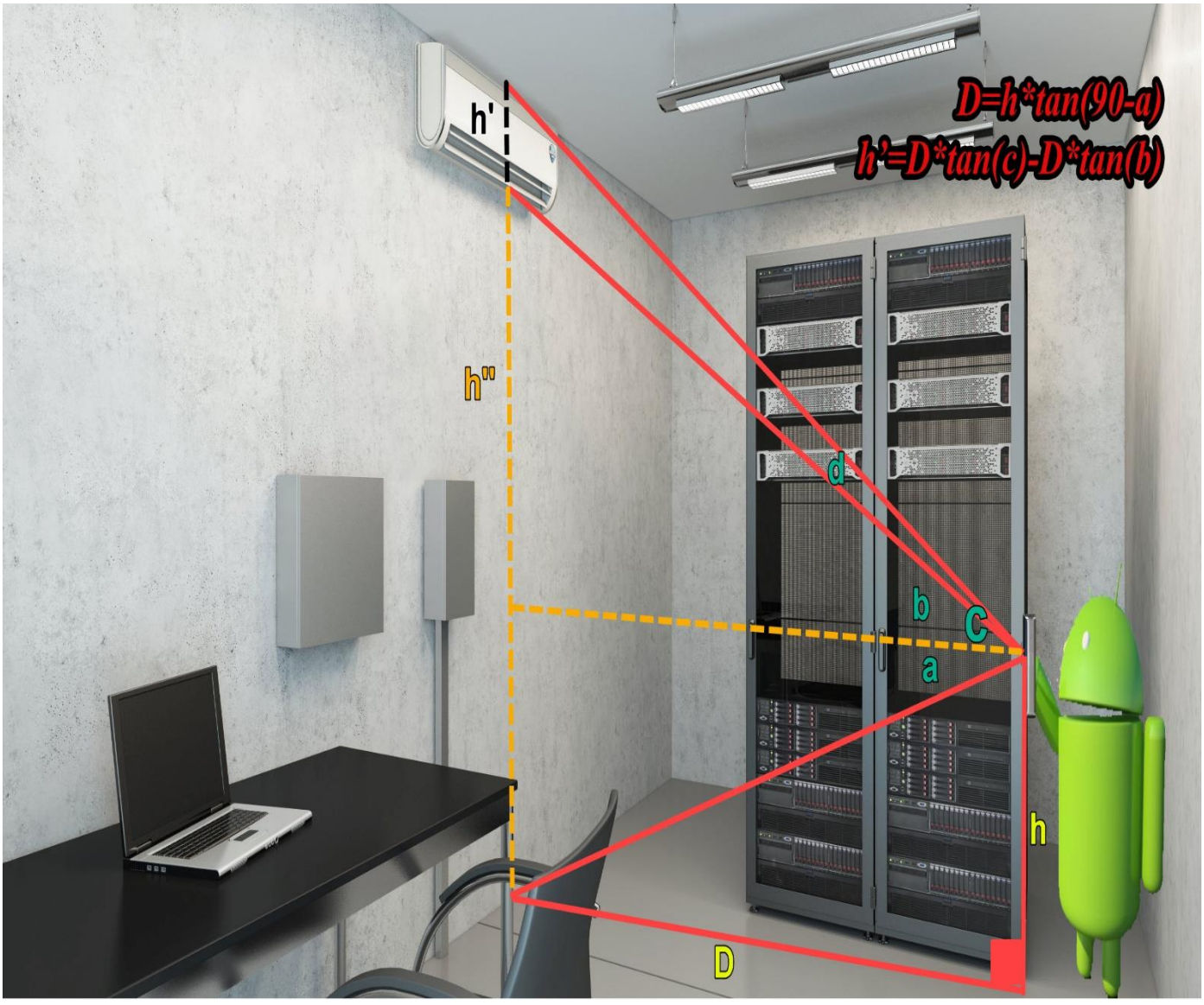
CASE3:

Related to the object that doesn't touches the ground and is at the eye's level.



CASE4:

Related to the object that doesn't the ground and is above the eye's level.



Accelerometer

Introduction:

One of the most common inertial sensors is the accelerometer, a dynamic sensor capable of a vast range of sensing. Accelerometers are available that can measure acceleration in one, two, or three orthogonal axes. They are typically used in one of three modes:

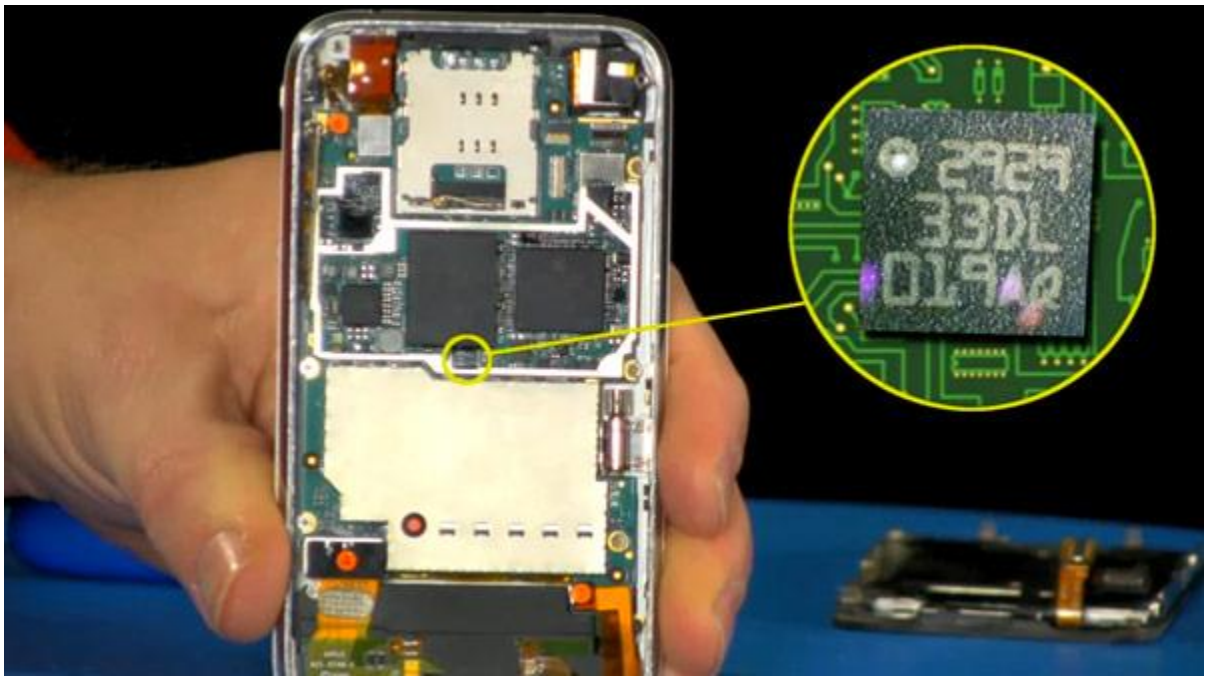
- As an inertial measurement of velocity and position;
- As a sensor of inclination, tilt, or orientation in 2 or 3 dimensions, as referenced from the acceleration of gravity

  (1 g = 9.8m/s2);
- As a vibration or impact (shock) sensor.

Accelerometers have multiple applications in industry and science. Highly sensitive accelerometers are components of inertial navigation systems for aircraft and missiles. Accelerometers are used to detect and monitor vibration in rotating machinery. Accelerometers are used in tablet computers and digital cameras so that images on screens are always displayed upright. Accelerometers are used in drones for flight stabilization. Coordinated accelerometers can be used to measure differences in proper acceleration, particularly gravity, over their separation in space; i.e., gradient of the gravitational field. This gravity gradiometry is useful because absolute gravity is a weak effect and depends on local density of the Earth which is quite variable.

Definition:

An accelerometer is a sensor which measures the tilting motion and orientation of a mobile phone.



Physical principles:

An accelerometer at rest relative to the Earth's surface will indicate approximately 1 g upwards, because any point on the Earth's surface is accelerating upwards relative to the local inertial frame (the frame of a freely falling object near the surface). To obtain the acceleration due to motion with respect to the Earth, this "gravity offset" must be subtracted and corrections made for effects caused by the Earth's rotation relative to the inertial frame.

Gravity Sensor

Introduction:

gravity sensor measures the acceleration effect of Earth's gravity on the device enclosing the sensor. It is typically derived from the accelerometer, where other sensors (e.g. the magnetometer and the gyroscope) help to remove linear acceleration from the data. The Gravity unit are in m/s^2 like the accelerometer, and they are measured along the X, Y and Z axes.

When a device with gravity sensor is held up, the source of gravity acceleration is, of course, the gravity pull of the Earth. Therefore, when the gravity sensor discloses the X, Y, Z components of the sensed gravity, one can calculate the tilt of the device relative to the face of the Earth. Ancient gravity sensors for measuring tilt were made by a small body (e.g. a sphere) that is tettered to a wire connected to the device. Crude angular scales were used to determine the inclination of the wire relative to a perpendicular reference line to the Earth.

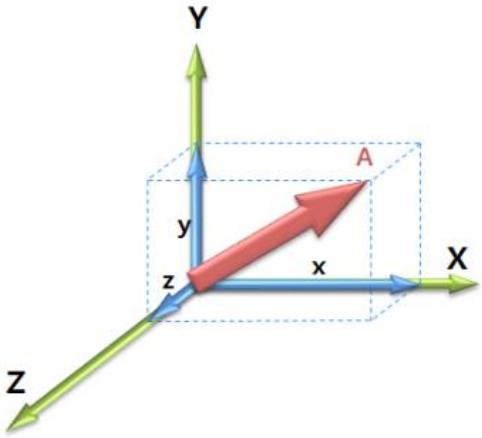
Conclusion:

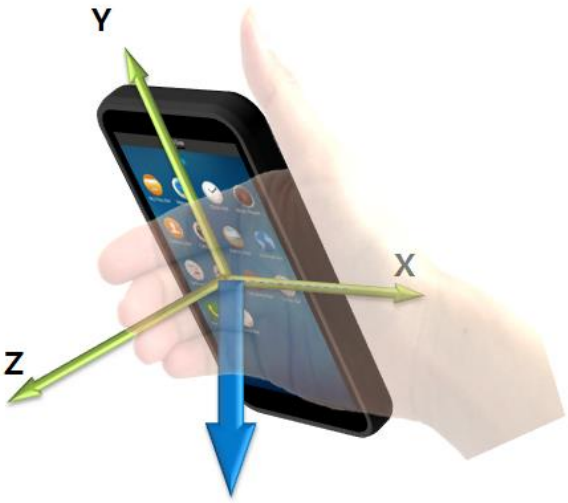
In modern mobile devices, the physical sensor that measure acceleration is the accelerometer. The accelerometer, however, measures all the accelerations that affect the device, which are the sum of the gravity acceleration and the actual linear acceleration that are associated with the movement of the device. A crude estimate of the gravity on hand-held device can be made on the accelerometer reading using a low-pass filter that minimizes the linear acceleration. Remember that a hand-held device is most of the time stationary, and cannot sustain continuous acceleration in one direction (unless the user travels within a vehicle that is subjected to wild accelerations).

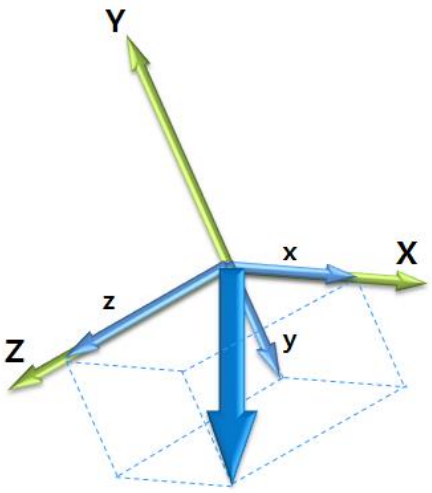
Therefore, the linear acceleration components in the accelerometer measurement are transient in nature, with a tendencies to return to 0.

Combine two sensors to get the angles using:-

1- rotation matrix that get rotation of the mobile.
2- Orientation matrix that get the angle.

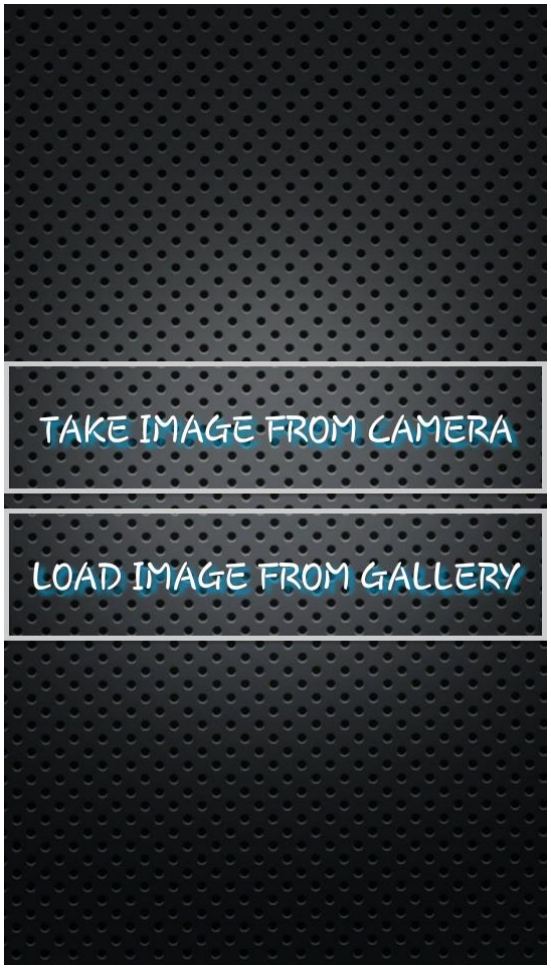






Main Menu Screen

OFF-LINE Screen Menu



ON-LINE Screen Interface

5- User Manual

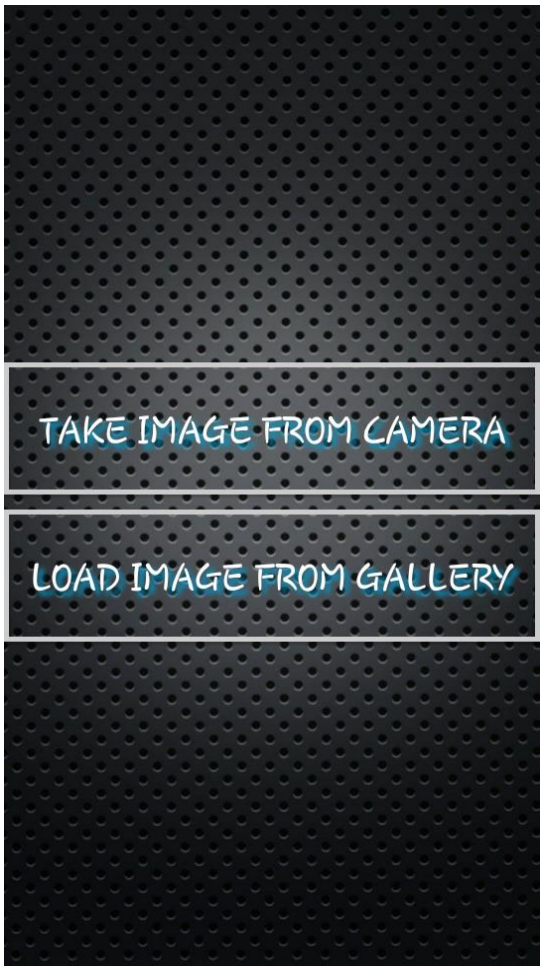
Firstly the Main Menu Screen contains two options:

OFF-LINE Mode / ON-LINE Mode.

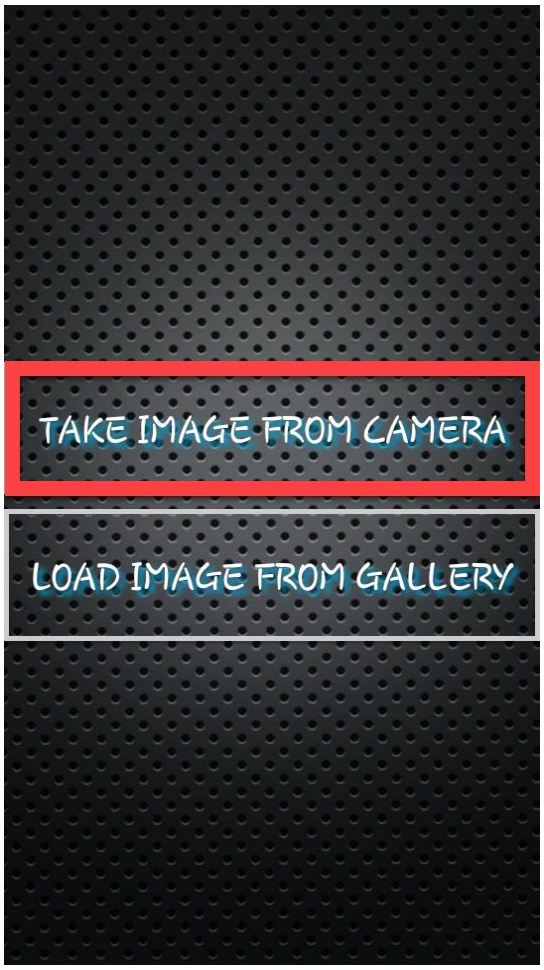
➢ Offline Mode:

There are two Buttons:

- **Take Image from camera.**
- **Load Image from gallery.**



When the Take Image from camera button is clicked the mobile's CAM App will be opened.



I-Capture image.

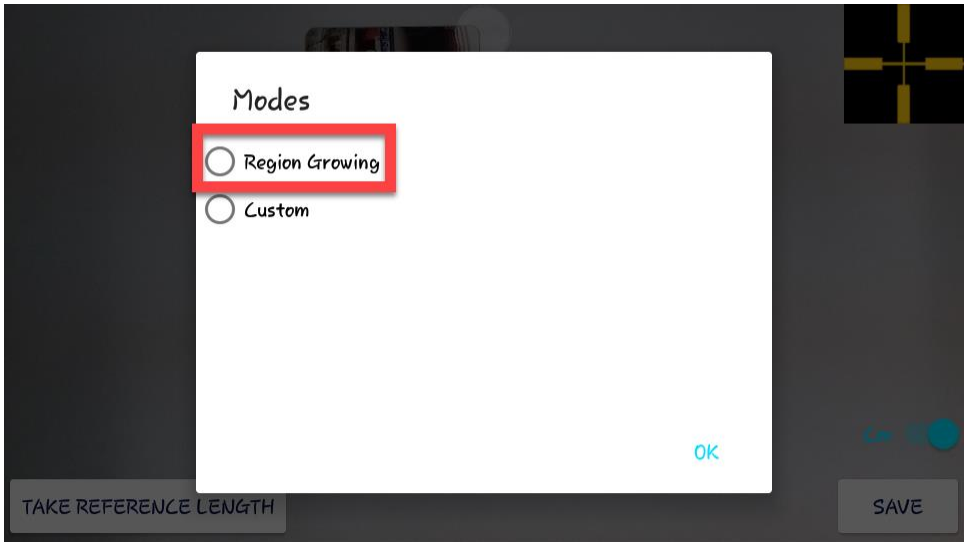
II-After capturing the image, it will be opened in new screen(MainActivity) to start your measures.

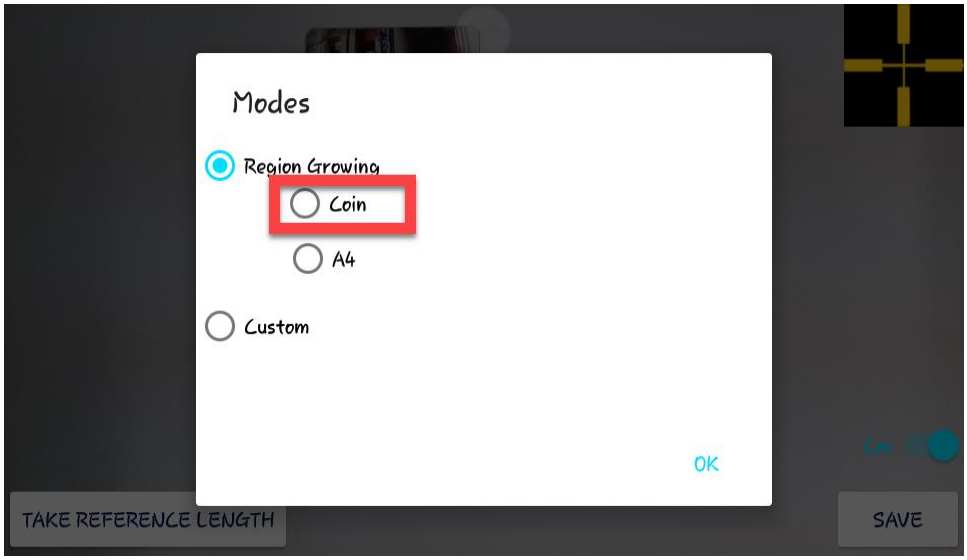
III-if the reference object in the image was **A4** or **Coin**

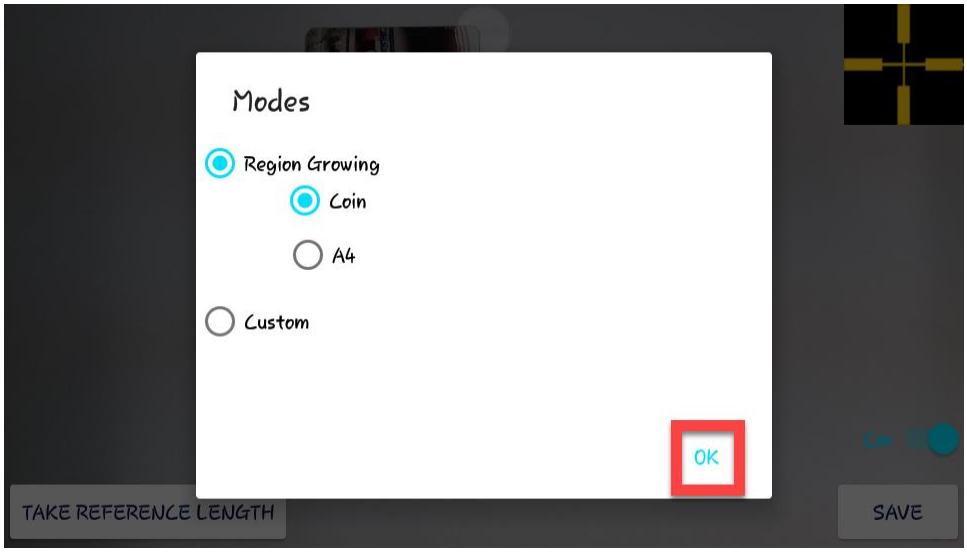
-There will be **Bullet** located at the upper center of the screen.

-Click on **Mode** button.

-New menu will be opened.
-Choose from it the **RegionGrowing** Option. Then choose the target reference(**A4**,**Coin**).Then click **OK**.

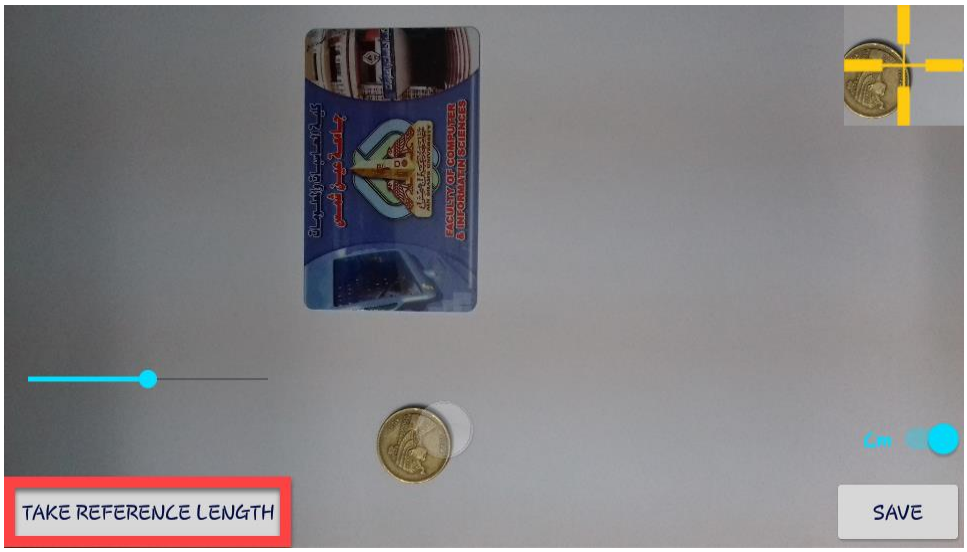






-Pull it to the reference object in the image.

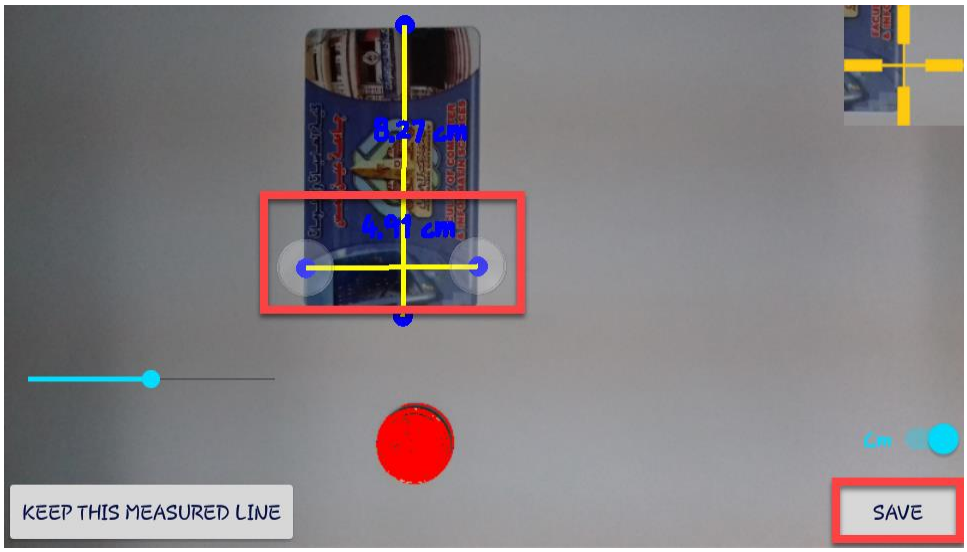
-Click on the TAKE REFERENCE LENGTH Button.

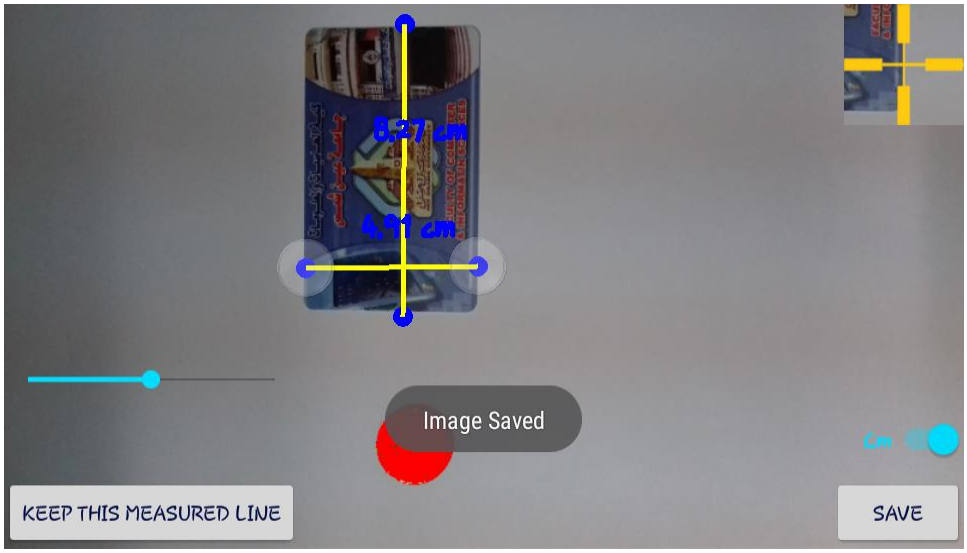


-wait moments until coloring the reference object with red color.

-if the reference object not colored correctly.
There is a slider bar, change it to change region growing threshold.

-If you want to save the image with the measurements that are calculated click on Save button.





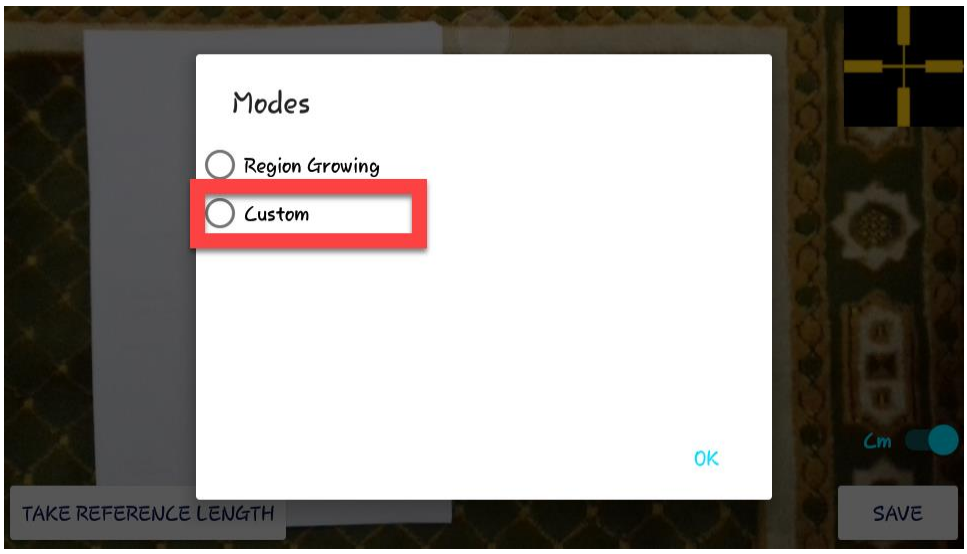
else

- it will be the Custom case.

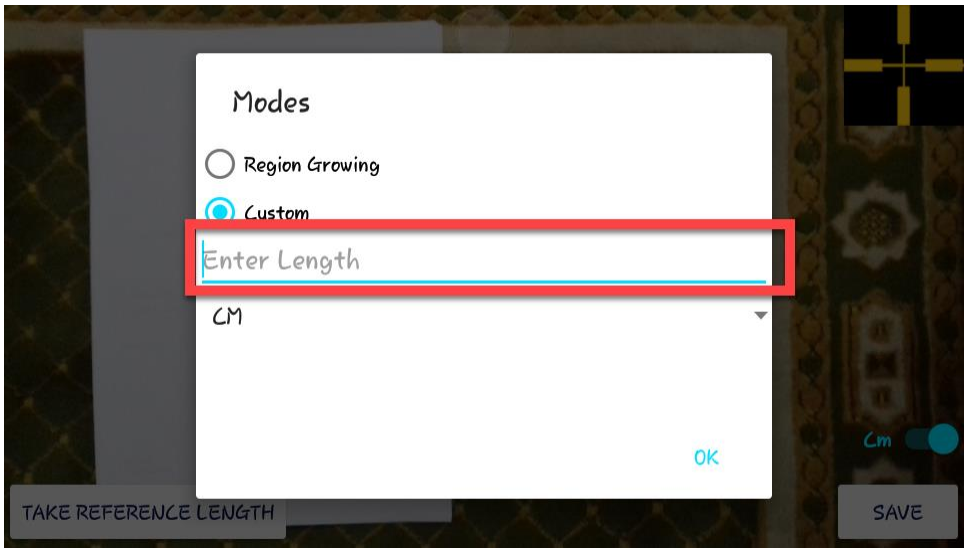
Important Note:

in this case you should know the real length (Width or Height) of the reference object.

-Choose the Custom Option.



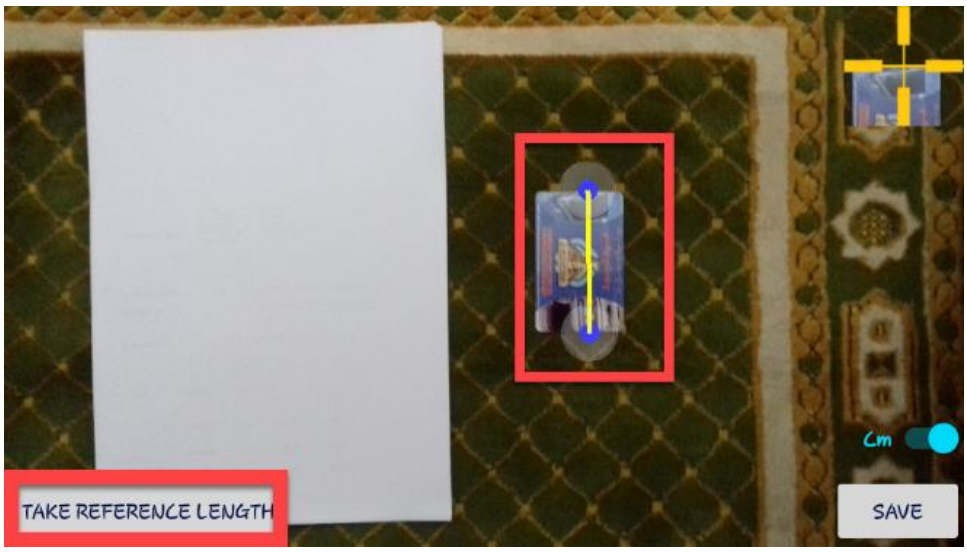
-There will be an open Edit Text.



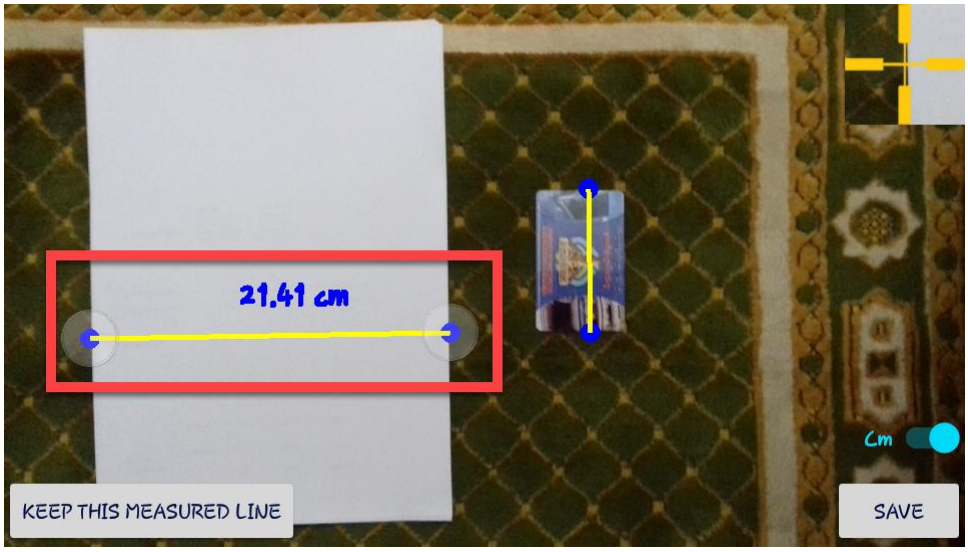
-Enter the Length that you know it then click Done.

- Two **Bullets** will appear to be used in marking the borders (width, height) of the reference object.

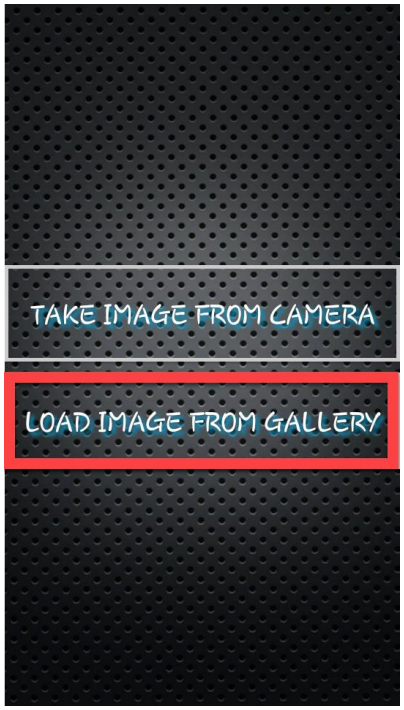
-Then Click on the **TAKE REFERENCE LENGTH** Button.



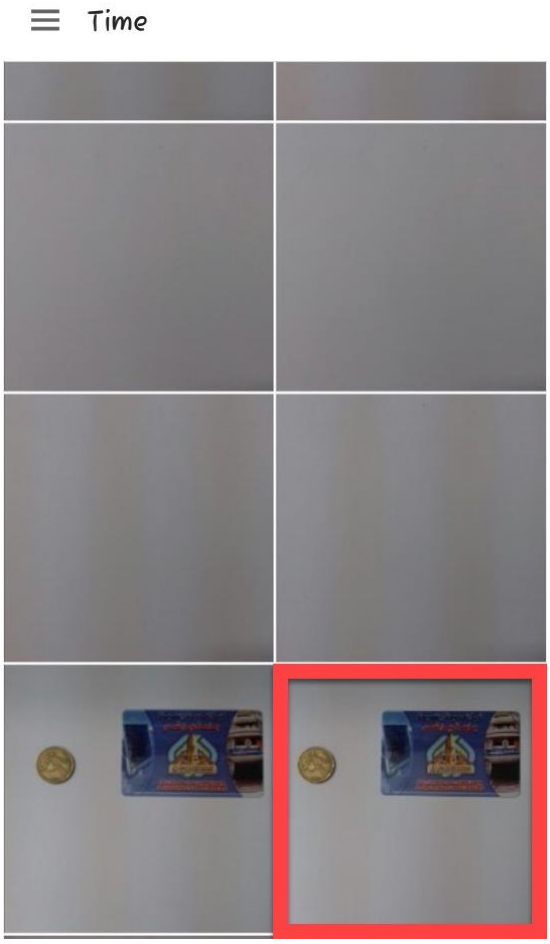
-Two new **Bullets** will appear. Use them to mark the object required to be measured.



-when **Load Image from gallery** button is clicked the mobile's gallery will be opened.



I-Choose from Gallery the image that contains the object that you want to measure.



II-After choosing the image, it will be loaded to new screen(Main Activity) to start your measures.

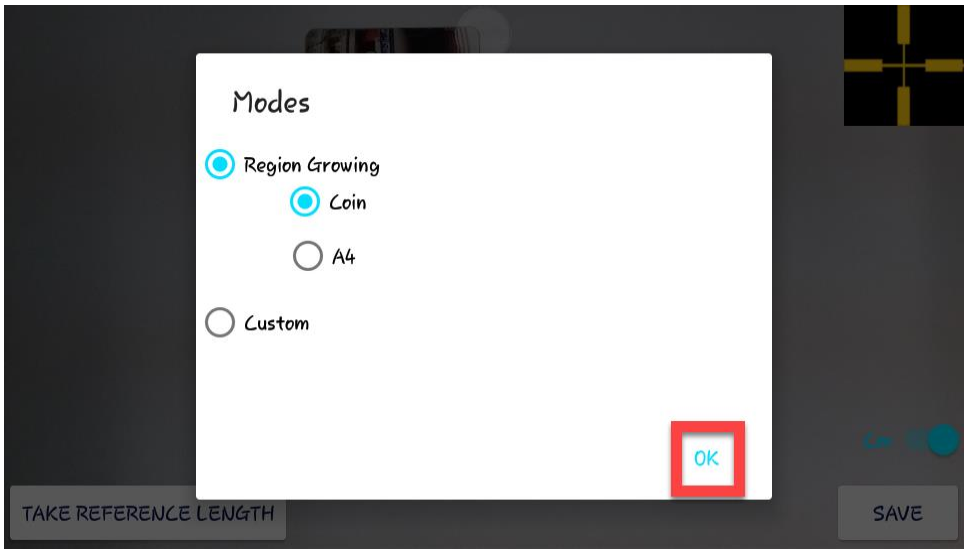
III-If the reference object in the image was **A4** or **Coin**

-there will be **Bullet** located at the upper center of the screen.

-pull it to the reference object in the image, then Click on **Mode** button.

-new menu will be opened.

-choose from it the **RegionGrowing** Option. Then choose the target reference(**A4**,**Coin**).



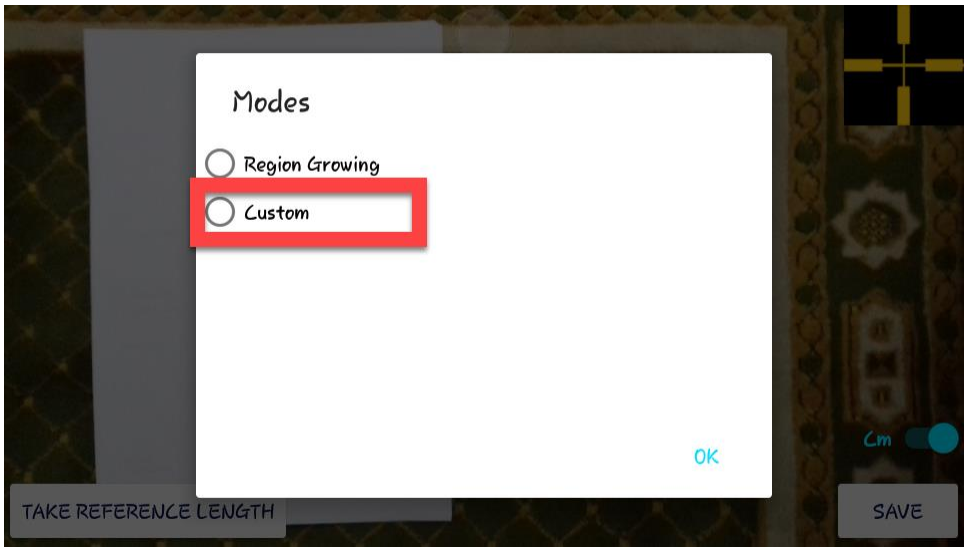
else

- it will be the Custom case.

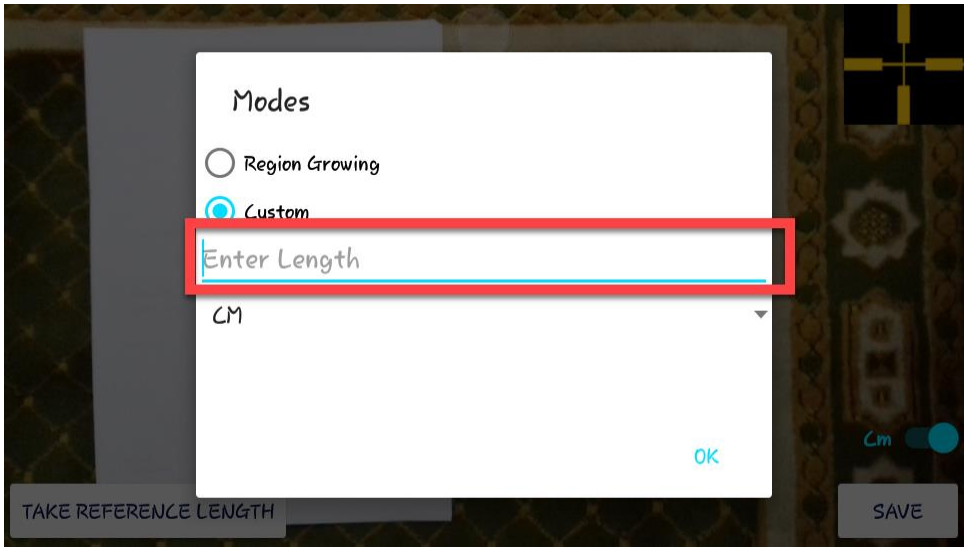
Important Note:

In this case you should know the real length (Width or Height) of the reference object.

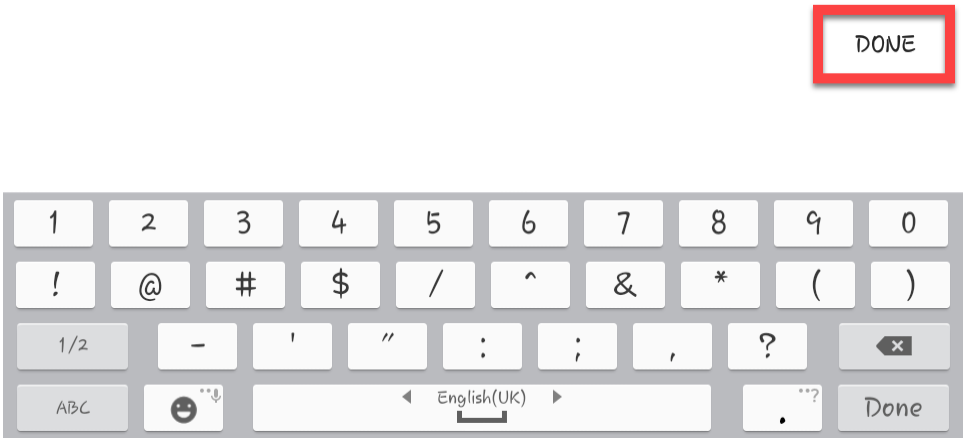
-Choose the **Custom** Option.



-There will be an open Edit Text.
-Enter the Length that you know it then click Done.

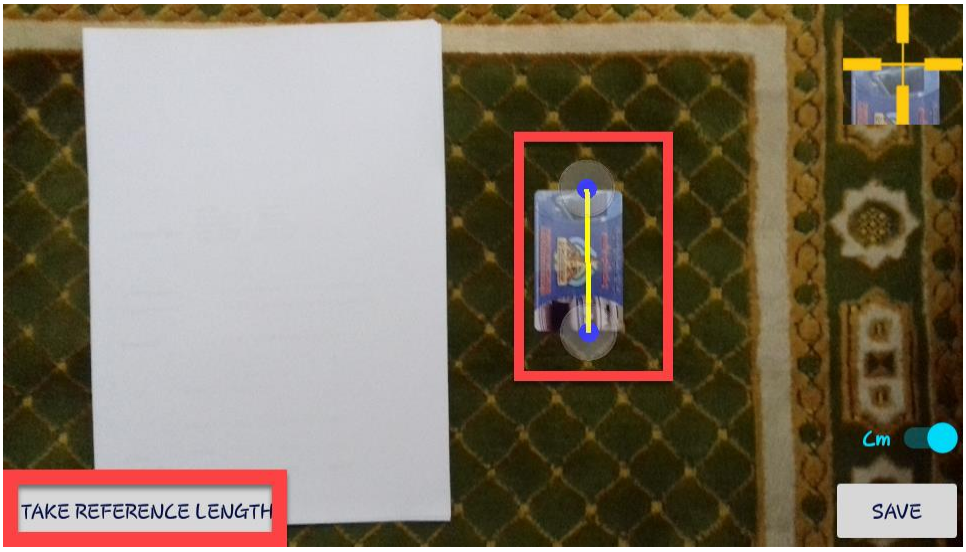


8.5

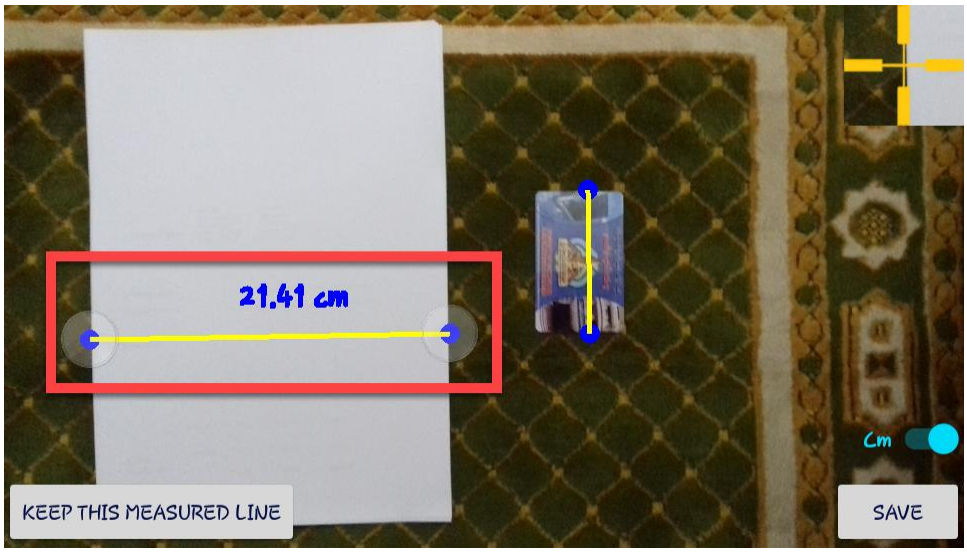


-Two Bullets will appear. Use them in marking the borders (width, height) of the reference object.

-then Click on The **TAKE REFERENCE LENGTH** Button.



-Two new **Bullets** will appear. Use them to mark the object that is required to be measured.



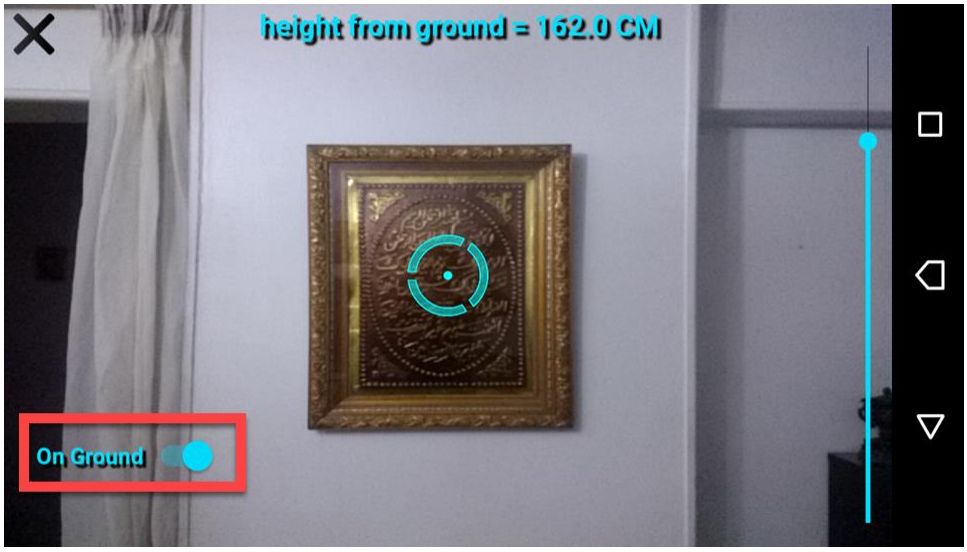
-if you want to save the image with the measurements that are calculated click on **Save** Button.

➢ *Online Mode*

-from the main menu.

-clicks on **Online** Button.

-The phone camera App. will be opened.



-Rotate the phone to landscape mode.

-Adjust the height of the mobile from the ground about the slidebar.

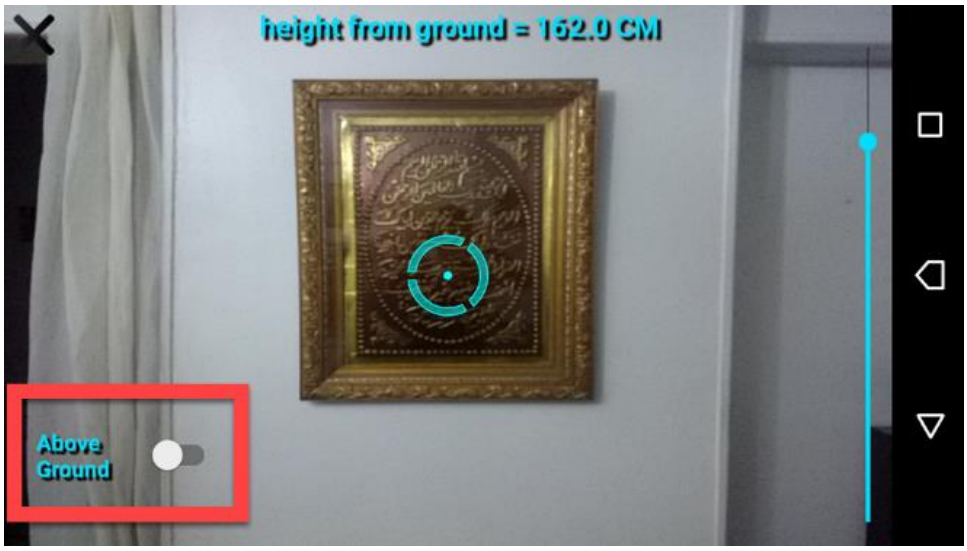
Important Note:

it's preferred but not mandatory to make the mobile height on your eye's level and in this case set the value of the slidebar to (your height – 8).

-If the object required to be measured touches ground then you should hit the crosshair twice one is for the intersection between the ground and the bottom of the object and other one is to the top of the object.

- Then height of object and distance from object will appear immediately.

-Else if the object that you want to measure it isn't touches ground then you should hit the crosshair three times.



Important Note:

You should take care about the order of taking the shots.

-First angle is taken by tapping the crosshair at the point is located at the intersection between the ground and the target object's extension.

-Second shot is taken by tapping the crosshair at the bottom of the target object.

-Third angle is taken by tapping the crosshair at the top of the target object.

- Then height of object and distance from object will appear immediately.

6- Conclusion and Future Work

6.1 Conclusion

The project is divided into two phases:-

- User knows a reference object length and wants to measure any other object within the same depth :
  1- Select the reference object from image.

  2- Measure the required object.

- User doesn't have a reference object and wants to measure objects :
  1-determine the height of the smartphone from the ground.
  2-choose method (on ground, above ground).
  3-take angles by hitting crosshair:
  - In above ground mode click on cross hair three times (on ground, begin of the object and end of the object).
  - In on ground mode click on cross hair twice (begin of the object and end of the object).

6.2 Future Work

❖ Trying to minimize the user input:

- The app automatically recognizes reference objects without user interference.
- The app automatically acquire phone height without user input.

❖ Trying to acquire more measurements:

- Get object measure in different depth of reference (offline).
- Get object width (online).
- Get object depth (online).

References

- Leszek Pawlowicz. “Smart Measure App” Internet: http://androgeoid.com/2010/09/measure-distances-heights-and-direction-with-your-android-phone-using-smart-measure/ ,September 2010.
- Jan Neggers .“Image Measurement Utility” Internet: https://www.mathworks.com/matlabcentral/fileexchange/25964-image-measurement-utility , 09 Apr 2014.
- Gonzalez and Woods. (2006) “Digital Image Processing”, 3rd edition, New Jersey: Prentice-Hall.