Frankfurt University of Applied Sciences

6/29/2016

Machine Vision:

Landmarks Detection for Pose Estimation

Supervised by:

Prof. Dr. Peter Nauth

Prepared by:

Muhammad Ehsan-Ul-Haq	1098587
Le San	1009686
Manan Malik	1098312

Department of Engineering FRA-UAS

Summer Semester- 2016

Abstract

An **autonomous intelligent system** is an intelligent system operating on an owner's behalf but without any interference of the ownership entity. Such system situated in, and part of, a technical or natural environment, which senses any or some status of that environment, and acts on it in pursuit of its own agenda. Such an agenda evolves from drives (or programmed goals). The system acts to change part of the environment or of its status and influences what it sensed. This area of engineering has found great applications in robotics. One of the main sub areas of AIS involves machine vision. **Machine vision** is the technology and methods used to provide imaging-based automatic inspection and analysis for such applications as automatic inspection, process control, and robot guidance in industry. The main idea of the project **Land Mark Detection and Pose Estimation** was to build an autonomous system which uses machine vision technique to detect some land marks in some space and then calculates the co-ordinates, distance and angles of camera with respect to that space and landmarks. The project involves circular land marks of different colors (red, green) and Microsoft Logitech USB camera. The software is written in LabVIEW 2015.

Table of Contents

Part 1: Machine Vision

1.1 Introduction

Machine vision is the technology to replace or complement manual inspections and measurements with digital cameras and image processing. The technology is used in a variety of different industries to automate the production, increase production speed and yield, and to improve product quality. Machine vision in operation can be described by a four-step flow: [1]

1. Imaging: Take an image.
2. Processing and analysis: Analyze the image to obtain a result.
3. Communication: Send the result to the system in control of the process.
4. Action: Take action depending on the vision system's result.

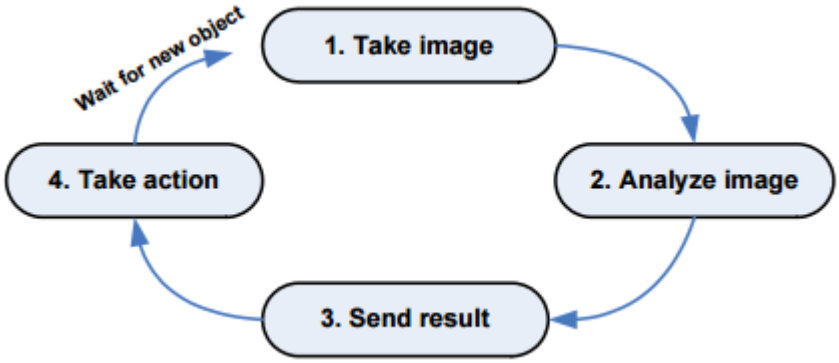


Figure 1: Machine Vision in Operation

1.2 Application Types

Machine vision applications can be divided into four types from a technical point of view: Locate, measure, inspect, and identify. [1]

1 Locate: In locating applications, the purpose of the vision system is to find the object and report its position and orientation. In robot bin picking applications the camera finds a reference coordinate on the object, and then sends the information to a robot which picks up the object

2 Measure: In measurement applications the purpose of the vision system is to measure physical dimensions of the object. Examples of physical dimensions are distance, diameter, curvature, area, height, and volume.

3 Inspect: In inspection applications the purpose of the vision system is to validate certain features, for example presence or absence of a correct label on a bottle, screws in an assembly, chocolates in a box, or defects.

4 Identify: In an identification application the vision system reads various codes and alphanumeric characters (text and numbers). A camera reads the best before date on

a food package. Examples of codes that can be read simultaneously on the same package are barcodes and matrix codes.

1.3 Landmark Detection and Pose Estimation

A mobile robot exploring an unknown environment has no absolute frame of reference for its position, other than features it detects through its sensors. Using distinguishable landmarks is one possible approach. In this approach we will use certain types of known landmarks at some known locations in a space and try to locate the position, distance and angle of camera with respect to those landmarks.

1.3.1 Landmarks Selection

There are different types of landmarks which can be used in machine vision applications e.g. square, circular, triangular with different colors, QR-Codes, Aztec-Code etc. In this project, as the main objective was distance calculation, so we decided to work with circular landmarks to keep the task as simple as possible. We used two circular landmarks (Red and Green) with equal diameters.

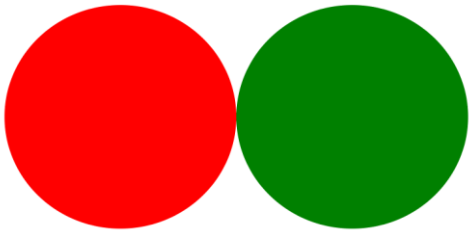


Figure 2: Circular landmarks (Red and Green) with 19.6 cm diameter

Red Landmark is the reference landmark which is put at known distance from the origin and is used to calculate the position of camera in the space e.g. x and y coordinate distance.

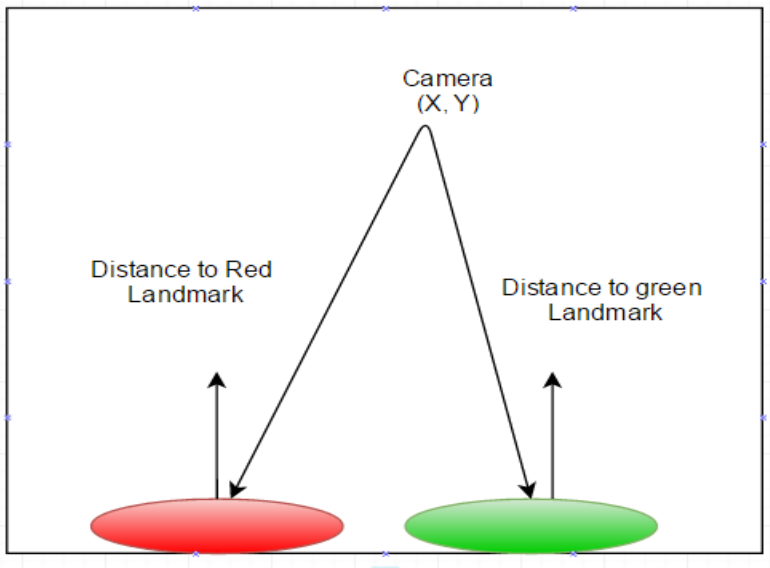


Figure 3: Placement of landmarks and camera position

While green landmark is used to make the selection between different regions of room. Another reason to use the two landmarks is to increase the efficiency of program by calculating the coordinates with respect to both landmarks and then take the mean of both coordinate points to reduce the error rate.

1.4 Camera Calibration

In this project we used Microsoft Logitech HD USB camera. A simplified camera setup consists of camera, lens, lighting, and object. The lighting illuminates the object and the reflected light is seen by the camera. The object is often referred to as target.

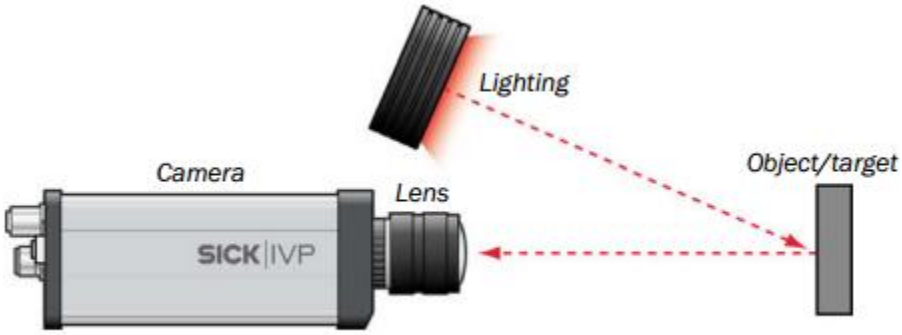


Figure 4: A simple camera and object

The lens focuses the light that enters the camera in a way that creates a sharp image. Another word for lens is objective. An image in focus means that the object edges appear sharp. If the object is out of focus, the image becomes blurred. Lenses for photography often have auto-focus, whereas lenses for machine vision either have a fixed focus or manually adjustable focus.

The main differences between lens types are their angle of view and focal length. The two terms are essentially different ways of describing the same thing. The angle of view determines how much of the visual scene the camera sees. A wide angle lens sees a larger part of the scene, whereas the small angle of view of a tele lens allows seeing details from longer distances. [1][2]

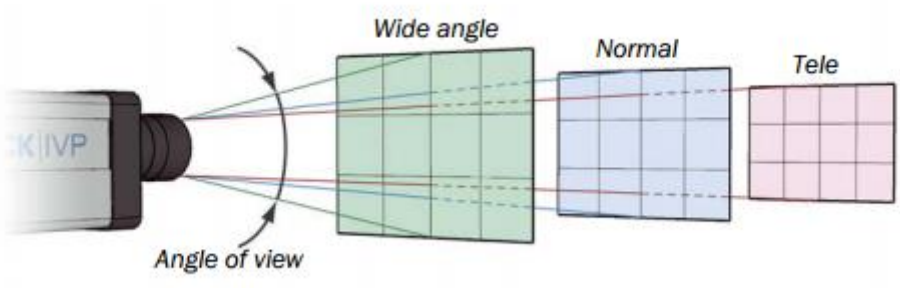
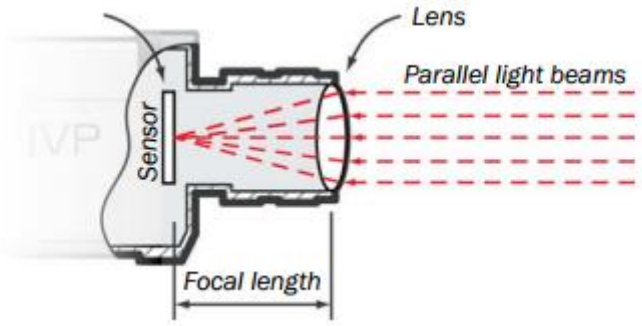
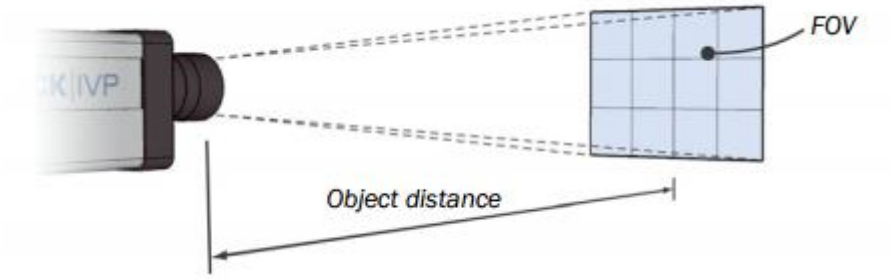


Figure 5: Lens angle of view

The focal length is the distance between the lens and the focal point. When the focal point is on the sensor, the image is in focus. Focal length is related to angle of view in that a long focal length corresponds to a small angle of view, and vice versa.



The FOV (Field of View) in 2D systems is the full area that a camera sees. The FOV is specified by its width and height. The object distance is the distance between the lens and the object.



The object distance is also called LTO (lens-to-object) distance or working distance.

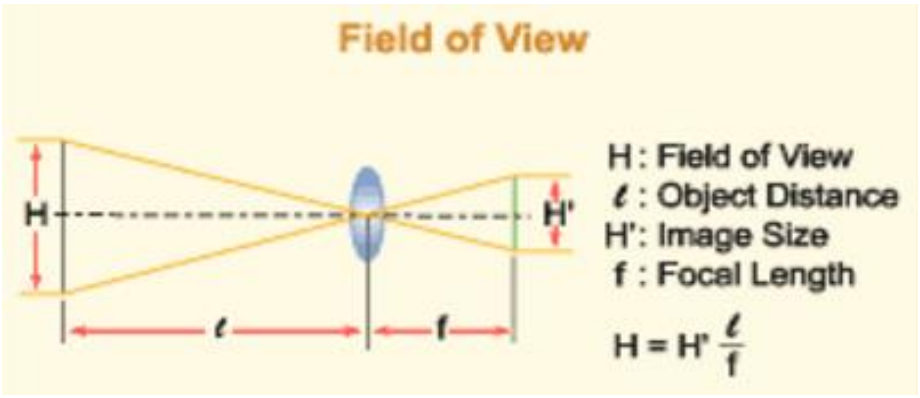


Figure 6:Object distance calculation using camera

From above image we can see that the ratio of the object size to object distance is the same as ratio of the image size to image distance. In this scenario, we already know the size of landmark (19.6 cm diameter), we can calculate image size by calculating number of pixels in the image of landmark using LabVIEW (varies with the distance), the only thing unknown is the focal length (in pixels).

$$\frac{Object\ Size\ (cm)}{Object\ distance(cm)} = \frac{Image\ Size\ (Pixels)}{focal\ length\ (Pixels)}$$

$$focal\ length\ (Pixels) = \frac{Object\ distance(cm)}{Object\ size\ (cm)} * Image\ Size\ (Pixels)$$

Manual Calibration: The object of size 19,6cm is placed in front of the camera at the distance of 44cm. The image size displayed on screen of the camera is 638 pixels. From these values, the focal length of the camera = 1432.24 pixels

Now we know the focal length 1432.24 pixels. We know the object size 19.6 cm. So we can calculate the distance by using following

$$\boldsymbol{Object\ distance} = \frac{1432.24}{Image\ size\ (Pixels)} * 19.6$$

1.5 Project State Diagram

The state machine has five stages. In the default state everything is null. If there is any Red or Green Landmark detected, it moves to the next state in which the Region of Interest of the landmark is calculated. After that, it calculates the distance and angle for red landmark, then it calculates the distance and angle for green landmark. In the end, it calculates the co-ordinates of camera with respect to origin.

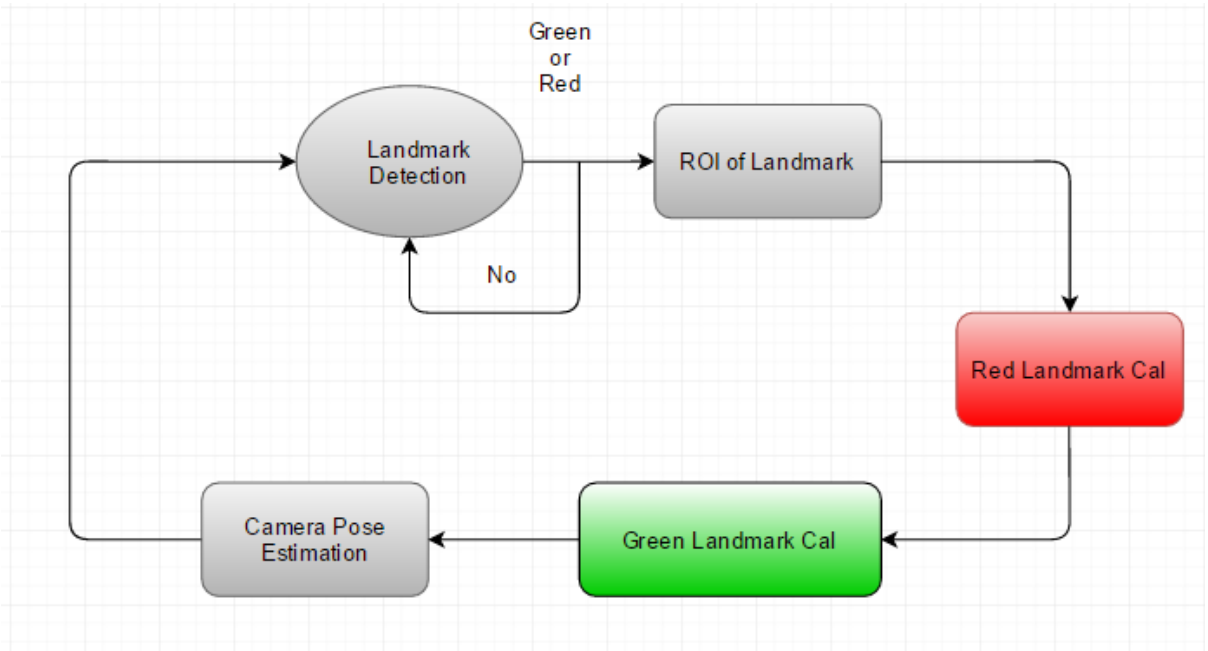


Figure 7: Project State diagram

Part 2: Program Modules LabVIEW

2.1 Introduction

LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. LabVIEW contains a comprehensive set of tools for acquiring, analyzing, displaying, and storing data, as well as tools to help you troubleshoot code you write. [3][4]

Opening a new VI from template

LabVIEW provides built-in template VIs that include the subVIs, functions, structures, and front panel objects you need to get started building common measurement applications. Complete the following steps to create a VI that generates a signal and displays it in the front panel window.

1. Launch LabVIEW.
2. Select **File»New** to display the New dialog box.
3. From the **Create New** list, select **VI»From Template»Tutorial (Getting Started)» Generate and Display**. This template VI generates and displays a signal. A preview and a brief description of the template VI appear in the Description section.
4. Click the OK button to create a VI from the template. You also can double-click the name of the template VI in the Create New list to create a VI from a template. LabVIEW displays two windows: the front panel window and the block diagram window.

Functions and Controls Palette

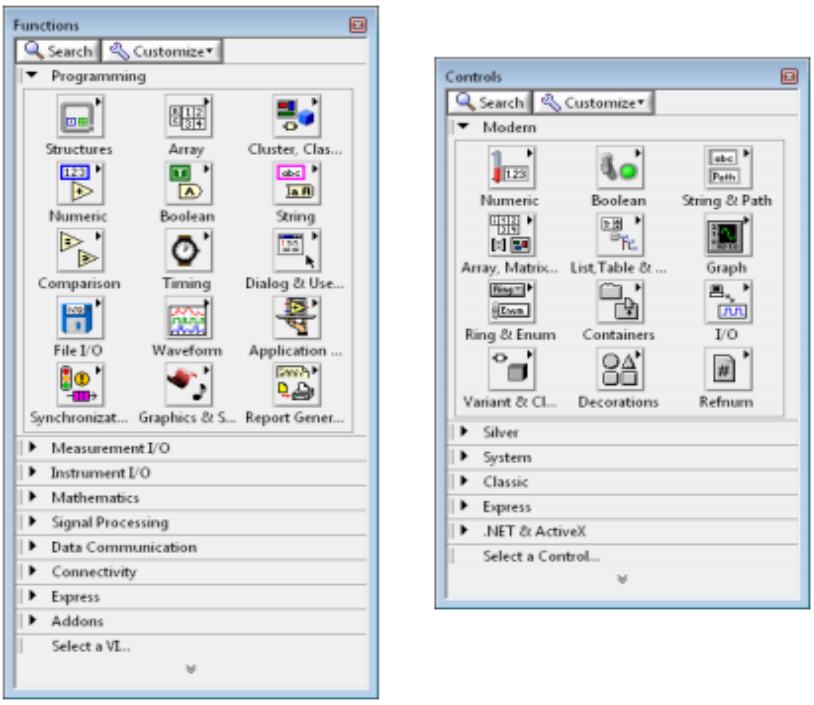


Figure 8: Functions and Control palette LabVIEW

2.2 State Machine

As we discussed earlier, program comprises of state machine. It has total of five stages. Following are the block diagrams of all five states.

2.2.1 Landmark detection

This is the default state of the program. When there is no landmark in the field of view of camera, it gives null value to all the variables. This state further involves following steps.

Step 1: Color Filter

Threshold: Color filter is used to filter out the color from a landmark. It smooths image, remove noise and finds edges. To create a custom color filter, first we applied a color threshold. For red filter we applied threshold in RGB and for green filter we applied threshold in HSV. The reason for using RGB mode for red is that HSV mode for red color has value range -10 to +10 and in LabVIEW we have range 0 to 255 and can't map negative values. The value of Red is set to 110. For green color filter we used HSV mode with Hue, Saturation and Value (55, 70,70) respectively.

Morphology: After setting threshold, we applied morphology with Auto Median mode for reducing noise outside landmark. Morphology removes particles that touch borders, noisy and unwanted particles. After that, we used 3*3 erosion with an erosion value of 5 to reduce the noise inside landmark. We tested our filter with different values of erosion, it gives best results with 5.

Binarization: Next, we binaries the image with a replace value of 255, which turns the landmark region to white and everything else to black.

Measuring and Counting: In the end, we used IMAQ Count Objects function with a setting "Bright Objects" to detect the and count the number of landmarks. Optional filters give the capability to ignore the objects smaller or larger than given sizes. Other options allow rejecting the objects touching the borders of the search area and ignoring the holes that the segmentation process may create in the objects. The segmented objects are then located and measured. Following page shows the block diagrams for both Red and Green filters.

Step 2: Object center co-ordinates

After done filtering, detecting and counting landmarks, we calculated the center co-ordinates of landmarks with respect to field of view.

Step 3: Bounding Box

We also drew bounding boxes around landmarks to calculate the length of vertical and horizontal side. As the camera moves along XY plane the length of horizontal side changes. While the length of vertical side remains the same. This change in the length, we will use later to calculate angle.

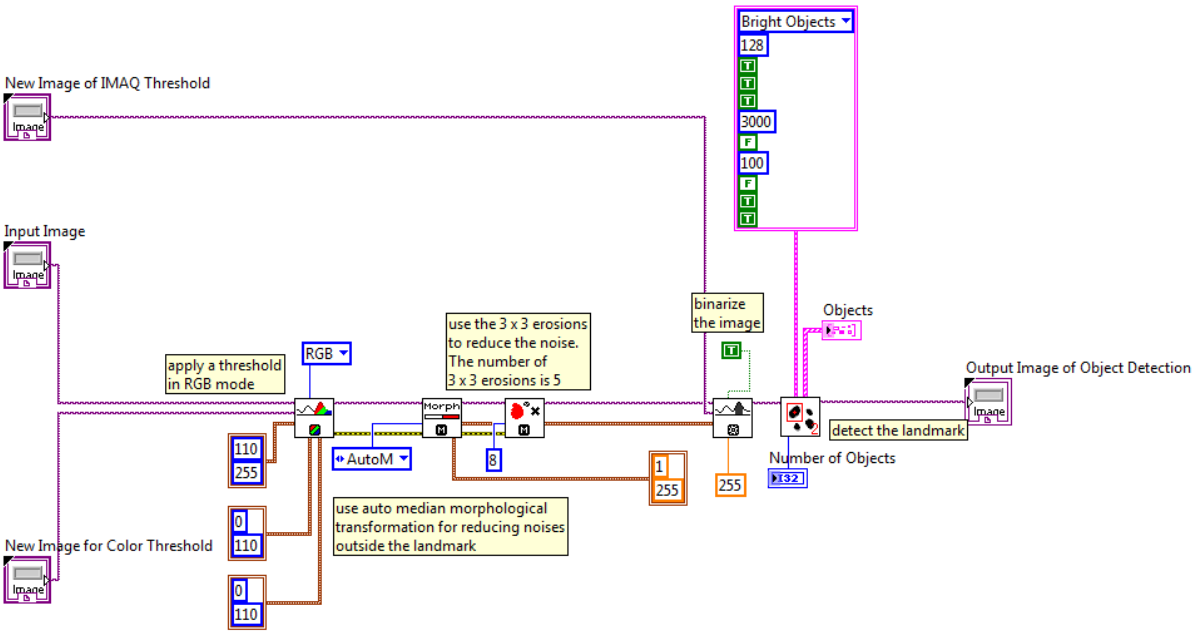


Figure 9: Red color filter

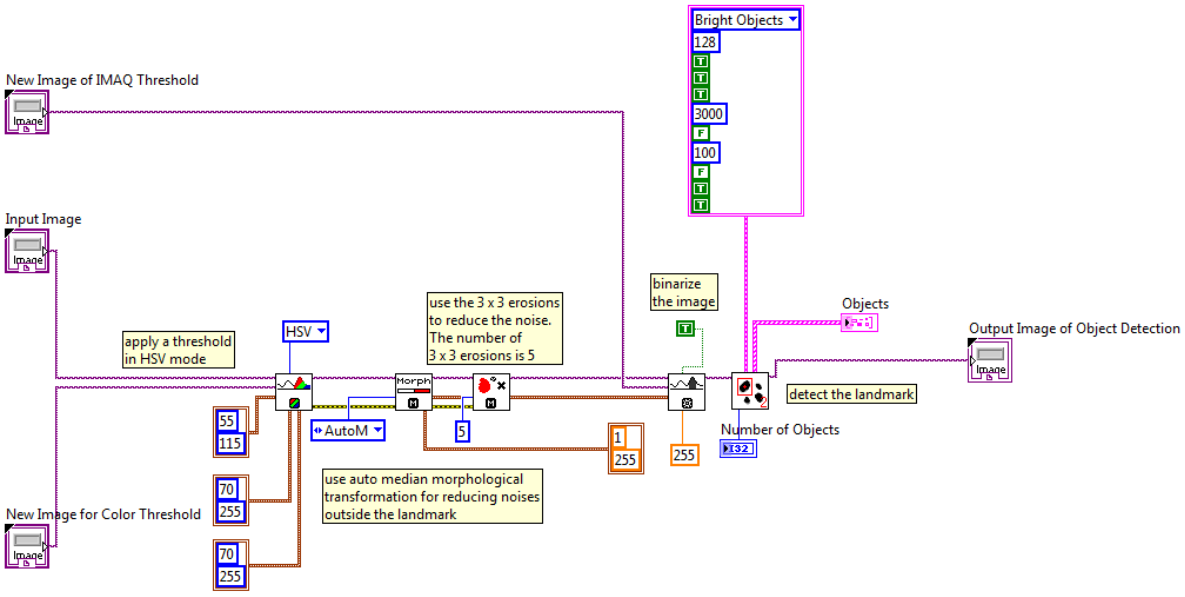


Figure 10: Green color filter

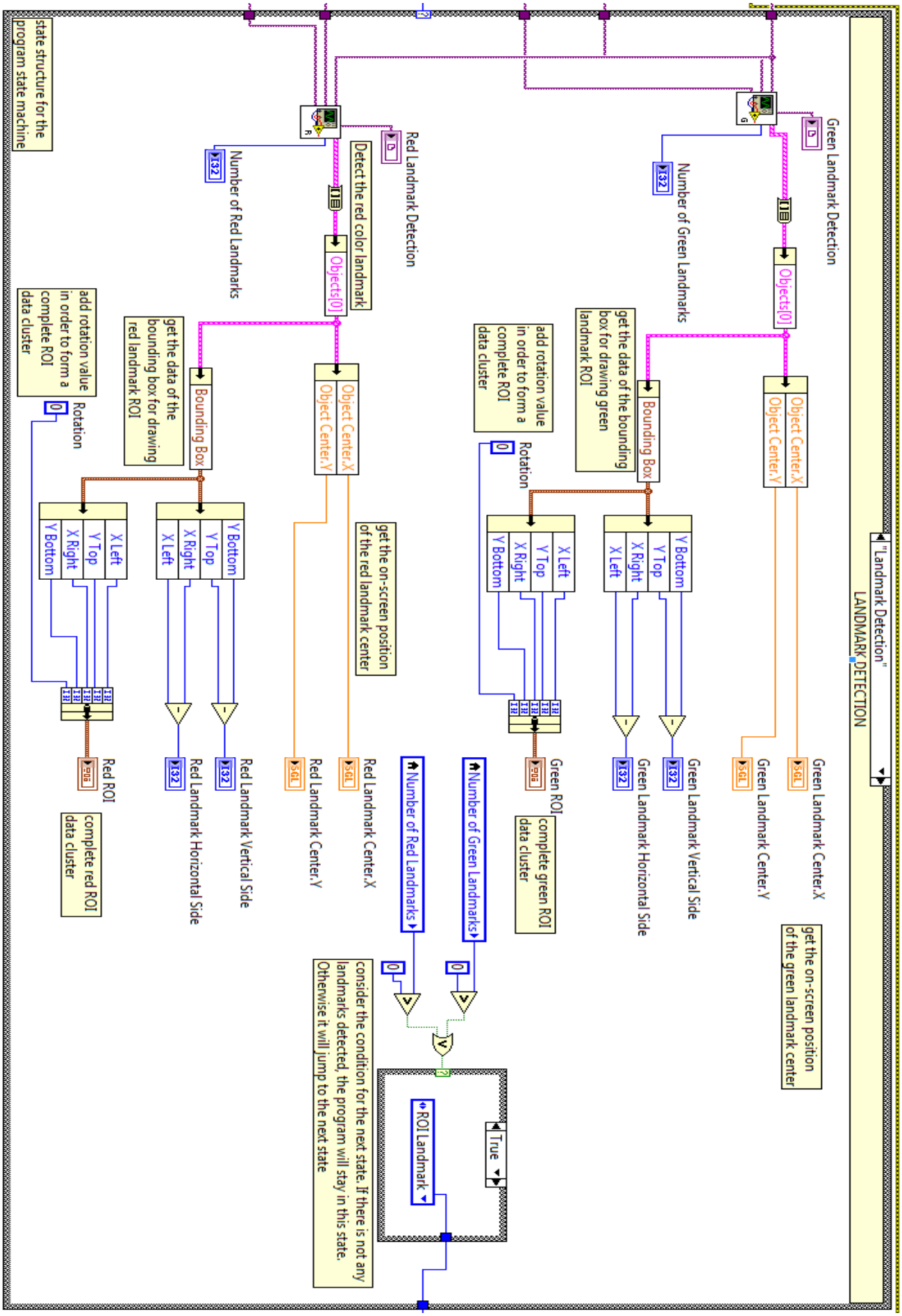


Figure 11: Default State of state machine

2.2.2 ROI of Landmark

If program detects a landmark, it jumps to this state. Here, it gets the ROI data from previous state and use overlaying images to add two bounding boxes to the original images. Following block diagram shows the workflow.

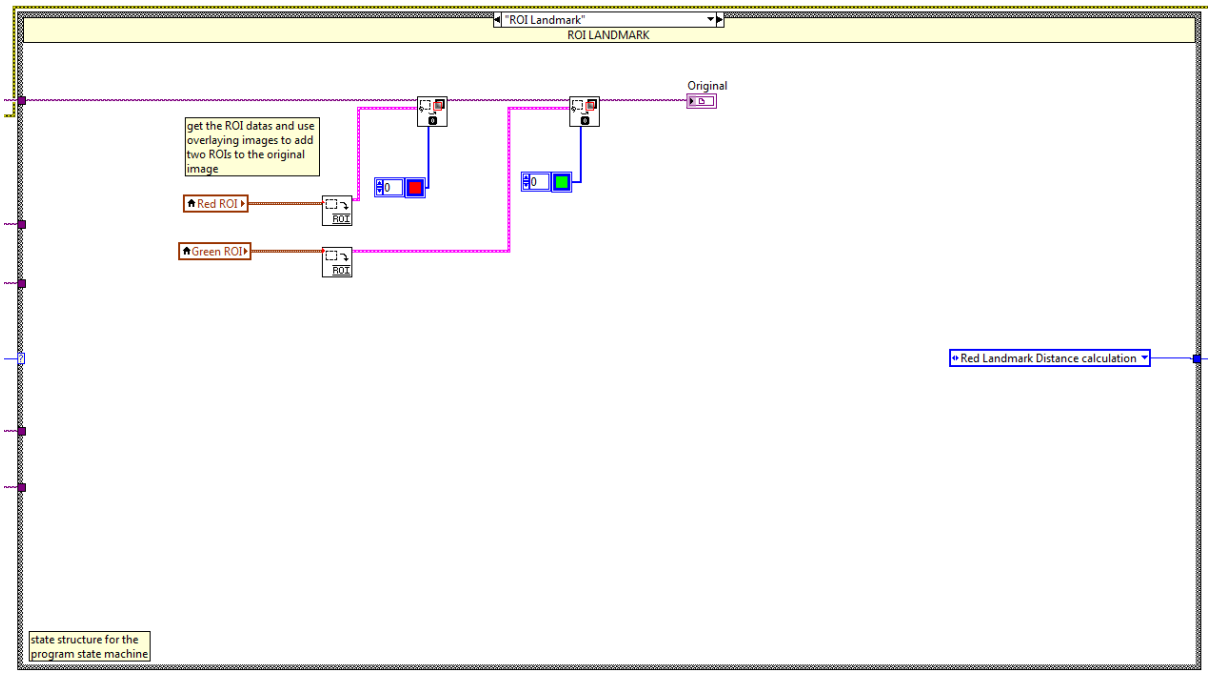


Figure 12: ROI of Landmarks state

2.2.3 Red Landmark Calculation

As we already calibrated camera by placing the object of size 19.6cm is placed in front of the camera at the distance of 44cm.The image size displayed on screen of the camera is 638 pixels. From these values, when changing the camera position, the new distance can be calculated. In the math script, we used if-else condition. If n (number of landmarks) is not equal to zero, then we calculate distance and angle. To calculate distance first, we calculated d by taking the difference of x co-ordinate values of field of view and center of land mark (multiplied with side_of_pixel to convert from pixels to cm). After that we applied Pythagorean theorem to calculate distance. Similarly, we calculated angle by using arctan(d/distance_at_center)

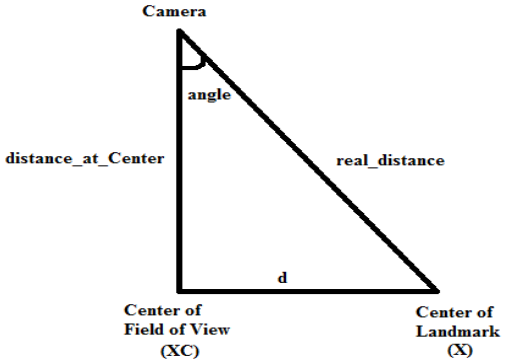


Figure 13: Realization of Distance calculation

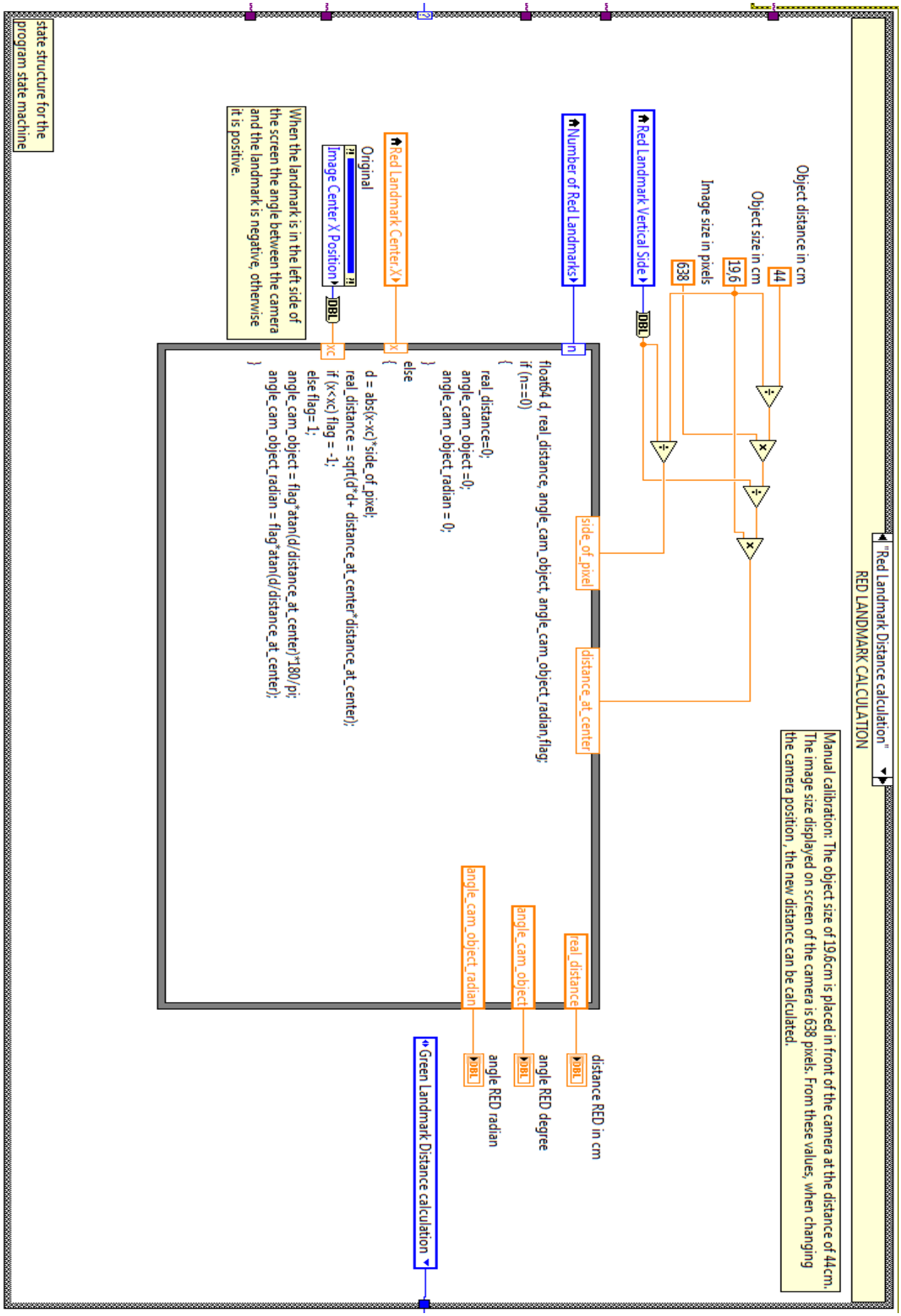


Figure 14:Red landmark distance calculation

2.2.4 Green Landmark Calculation

Just like previous case we calculated the distance and angle for green landmark.

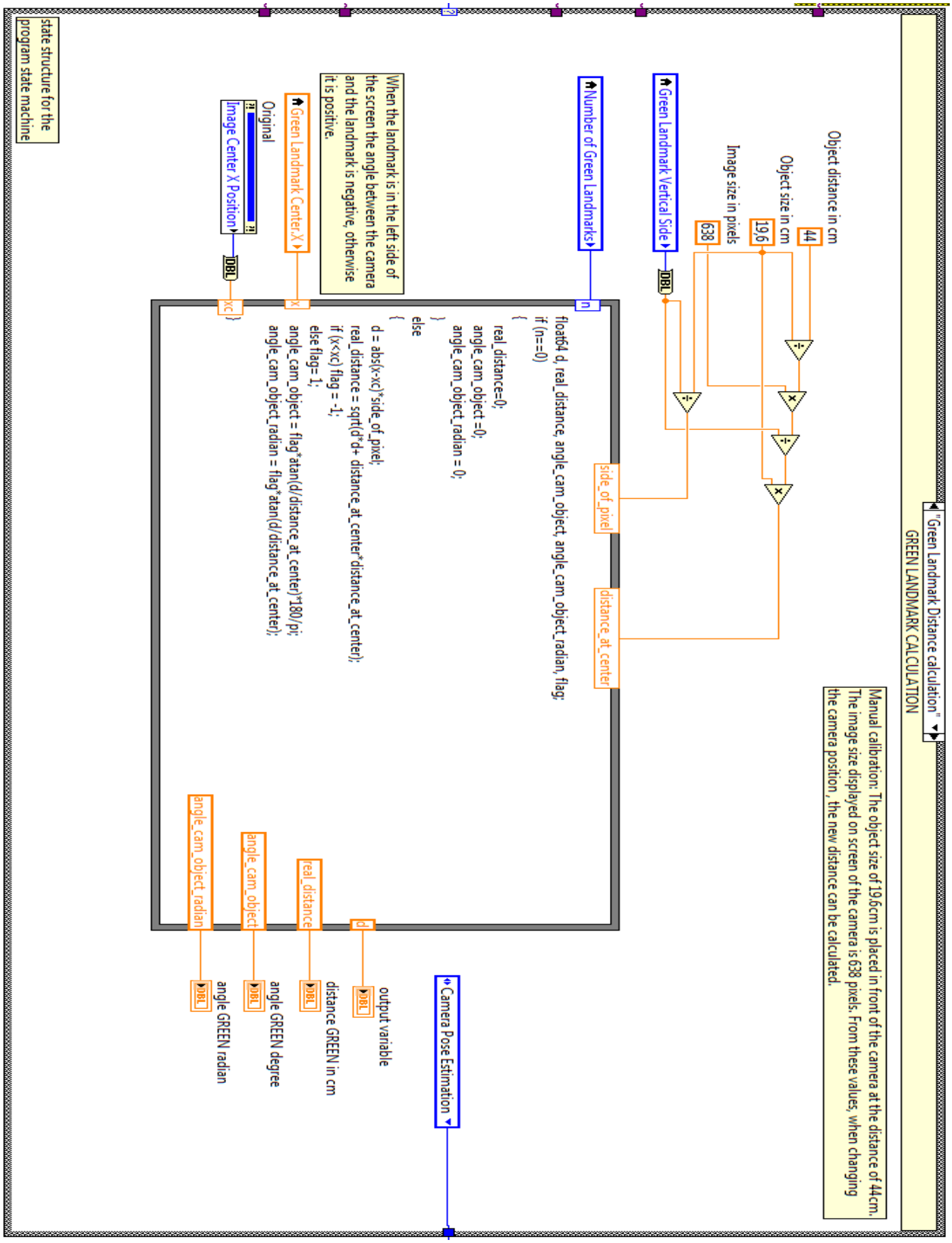


Figure 15:Green landmark distance calculation

2.2.5 Camera Pose Estimation

Considering the positions of two landmarks on the screen so that the position of the camera could be estimated. There are 3 cases. Case 1 and case 2 is when the camera is at either side of both landmarks. Case 3 is when the camera stands in the middle of the red and green landmarks.

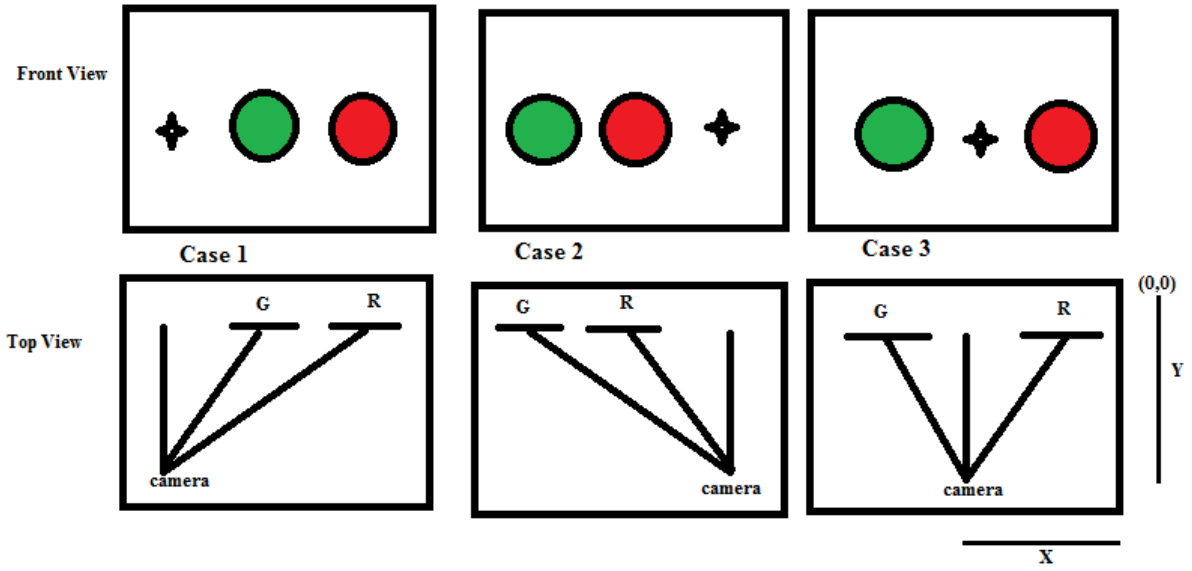


Figure 16: Three cases for pose estimation

In **Case 1**: we add the value of x_red or x_green and y_red or y_green to get the co-ordinates with respect to origin

In **Case 2**: we subtract the value of x_red or x_green and y_red or y_green to get the co-ordinates with respect to origin.

In **Case 3**: we again add the value of x_red or x_green and y_red or y_green to get the co-ordinates with respect to origin.

After getting the co-ordinates with respect to both red and green we take mean of both values to reduce the error.

Following block diagram shows the Case 1

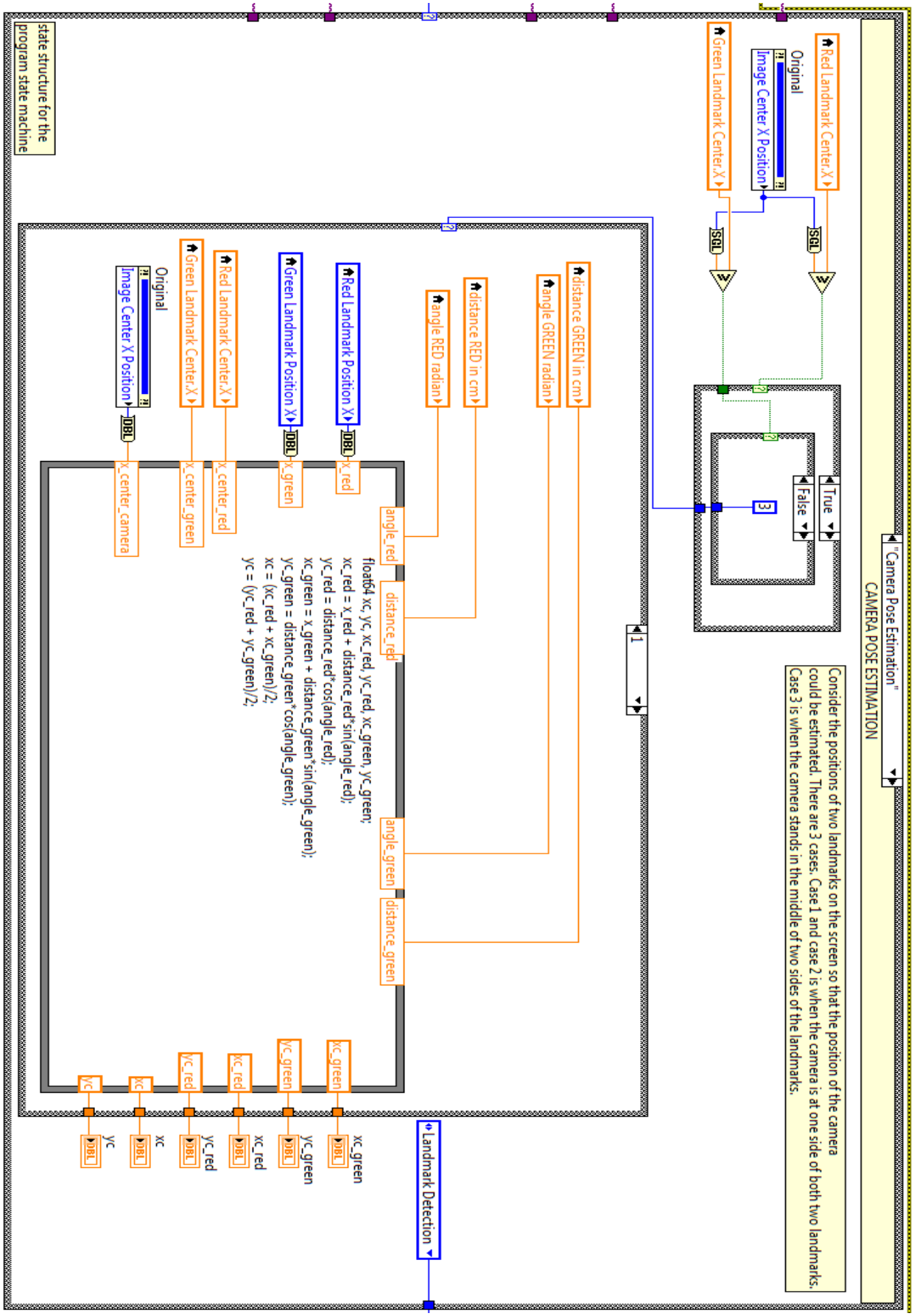


Figure 17: Case 1 for pose estimation

2.3 Final Application

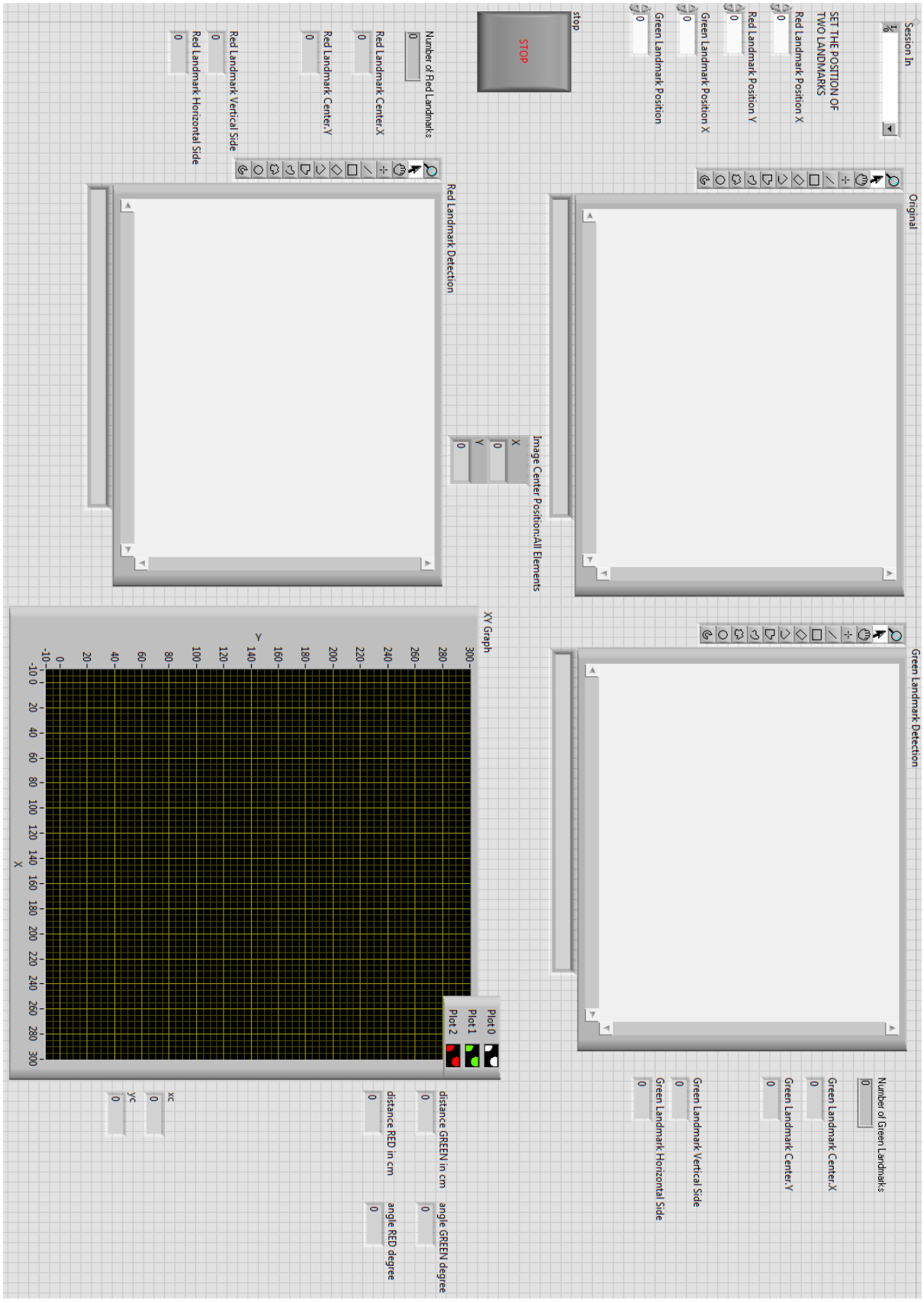


Figure 18: Final application for Landmark detection and pose estimation

Part 3: Program Testing

In order to testify the efficiency of the program, two experiments were processed. The first experiment was made to measure an object distance, while the second one was used for estimating the camera position.

3.1 Distance measuring

In the first experiment a green landmark was set in front of the camera. During the measurement, the camera was moved gradually far away from the landmark until it cannot detect the landmark. For an amount of displacement, the distance between the object and the camera was measured by two methods: the program and a distance measuring device, which has a higher accuracy (the variance is ±1mm). Then we compared the two results to find out the error of the program.

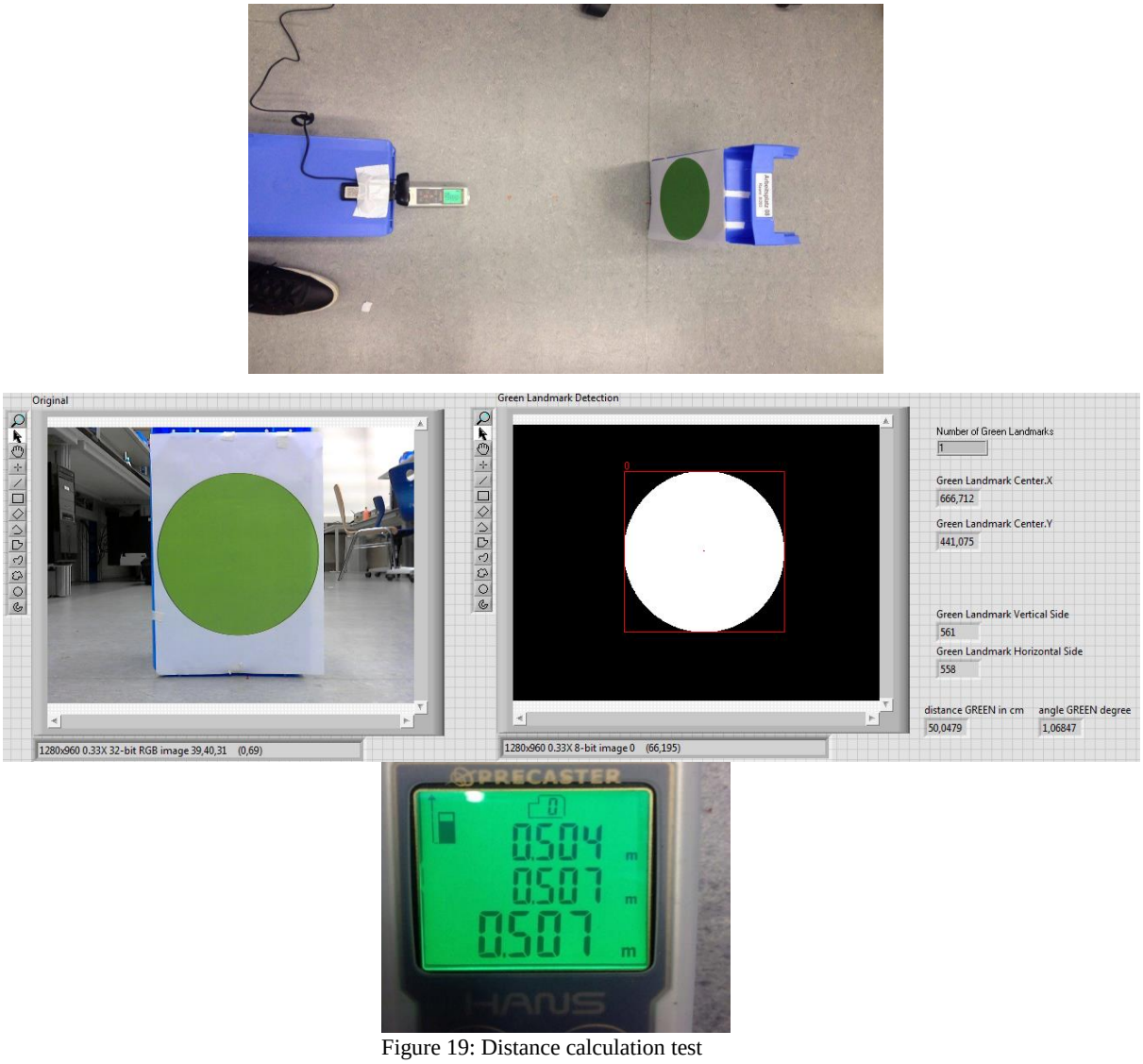


Figure 19: Distance calculation test

Distance measured by LABVIEW program (cm)	Distance Measured by Device (cm)	Error (%)
39.7	40.8	2.7
50	50.7	1.4
59.6	60.3	1.2
70.2	71	1.1
80.9	79.9	1.3
91.2	90	1.3
100.3	98.5	1.8
201	198.4	1.3
330.6	318	4
413.4	396.6	4.2
425.7	413.2	3

From the measuring, we know that the distance range for the program working effectively is from 40cm to 4m, and the average error of the LABVIEW program is approximately 2%. There was also a problem with the landmark recognition part, that it might also detect the other object having the same color with the landmark.

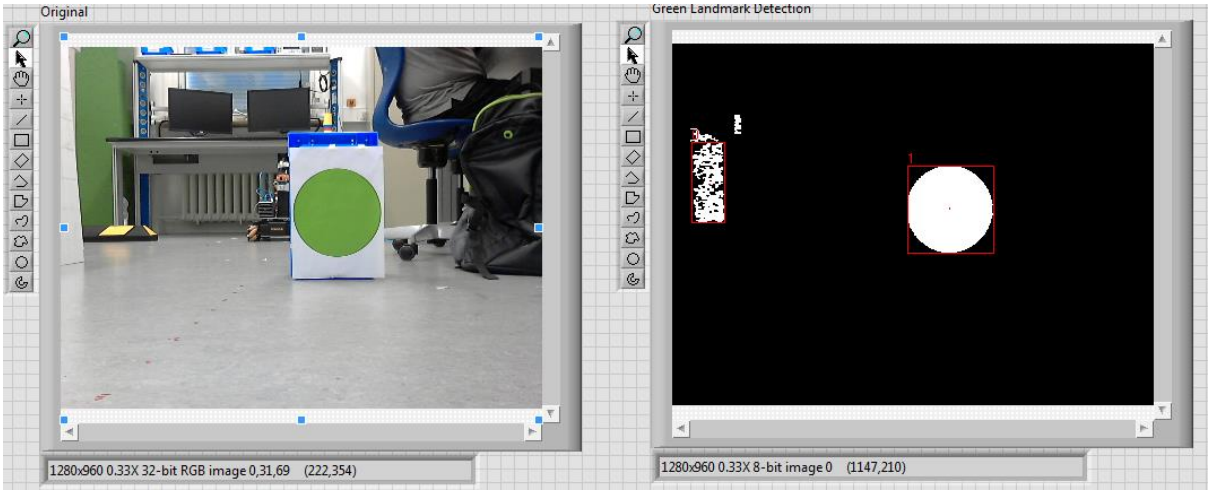


Figure 20: Objects other than landmark detection

3.2 Camera Pose Estimation

In the second experiment two landmarks red and green were set with a known position, while the camera was moving in front of these objects. The camera position then could be calculated and simulated on the main screen.

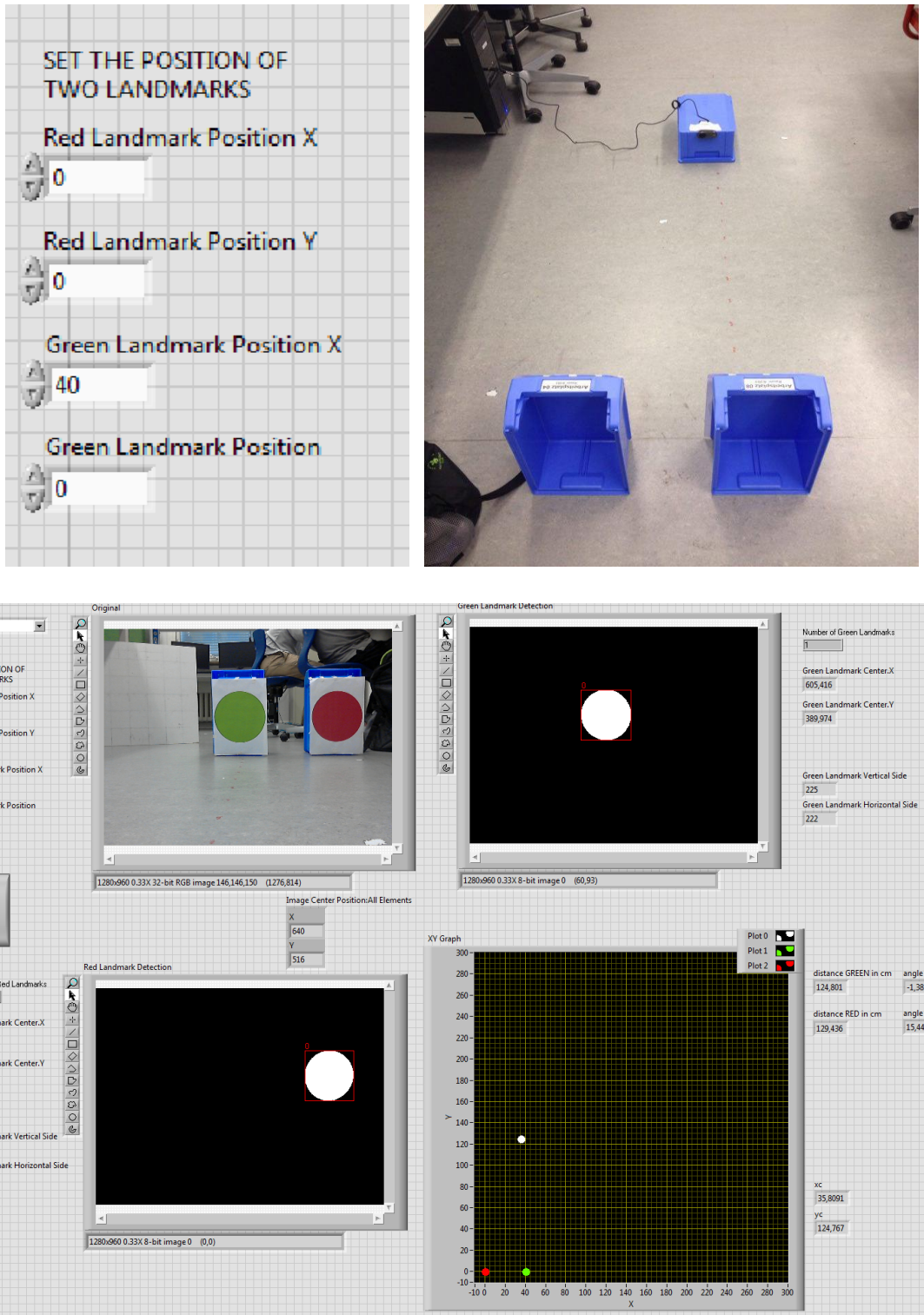


Figure 21: Camera pose estimation, complete experiment

Part 4: Limitation and Future Improvement

4.1 Limitation

Although we achieve the main goal of the project, determining the camera pose, there are still several limitations throughout the work.

Firstly, we have not made an auto calibration for the camera, which means the program only runs well for the Logitech webcam. If another camera is applied, we will need to redo the manual calibration step.

Secondly, the color landmark detection is just effective in a simple environment. If there are colorful objects surrounding the camera, the algorithm might go wrong.

Finally, the camera can only stand in one direction so that the camera optical axis is perpendicular to the plane containing the circle landmark. When the angle of the camera is changed, the output accuracy decreases.

4.2 Future Improvement

In other to reduce the above limitations, there are some points that need to be improved in the future.

The first thing is creating an auto calibration step, so that the program can be compatible with variable cameras.

The second thing is replacing the circle color landmark with QR code, since it can remove the influences from the ambient objects. The QR codes can also store its position information.

The third improvement, is further enhancing the project and calculating the position of camera in 3D space.

The last improvement is if we know precisely the behavior of the error, we could enhance the accuracy of the measurement.

References

[1] Machine Vision, SICK IVP, version 2.2, December 2006; www.sickivp.com

[2] J. Adam Jones; Evan A. Suma; David M. Krum, Comparability of Narrow and Wide Field-Of-View Head-Mounted Displays for Medium-Field Distance Judgments; Institute for Creative Technologies, University of Southern California

[3] Getting Started with LabVIEW, User manual June 2013
http://www.ni.com/pdf/manuals/373427j.pdf

[4] http://www.ni.com/manuals/

Table of figures