UNSYS digital

Journal of Unmanned System Technology

Development of Simulink-Based Object Detection Program for a Camera for Micro-Aerial Vehicle

Sutthiphong Spot Srigrarom

University of Glasgow Singapore, Singapore

***Abstract*—**This paper presents research and development ofour in-house object detection program for a digital camera that can be used in conjunction with a microprocessor on a micro-aerial vehicle (MAV) for autonomous flight in an indoor environment. The object detection program functions to create object boundaries and edges found in the video recorded by the camera thus creating boundaries of an environment. This work will use video and image analysis techniques to build colour co-relations and edges of the objects. The results from the analysis can then be passed to the microprocessor for processing. This then enables the micro-aerial vehicle to be able to avoid and head towards objects in its surrounding autonomously. This work is meant to be used for a 3D indoor navigation system, which is able to aid MAVs to safely navigate through the unknown and complicated indoor environment and complete autonomously necessary flight missions.

***Keywords*—**Simulink, object detection, computer vision.

I. Introduction

Today, UAVs come in different shapes and sizes and conventional flight have been pushed aside. Aside from the traditional helicopter and fixed wing aeroplanes, multi-rotor vehicles have been gaining popularity. Some examples of these unconventional vehicles are the tri-copter and quad-copters. These vehicles unlike a traditional helicopter use more than one rotor to function. Quad-copters have attributes similar to that of a traditional helicopter with additional stability control. With four rotors spinning in opposite directions, speed and attitude controls are depended solely on power distribution to the rotors by the microprocessor. Flight stability of the quad copter has also increased with four rotors. Programmable microprocessors with auto stabilization function help to stabilize these vehicles in flight as well.

In an outdoor environment, with the aid of global positioning system (GPS) modules, UAVs can perform automated task such as flying a specific route defined by the user and circling around a perimeter specified by the user. With the aid of barometers and sonar sensors, UAVs are able to maintain specific heights to hover and deploy. Thus, with these sensors and GPS modules, UAVs are able to fully function autonomously in an outdoor environment.

However, in an indoor environment, the scenario is different due to limitations of GPS triangulations. This is due to weak signal strength of GPS satellite in an indoor environment. Thus, UAVs are not able to function fully autonomously as they cannot detect their exact location in reference to that of indoor environment. To overcome this issue in an indoor environment, UAVs have to be installed with additional sensors that help locate its position in reference to that of the environment. These additional sensors include a camera system that records real time video of the environment.

By developing an object detection program that detects key surfaces and references of the environment will enable the UAV to locate its position in reference to the surroundings. Used in conjunction with a camera system, results from the program can then be passed to the main controller module of the UAV thus enabling autonomous flight in indoor environments. This will then enable automation function of UAVs in an indoor environment.

The objective of this work is to develop an object detection program for UAVs or micro-aerial vehicle camera systems. The program will take video inputs from the camera and detect objects in the video allowing the UAV or micro-aerial vehicle to be able to fly autonomously towards or away from obstacles. The information processed by this program will enable the UAV to detect objects or key surfaces in its surrounding.

This work will consist of the following stages of development:

1. Video recording phase (to simulate real time video of the camera system in the micro-aerial vehicle)
2. Video rendering phase (splitting the video into still picture frames for analysis)
3. Initial development of the program
4. Co-relate each frame of the video using programming software such as MATLAB to find difference in the frame.
5. Testing and trail phase (different videos with different objects are tested to test the robustness of the program)
6. Implementation corrections from testing and trial data

II. Methodology

The proposed method for tackling in this work will be in-line with image and video analysing techniques. There will be a few stages for the development of the object detection program. The main component to this work will be the detection of object boundaries in an image or video. The flow chart in **Figure 1** below shows the development process of video or image analysing program and this program will follow closely to this development flow chart.

Corresponding author: Sutthiphong Spot Srigrarom (e-mail: spot.srigrarom@glasgow.ac.uk).

This paper was submitted on July 13, 2013; revised on December 3, 2013; and accepted on December 4, 2013.

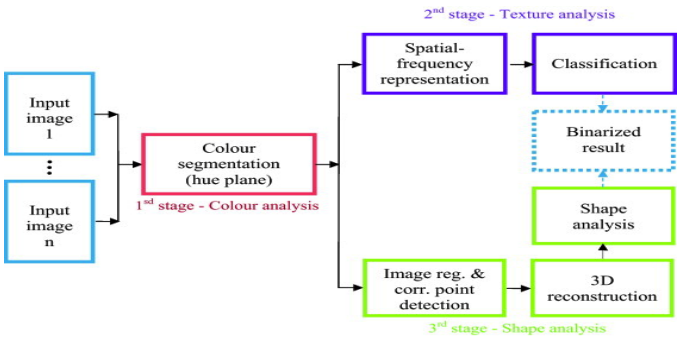

Figure 1 Image analysis flowchart

III. Development Goals

To develop an object detection program for a UAV or micro-aerial vehicle we need to review the purpose and function of the program. These are our requirements:

- Grab and capture images and video from an image-recording device
- Co-relate frames of images captured (histogram)
- Detect object boundaries, surfaces and textures.

IV. Object Detection Program Overview

This section describes the design overview of the object detection program. The object detection program will come under the Vision Controlled Motion system or (VCM) in short. VCM is used in robotics and in our case an unmanned aerial vehicle (UAV) or micro-aerial vehicle. VCM as a system can be shown graphically in the figure below.

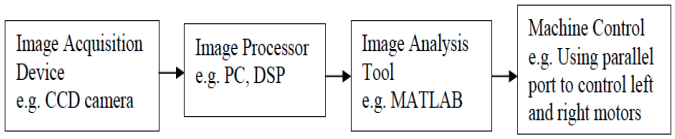

Figure 2 Vision Controlled Motion System

VCM is a very important concept and our object detection program is a part of this system. The object detection program is the image analysis portion of the VCM system shown in **Figure 2**. The program analyses images and videos acquired from an image-capturing device. This then enables the UAV the ability to sense its surrounding. User-defined inputs will then allow the UAV to avoid obstacles and identify key surfaces and track user-defined object autonomously. In an indoor environment, ability to sense the surrounding is the key to autonomous flight.

The primary design objective of the program will be to read and acquire images and videos from the image-capturing device attached to the microprocessor. The program shall then need to filter and convert these images and videos to usable data for analysis. Once these data has been converted, analysis operators can be applied to these images and videos and the results can be displayed. A detailed workflow of the program can be found in the figure below.

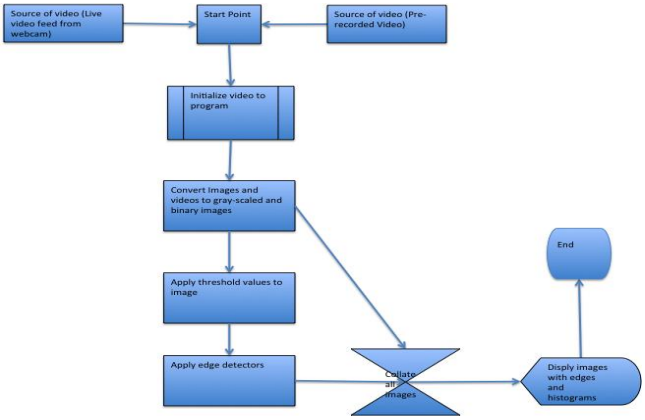

Figure 3 Object detection program workflow

V. Software Design Procedures

The object detection program will consist of critical processes and procedures. The table below will show the steps involved in creating the program. Program steps are as follows:

1. Initialize video data to program
2. Convert video and image data for processing
3. Apply edge detectors
4. Display output data

As described in the above table, there are 4 critical steps in our program. The initial step will be to initialize an image or video that was captured using a digital camera and then uploaded to the computer. Once initialized, conversions to binary and grey scale images will be carried out and an edge detector will be applied to the converted images. Lastly, the images will be displayed with edge detectors. Detailed procedures of each process will be described in the following sections. For this work, we will be using the Simulink® function of MATLAB software.

1. Initialization of image

The initial step for our object detection program is to sample images captured by the webcam. This step involves initializing the image or video to the MATLAB software. By doing so, we can then read pre-recorded videos or image. Below shows an image being imported into MATLAB.

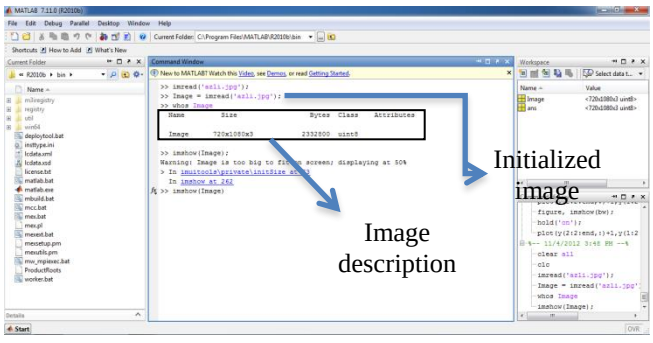

Figure 4 Image import to MATLAB

In the figure above, an image was imported into MATLAB and we can see the description of the image. The description of the image includes size, bits and class of the image. Once the image has been imported, we can then show a preview of the

image in MATLAB. **Figure 5** below shows a preview of the imported image.

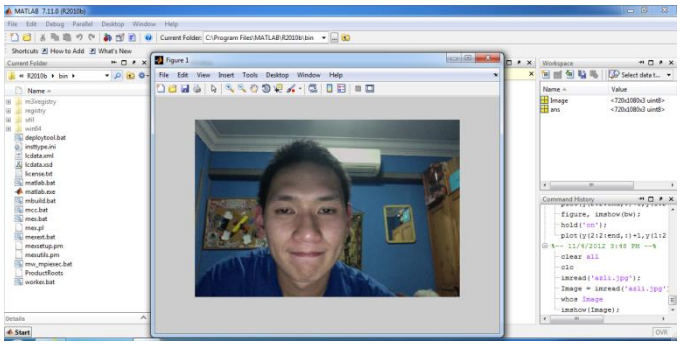

Figure 5 Displaying of image in MATLAB

2. Image Conversion

Once we have imported the image, we can then convert the image to binary images and grey-scaled images. This process is a vital step in image analysis. **Figure 6** below shows the above image in **Figure 5** converted to a binary image. Converting the images to binary images, we have changed the values of the each pixel and given them new values of (0 and 1). The black areas in the picture represent pixels with values 0 while the white areas in the picture represent pixels with values 1.

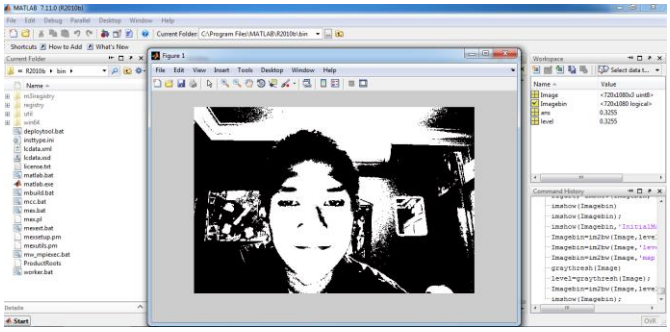

Figure 6 Binary image conversion

Next we can then display the histogram of the binary image. The figure below shows the colour histogram of the binary image above. The histogram below shows the number of pixels that have (0 or 1) values.

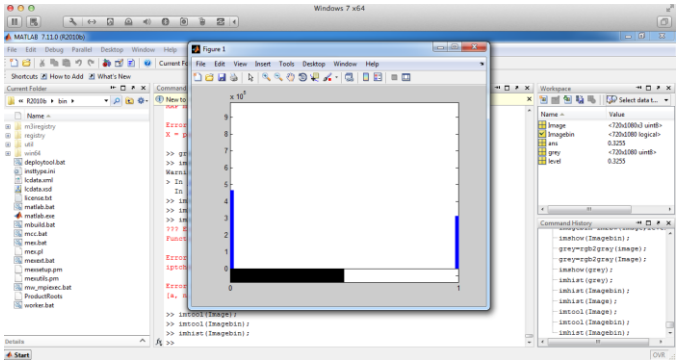

Figure 7 Binary histogram

The binary images have pixel values of either 0 or 1. This can then be seen in the colour histogram in **Figure 7**. The next step of conversion is to create a grey scale image. A grey scale image, replaces the colour pixels of the image with grey shades from white to black. Once the image has been converted, we will take the grey-level histogram of the image as shown in **Figure 8**.

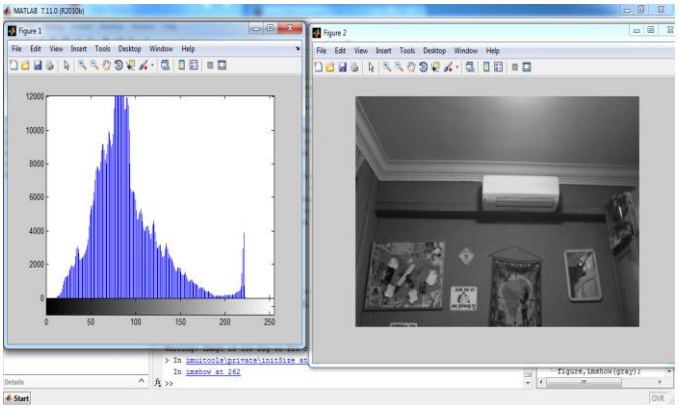

Figure 8 Grey scaled image and histogram

3. Edge Detectors

Applying edge detectors is the next important step of our object detection program. The following figure show the above image being edge segmented. Edge segmentation creates boundaries in objects of the image. This is an important component of our program as object boundaries and surfaces are important for a micro-aerial vehicle.

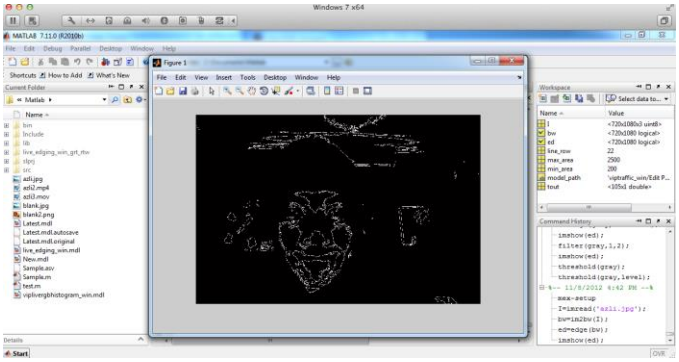

Figure 9 Edge detector

By applying an edge detector to the image, we can see object boundaries in the image. The white lines in **Figure 9** show the boundaries of the image. The above figure was applying an edge detector in this case a "sobel" edge detector to a binary image. Edge detectors can only be applied to a grey scale or binary image. Therefore the conversion step is very important.

4. Display of Data

The final step for the object detection program is to display the output image as useable data for the UAV. The image can then be written as a binary file or written as a set of codes for the microprocessor. With this data, the microprocessor can then compute these data and instruct the UAV to fly towards or avoid structures in the environment. However, for this work we will display the output of the image using the video display screen of the software. We also use statistical methods such as histograms to view the data.

The final design Simulink Code is shown in **Figure 10** below.

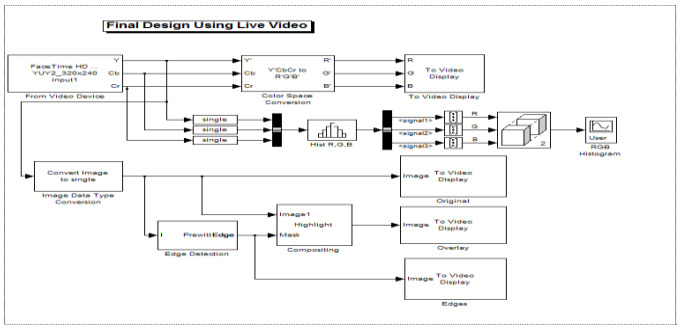

Figure 10 Final design using live video in Simulink

VI. TESTING AND EVALUATION

1. Object Detection Program Test (Using Pre-recorded Video)

The first initial program design takes on the first approach in design flowchart of the overall object detection program. This approach uses a pre-recorded video that has been recorded by a digital camera and then uploaded to the computer. A video of an indoor environment was taken using a digital camera. To simulate structures such as pillars and ceiling, we used a video of a typical corridor. Some images of this corridor can be seen in the figure below.

Figure 11 Corridor video snapshot

The pre-recorded video will then be initialised to the program and converted to useable images for analysis. Once this is done, an edge detector will be applied to the image. For this design, a "Prewitt" edge detector will be used. It calculates the gradient of the image intensity at each point giving the possible increase from light to dark and the rate of change in each direction. The output image of this video will be displayed using video displays and a colour histogram. The figure below shows a snapshot of this program design and output.

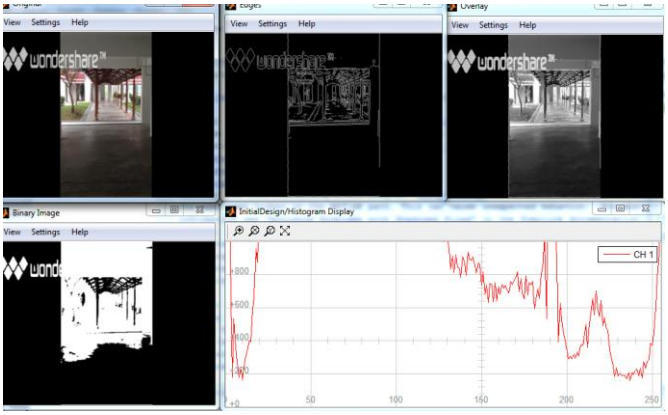

Figure 12 Result display (pre-recorded video)

2. Object Detection Program Test (Using live acquisition Video)

The second design of the object detection program uses life feed video instead of pre-recorded video. This approach uses a standard webcam that is connected to the computer via USB and processes and converts live images instantly. As mentioned in the flowchart of the design overview, this is the second method of image acquisition. This process takes place is in real time and any objects in the video are instantaneously analysed and an output image is displayed. Real time processing analyses the image at the same speed as the images are occurring. In the context of a UAV or micro-aerial vehicle, real time processing is required to process images captured by the UAVs camera system. With real time processing, the on board microprocessor is able to analyse data from the object detection program while the UAV is in flight. This will then enable UAV autonomous functionality. Thus, this will be the final design for the object detection program. The first step to this design is to initialize the webcam to the program. Once that is done, the webcam will grab frames of images at 30 frames per second. These images will then be converted and an edge detector will be applied to these images. The output data will be displayed as images and a colour histogram. A "Prewitt" operator edge detector mentioned in the first design will also be applied here. The figure below shows a snapshot of this design output.

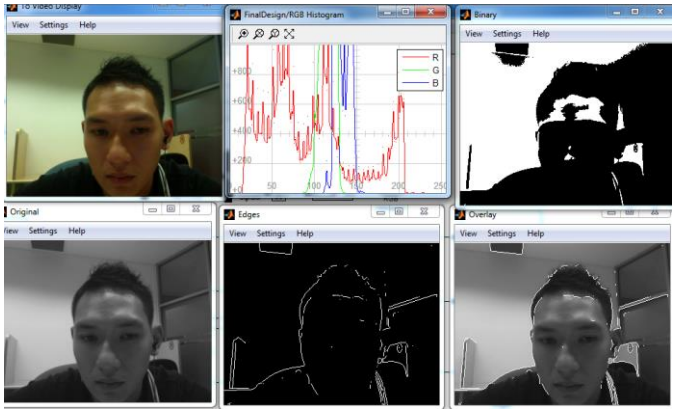

Figure 13 Live feed design output

From the output above, real time images were processed and an output was displayed in the form of video displays. These displays include the original image, binary image, grey-scaled image, edged image and lastly an overlay of the edges on the grey-scaled image. A colour histogram of the original image is also shown.

3. *Field Test of Final design*

To verify the final design of the object detection program, a field test was done. Using different objects, the output image of the program was recorded and analysed to see the robustness of the edge detector of the program. To test the design even further, a test was carried out in different lighting conditions. Different lighting condition is an important factor for indoor environment.

3.1. *Object Test*

During this test, different object were used to simulate common object found in an indoor environment. First object used was a common drinking bottle found in an indoor environment. The figure shows the output of a bottle after being processed by the object detection program. A histogram of the colours of the video is also displayed.

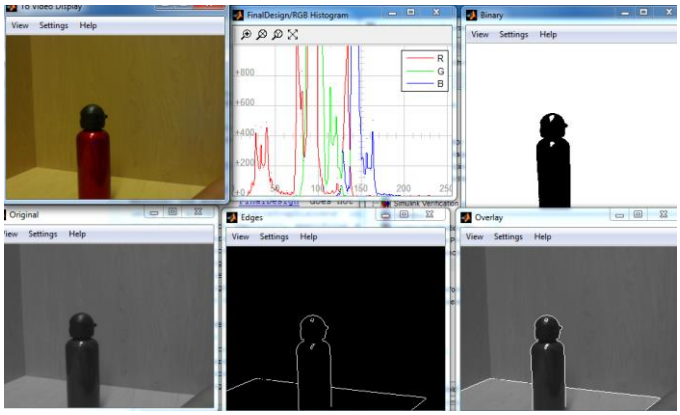

Figure 14 Bottle test

Next, tables and chairs were used to simulate an indoor environment. As the previous test, tables and chairs were placed in front of the camera and the output data was being displayed in the video display. The figure below shows the output data of the chair and table after processing.

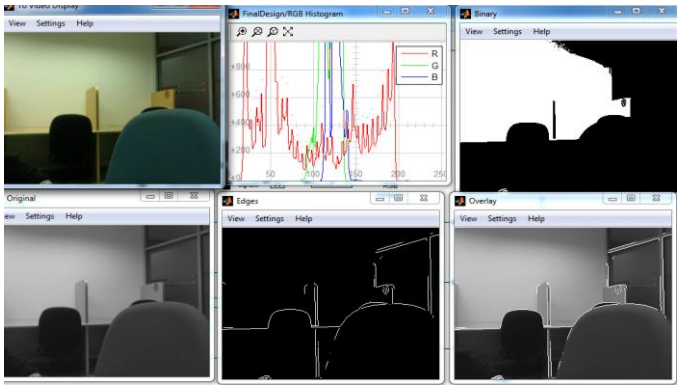

Figure 15 Indoor table and chairs

3.2. *Object Test Evaluation*

The object test satisfies our requirement of an object detection program to detect object boundaries and edges. From test carried out, we can see in **Figure 14** of the bottle test that the full shape of the bottle was being detected and its boundaries outlined. The colour histogram also reflects an increase in R channel, shows that the bottle is being detected as it is red in colour. Using the colour histogram, we can gauge distance of the object from camera. This means that if the amplitude of the colours increase and the graph of colours widen the object are near the camera. This test appears very satisfactory. From **Figure 15**, it is evident that the surfaces and boundaries of the table and chairs were detected and outlined in the output display. In the binary image display, it is observed that the walls of the indoor environment were given a value of 1 and it displays white while the rest of the environment was given a value of 0 and displayed as black. Boundaries of the table and chair on the other hand were detected and displayed in the edges display. This showed that the program performed its desired function. However, the program did not detect the shadow on wall.

3.3. *Lighting Test*

Lighting test is the next step for verifying the functionality of the object detection program. For this test, we simulated an indoor environment with two distinct lighting conditions, low lighting and bright lighting. This test is to simulate typically an office environment whereby the UAV or micro-aerial vehicle maybe placed to do surveillance and monitoring. The figure below depicts the output data of program in a normal office environment.

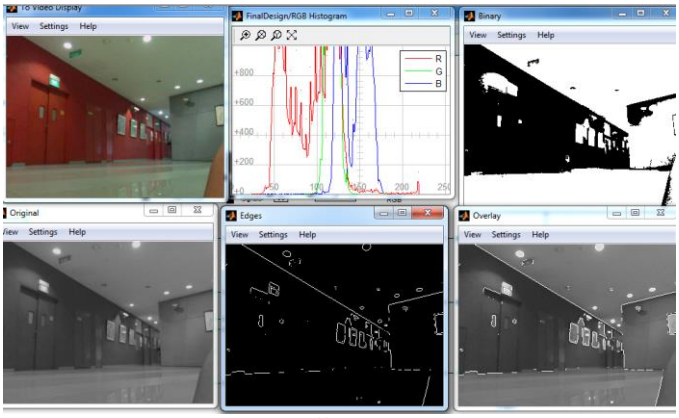

Figure 16 Office environment

The last and final test carried out was a high light environment. This test simulates an outdoor environment. This test is simulated to captured output data of the object detection program if it is coupled with a UAV or micro-aerial vehicle that has outdoor autonomous functionality. Outdoor functionality usually uses GPS triangulation; however, if the program is able to detect surfaces in the environment, the program can be used as an object avoidance system for the UAV or micro-aerial vehicle. The figure below shows the output data of the object detection program in an outdoor environment.

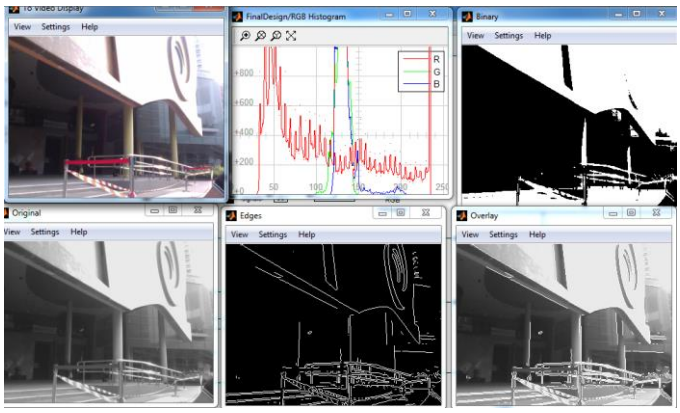

Figure 17 Outdoor environment 1

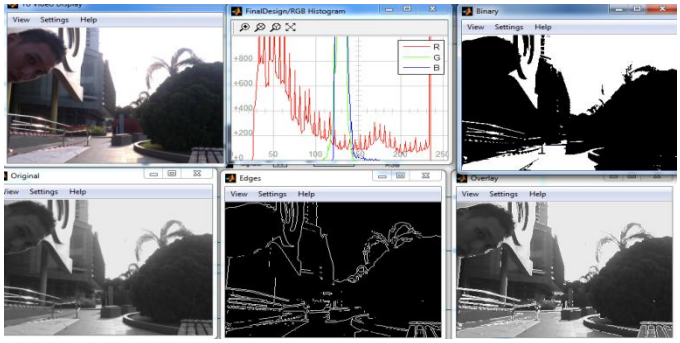

Figure 18 Outdoor environment 2

3.4. Lighting Test Evaluation

From the lighting test, the object detection program fulfils the requirements of our work. The object detection program is able to detect surfaces and boundaries of objects in the image. The downside of the program however, is that the program is unable to detect fully boundaries of objects that have low illumination levels. From the output display in **Figure 16** above, it is observed that key structures of the environment were detected. Wall boundaries and picture frames on the walls were detected. However, the boundary of the door and the wall was not detected due to shadow being casted on the door. This shows that in low lighting environments, the object detection is not robust enough to detect object boundaries. Low light objects were not detected and edges were not shown in the output display. From the results in **Figure 17** and **Figure 18** of the outdoor environment test, surfaces and object boundaries were detected. However, as in the case of the indoor environment, object that have low light levels were not detected. The outdoor environment results show that this program can be used in an outdoor application as an object avoidance system.

VII. CONCLUSION

After every tests and evaluations that were carried out, it has been determined that our in-house Simulink-based object detection program is able to detect object boundaries in the video that was pre-recorded or live video captured by a webcam. The initial design of the program was able to detect object boundaries from a pre-recorded video taken from a digital camera. The second and final design of the program was able to detect object boundaries in real time via a USB webcam attached to the computer. Therefore in summary, this work is a success whereby; both designs of program are able to detect object boundaries.

Overall, our in-house object detection program satisfies our requirements. The object detection program is able to initialize images and videos either a pre-recorded video or real time video. The object detection program is then able to convert these images and video into grey-scaled images and binary images. Finally the object detection program is able to detect object surfaces and boundaries in the image and video by image analyzing techniques such as segmentation and applying an edge detector. Our future work will be the comparison of the proposed technologies with the already-existing functions, such as by using pseudo-code mechanisms, as well as the conversion from Simulink-based program to executable program to be embedded in the MAV.

REFERENCES

[1] Burger, W., Burge, M.J., "Digital Image Processing: An Algorithmic Approach Using Java" Springer, 2007, ISBN 1-84628-379-5

[2] Cesetti, A., Frontoni, E., Mancini A., Zingaretti P., Longhi S., "A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks", *J Intell Robot Syst* , 2010, Vol.57, pp.233–257, DOI 10.1007/s10846-009-9373-3. CrossRef

[3] Fisher, R., Dawson-Howe, K., Fitzgibbon, A., Robertson, C., Trucco, E., "Dictionary of Computer Vision and Image Processing", John Wiley, 2005, ISBN 0-470-01526-8

[4] Gonzalez, R.C., Woods, R.E and Eddins, S.L., "Digital Image Processing using MATLAB", Pearson Education, 2004, ISBN 978-81-7758-898-9

[5] Jähne, B., "Digital Image Processing", Springer, 2002, ISBN 3-540-67754-2

[6] Ma, W.Y., Manjunath, B.S., "Edge Flow: A Technique for Boundary Detection and Image Segmentation", *IEEE Transactions on Image Processing*, Vol.9, No.8, pp.1375-1388, 2000, CrossRef

[7] Mathworks, "Computer Vision System Toolbox", www.Mathworks.com)

[8] Sharon, E., Brandt A., Basri, R., "Segmentation and Boundary Detection Using Multi scale Intensity Measurements", *Proceeding of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001, pages 469-476, CrossRef

[9] Solomon, C.J., Breckon, T.P. ,"Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab", Wiley-Blackwell, 2010. CrossRef

[10] Sonka, M., Hlavac, V., Boyle, R., "Image Processing, Analysis, and Machine Vision", PWS Publishing, 1999, ISBN 0-534-95393-X

[11] Young, I.T., Gerbrands, J.J., van Vliet, L.J., "Fundamentals of Image Processing", TU Delft, ISBN 90–75691–01–7, 1998