Development of Human Computer Interface for Autopilot Control of Unmanned Aerial Vehicle

May San Hlaing*, Zaw Min Naing, Maung Maung Latt***, Win Khine Moe**, Hla Myo Tun***

* Department of Electronic Engineering, Mandalay Technological University
** Department of Research and Innovation, Ministry of Education
*** Technological University (Toungoo)

Abstract- As a result of the limitations of human cognitive skills, judgment, decision-making, and tactical understanding in the use of Unmanned Aerial Vehicles (UAV), there is a need to redesign the current human-computer interface (HCI) for Autopilot control to improve the interaction and communication links between operators and the UAVs. This system displays the information to increase situational awareness; the operator will see everything, interpret it, make appropriate decisions, and have the ability to implement the decision. Multiple interfaces are developed using C's Graphical User Interface (GUI) capabilities as a simulation environment. Both alternatives will combine buttons and place them in sequential order according to the steps needed to initialize UAVs and flight paths. Usability tests with participants are conducted to measure their performance based on previously determined metrics: the time it takes to train a participant on how to use the interface, the time to complete a task, the number of errors that occur during the task, and the satisfaction level. Based on the simulation results alternatives are redesigned and retested in order to achieve improved performance. An interface that reduces the cognitive workload on an operator to allow better situational awareness of the environment is determined.

Index Terms- HCI, UAV, Autopilot System, Control Design, MATLAB

I. Introduction

Unmaned Aerial Vehicles (UAVs) are currently being used extensively for all types of military missions including, reconnaissance, standoff laser designation, autonomous scouting and relay, and crowd control or riot dispersion. These applications include reconnaissance, surveillance, search and rescue, remote sensing (nuclear, biological, chemical), traffic monitoring, natural disaster damage assessment, etc. UAV can accomplish its mission without risking the pilot/operator and usually with lower operational cost compared to manned aircraft. From science and academic point of view, UAV research and development has the potentials of integrating several science and engineering disciplines such as Embedded System, Real Time System, Sensor Fusion and Integration, Power System (for on board power generator), Image Processing (for obstacle avoidance), Information System (for ground control station), Telecommunication (for telemetry), Networking (for information dissemination to remote audience from ground control station), Control System (for automatic airframe stabilization and waypoint navigation), Multi Agent System (for multiple UAVs coordination), etc.

Operators use Human-computer interfaces (HCI) to monitor UAVs throughout their missions. The situational awareness (SA) of the operator is important. The first level is the perception of information. Next, SA deals with the comprehension of meaning. Finally the last level is the projection of the situation over time.

There is an interest to create teams consisting of troops combined with unmanned vehicles, but the current HCI for control of multiple UAVs falls short due to the poor organization of information and decision making capabilities incorporated into UAVs. A new interface is needed to increase situational awareness so that a single operator can control multiple UAVs more proficiently. The current issue with the HCI for UAVs is that too much information is being displayed through the interface to the controller when more than one UAV is being monitored at once. This creates an information overload on operator and reduces the situational awareness of the operator.

It is imperative that the design improves the current interface by reducing cognitive workload. There is a need to increase situational awareness and accuracy of the operator. This will lead to an overall better performance in controlling multiple unmanned aerial vehicles. Throughout the scope of the work, the design of the HCI is considered. The execution of a reconnaissance mission and the human performance of the HCI are mainly focused. The human performance is measured by training time, time to complete tasks, number of errors, and satisfaction level of the operator. Maintainability, functionality, and reliability will not be measured due to time constraints.

II. Development of a User Interface Design

User Interfaces are not stand-alone applications. Rather they are essential elements of most software packages that allow users to interact with complex programs. Nonetheless, it is not enough for UIs to be visually appealing; they must also be user friendly. UIs need to be consistent and fully functional. Based on three fundamental principles of UI design, the design constraints of the proposed UI to the UAV coordination problem were carefully identified and determined. Table 1 indicates all design constraints. Among the many UI options, graphical user interfaces offer a graphical method of solving many problems. In addition, GUIs provide the functionality to vary problem parameters and present the results graphically. For these reasons, GUIs have become popular and are utilized by the majority of computer users.

Table .1. UI Design Criteria

Users must be able to
- identify relationship between ETA and each vehicle's cost
- manipulate ETA
- choose any ETA or path available
- make a visual confirmation of any decision made before a final decision

III. Developing GUIs Using C Language and MATLAB

The main objective behind developing the proposed GUI is to obtain easy, yet intuitive control of a group of mobile robots that may have different properties. In addition, these robots are restricted by timing constraints such as simultaneous and sequential arrivals. The GUI is to be designed to assist users to determine final paths and velocities of each mobile robot in variable scenarios by providing calculated cost data for the vehicles in a graphical display.

Previous cooperative timing problems (i.e., the multi-agent rendezvous problem [11]) were developed and solved using MATLAB and it was necessary to maintain interoperability with this software. MATLAB allows users to develop user friendly GUIs that are implemented using the UI controls [49, 50, 51, 52]. Because of these advantages, MATLAB was chosen as a GUI design tool for this research. The MATLAB software package provides easy-to-use editing, debugging, and powerful graphic functions. In addition, MATLAB provides its own basic UI, including a GUI building interface called GUIDE, which allows users to select and place custom UI features (i.e., graph, buttons, and text) into the main GUI frame and connect them to the GUI control programs, known as script files or m-files. All MATLAB graphics are object oriented and each of these objects has its own identifier or handle. By placing and modifying these handles on a GUI, the proposed GUI was created and revised.

IV. Proposed Graphical User Interface

The results provide insights into the types of interfaces most useful for multi-vehicle control by a single user. Designing the proposed graphical user interface (GUI) is based on the results acquired. The results suggest that using the cost control and the coordination variable and function control interface can reduce the amount of workload. The spatial control interface has known disadvantages such as the highest workload and the worst quality of decision though it has an advantage in the response time with the constrained path cases. In addition, the coordination variable/function control interface has the same feature set used in the temporal control interface. Hence, the temporal control and spatial control interfaces are excluded in the proposed GUI.

The proposed GUI consists of four distinct interfaces: coordination variable/function interface, spatial map display, cost interface, and variable option selectors. Each interface presents different information that helps users to determine a final timing decision. In addition to these four zones, a cluster of executable buttons below the cost interface allows a user to operate the GUI.

V. Simulations

To demonstrate a cooperative timing problem in an urban environment, several different simulation scenarios are investigated and a few are chosen for testing. Among these chosen scenarios, some assumptions are made. The list below describes all assumptions made for chosen simulation scenarios.

- All vehicles must avoid obstacles while maneuvering.
- There is one hazardous area (e.g., sniping zone, toxic waste spilled) in each scenario.
- If a vehicle enters a hazardous area, the vehicle must change its current velocity to the maximum velocity for the rest of its path.
- All vehicle positions and velocities are locally controlled.
- Collisions among vehicles are ignored.
- Vehicle paths must be contained within the map

VI. Interfacing for the Parallel Port Communication

The parallel port is used to communicate, control and collect data with operation mode linking with remote terminal unit. It uses library files ".dll" to control Visual C for input/output from actual process. Windows have other options for driving devices, including DLLs. A visual C program can call a DLL directly to access a DLL. A DLL (dynamic linked library) is a set of procedures that windows applications can call. When an application runs, it links to the DLLs declared in its program code, and the corresponding DLLs load into memory. Multiple applications can access the same DLL. The application calls DLL procedures much like any other subroutine or function.

Many programming languages enable to write and compile DLLs. Creating a DLL can be as simple as writing the code and choosing to compile it as a DLL rather than as an executable (.exe) file. Visual C programs can call any DLL, whether it was originally written in Basic or another language. The parallel port in the original IBM PC, and any port that emulates the original port's design, is sometimes called the SPP, for standard parallel port, even though the original port had no written standard beyond the schematic diagrams and documentation for the IBM PC. Other names used are AT-type or ISA-compatible. The port in the original PC was based on an existing centronics printers have continued.

SPPs can transfer eight bits at once to a peripheral, using a protocol similar to that used by the original centronics interface. The SPP doesn't have a byte-wide input port, but for PC-to-printer transfers, SPPs can use a Nibble mode that transfers each byte 4 bits at a time. Nibble mode is slow, but has become popular as a way to use the parallel port for input. There are many ways to access a parallel port in software, but all ultimately read or write to the port's registers. To distinguish between I/O ports and system memory, the microprocessor uses different instructions and control signals for each. On the original PC, port address could range from 0FFh to 3FFh. Many newer parallel port decode an eleventh address line to extend the range to 7FFh. Many programs that access the parallel port use this table to get a port's address. This way uses only have to select LPT1, LPT2, or LPT3, and the program can find the address. Use the control bits as inputs on the PC only on SPPs or ports that emulate the SPP. If it uses the control lines as inputs, drive them with opencollector outputs. This will protect the port's circuits if a law control-port output should connect to a high output.

Visual C has been the most popular choice for basic programmers developing window programs. It can add Inp and Out to the language in a dynamic linked library (DLL). A DLL contains code that any windows program can access, including the program written in Visual C. This DCS based software includes DLLs for port access: inpout32.dll, for use with 32-bits Visual C program.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;

public class PortAccess
  {
     [DllImport("inpout32.dll", EntryPoint="Inp32")]
     Public static extern int Inp32(int PortAddress);
     [DllImport("inpout32.dll", EntryPoint="Out32")]
    public static extern int Out32(int PortAddress, int Value);

public static int Outs;
    public static int outselect;
    public static int outStart;
    public static bool inUAV1;
    public static bool inUAV2;
    public static bool inUAV3;
    public static bool inError;
}
```

The program declares the Inp32 and Out32 contained in input32.dll and declares variables using software for HCI for control of UAV. A program that writes directly to parallel port has no way of knowing whether another application is already using the port. But sometimes a port is intended to use with a single application. If other applications have no reason to access the port, direct I/O with Inp and Out should cause no problems.

When receiving data from Operation mode two timers are used to different the condition of sensors and motors using HCI for control of UAV. With Inp and Out declared in program, it can use them much like Inp and Out in QuickBasic. On the user's system, the file Inpout32.dll should be copied to one of these locations: the default windows directory (usually/Windows), the default system directory (usually/Windows/System), or the application's working directory. These are the locations that windows automatically search when it loads a DLL. If for some reason the DLL is in a different directory, it will need to add its path to the filename in the declare statements.

VII. System Implementation

The operation of the process can be run and stopped from main page by the manager level. The page serves to collect data from the operation mode and display in real-time running and save the collected data. The graph line is drawn in approximately one second step until the STOP command is accepted. When the main page program is shutdown the data acquisition and processing is shutdown all process except receiving command from the main page. So, the application software can be viewed in pages, the main page of PC and other pages linking each other as software of HCI system.

The function of this module is to send the control command for operation mode and collect data to this module. The complete block diagram is shown in Fig.1. This page can be divided into the following function-

- Operation mode (real-time simulating as hardware),
- Hardware mode (displaying components like real devices),
- Data mode (List of facts of HCI based system), and Control mode (including Start, Stop and Exit for security).

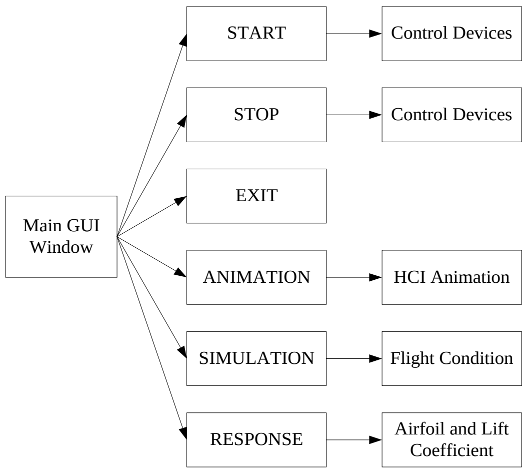


Fig.1. Block Diagram of Linking Window System

VIII. Design Program for Main Window

The main page uses the Visual C portion of window application from Visual Studio 2008 IDE. It can be designed a window form by using the software application data and used data by depending on the process from the properties of items. The control devices are designed on the form by using the toolbox. And the codes of programming for each of the component are written on the code viewer by using Visual C.

Timer tools are used for timing of data interfacing and input/output for process data. A timer is used to raise an event at user-defined intervals. Gets or sets the time, in millisecond, between timer ticks. Window forms timer component has an interval property that specifies the number of milliseconds that pass between one timer event and the next. Unless the component is disabled, a component is designed for a windows forms environment. The read or write signal of timer continues to receive the Tick event at roughly equal interval of time. This input/output system activates every time. It uses timer tool (timerIn_Tick) to watch the port system. The signal from timer state is flowed to selected pin of PC.

The error signal from the process activates every time. So it uses timer tool (timer_Error_Tick) to watch this condition. If error signal from input/output system is true, it plays the audio file in my computer and the back color button is red. If it is false, the back color of error button is blue. This system is always turn on. It is shown in Fig.2.

The simulation button (cmdSimulation) for linking the real-time simulation process is used to open the simulation dialog box (SimulationForm). It can be written:

SimulationForm SF=new SimulationForm();

SF.Showdialog();
And the animation simulation (cmdHardware) is utilized to open the hardware dialog box (HardwareForm). It can be written:
HardwareForm HF=new HardwareForm();
HF.ShowDialog();
The data box (cmdData) for user guide for present system is used to open the data box (DataForm). It can be written:
DataForm DF= new DataForm();
DF.ShowDialog();
The exit button (cmdExit) for closing the system is used. It includes the message box depended the user decision.

```
if (MessageBox.Show("DO YOU WANT TO EXIT?", "HCI System",
MessageBoxButtons.YesNo,MessageBoxIcon.Exclamation) == System.
Windows.Forms.dialogResult.Yes)
{
   Application.Exit();
}
```

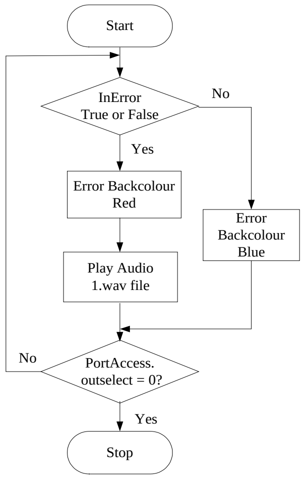


Fig.2. Flow chart of Error System

IX. Design of Animation Window

According to the HCI for control of UAV system, there are two main portions to animate the UAV control system. They are HCI portion and controlled units (Multi-UAV) system. The connection between these two portions is wireless access point.

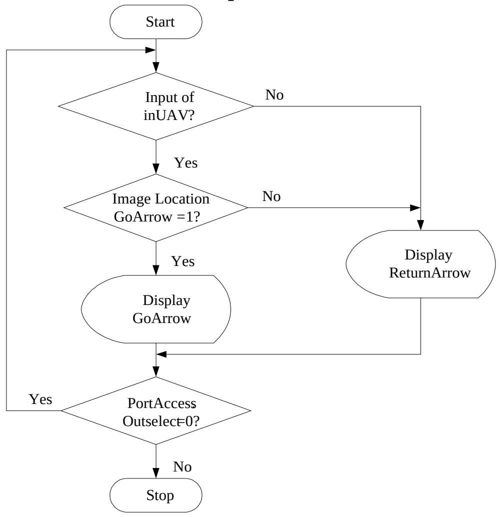


Fig.3. Flowchart of Animation Window

X. Design of Flight Condition

The flight condition can be created based on the on-off condition of conventional switch operation. The program is debugged on the C platform and the imagelist_UAV1 to 3 are alternately on and off condition based on the timer commands from the main window of GUI.

XI. Design of Response Window

The airfoil response and lift coefficient due to the actual flight path condition are evaluated by using airfoil equation and others.

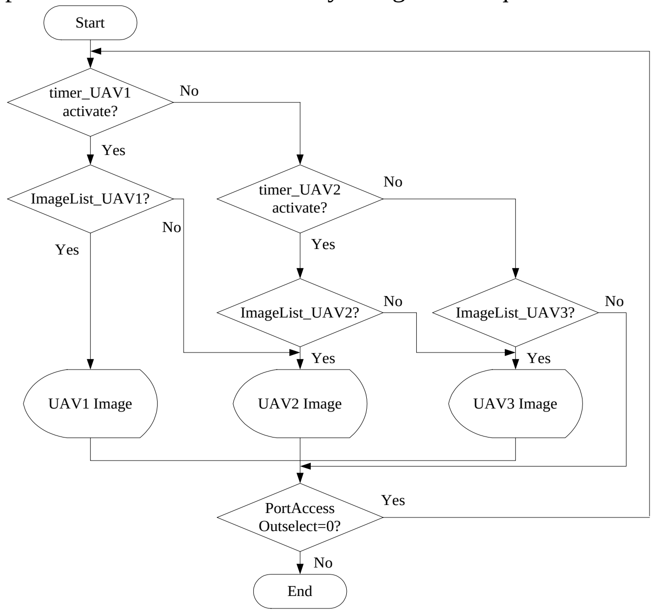


Fig.4. Flowchart of Response Window

XII. Simulation Results for HCI

In this system, HCI system based monitoring programs are written by Visual Studio C programming. In this chapter, the display result of HCI for control of UAV systems is described. The result contains the main window that links other window application and simulation page for the real-time operation. This display system only intends to monitor and control for this UAV control system. For the overall system, the hardware devices for control unit are connected to the computer including window application software using the connectors of parallel port.
In this system, parallel port pin are used to control the input and output signal of the hardware devices. And it also links to the window application with the C programming. It includes main page which links other pages, operation page which links the device operations, response page which expresses the condition of airfoil and lift coefficient, animation page that links the monitoring of human with computer and user guide page which expresses as the user guide.

XIII. Simulation Result of Main Page

In this page, it includes seven buttons – start, stop, simulation, user guide, animation, response and exit. And it used two label name including title and designer name, and alarm color system using the condition of error signal. This page is essential for the overall system. It includes the interfacing system for the I/O signal from parallel port and links with other pages and sends to the other pages the condition of I/O signal.

The start and stop buttons are used to run and stop the overall system which is real-time monitoring process. The simulation button is used to display the simulation result of the process. The user guide button is used to express the user guide for the system. The animation button is utilized to show the monitoring of hardware devices by linking with it. The exit button is used to end the process and to exit the monitoring system by linking with the yes or no message box. It is shown in Fig.5.

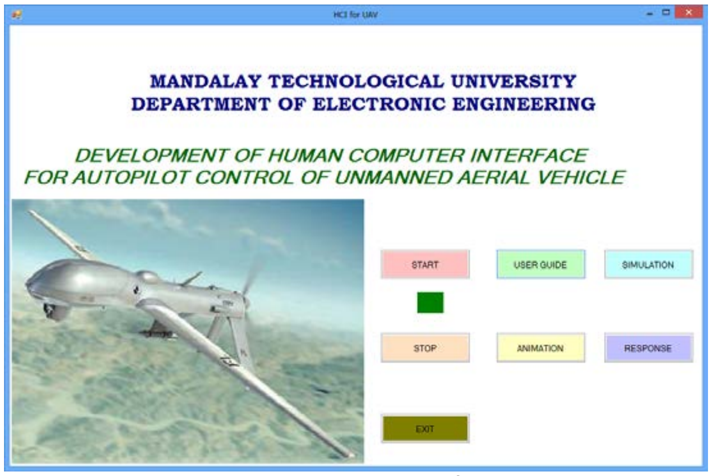

HCI for UAV
MANDALAY TECHNOLOGICAL UNIVERSITY
DEPARTMENT OF ELECTRONIC ENGINEERING
DEVELOPMENT OF HUMAN COMPUTER INTERFACE
FOR AUTOPILOT CONTROL OF UNMANNED AERIAL VEHICLE
START
USER GUIDE
SIMULATION
STOP
ANIMATION
RESPONSE
EXIT

Fig.5. GUI Main Window for HCI

XIV. Simulation Result of the Animation Window

This page is a main process to show for all operation. On the form, it is designed with the sample components for the real devices of UAV control system by using the sample figures of actual devices. It uses I/O signal from main page for operation of device by using each signal. It is shown in Fig.6. The back button is used to go back the main page like to close the operation page.

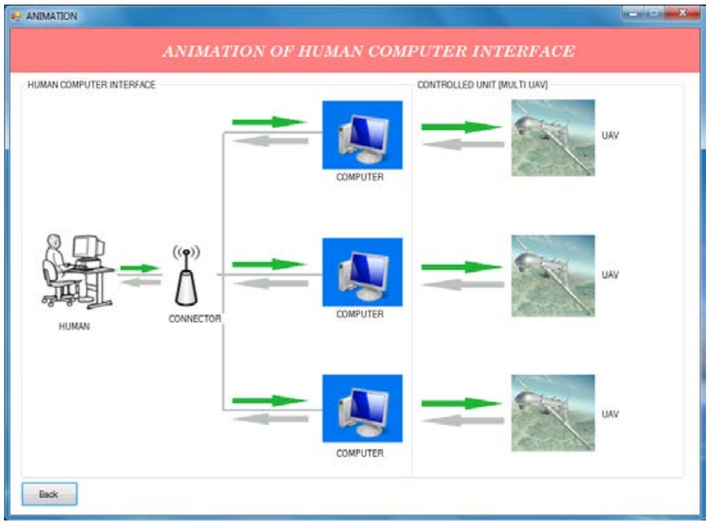
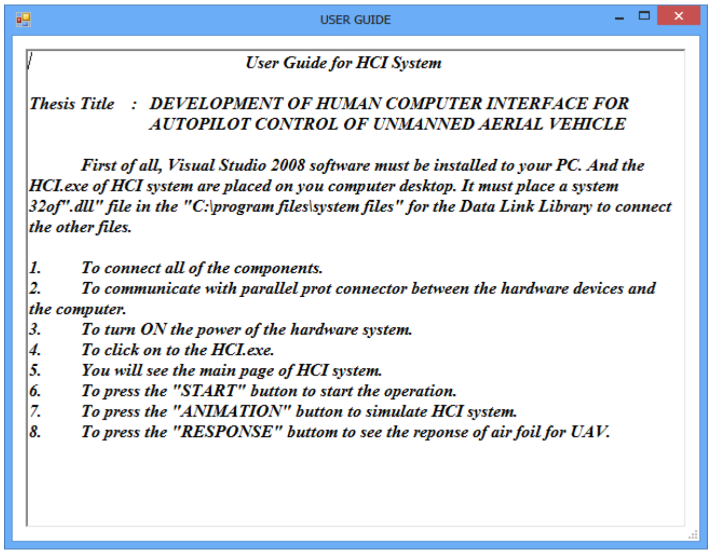


Fig.6. Animation Window of HCI for Control of Multi UAV System

XV. Simulation Result of the User Guide Window

This window shows the user guide for the overall system how to use and what contain it. It is to link with the main page. It is shown in Fig.7. If the user wants to go back the main page, the data window will be closed.



USER GUIDE

User Guide for HCI System

Thesis Title : DEVELOPMENT OF HUMAN COMPUTER INTERFACE FOR AUTOPILOT CONTROL OF UNMANNED AERIAL VEHICLE

First of all, Visual Studio 2008 software must be installed to your PC. And the HCI.exe of HCI system are placed on you computer desktop. It must place a system 32of".dll" file in the "C:\program files\system files" for the Data Link Library to connect the other files.

1. To connect all of the components.
2. To communicate with parallel prot connector between the hardware devices and the computer.
3. To turn ON the power of the hardware system.
4. To click on to the HCI.exe.
5. You will see the main page of HCI system.
6. To press the "START" button to start the operation.
7. To press the "ANIMATION" button to simulate HCI system.
8. To press the "RESPONSE" buttom to see the reponse of air foil for UAV.

Fig.7. User Guide Window

XVI. Simulation of the Flight Condition

This window shows the hardware images like the components of HCI for control of UAV system. If the user pushes the start button, the overall system will run and the UAV will show on the form. It links with main page. The back button is used to go back the main page that shown in Fig.8.

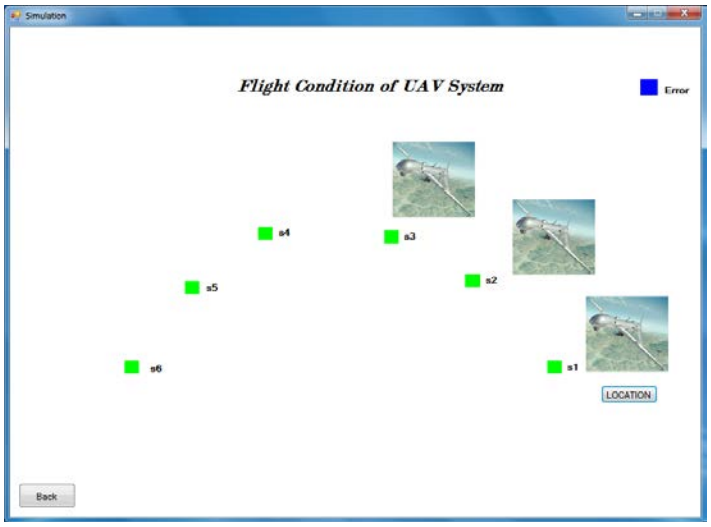
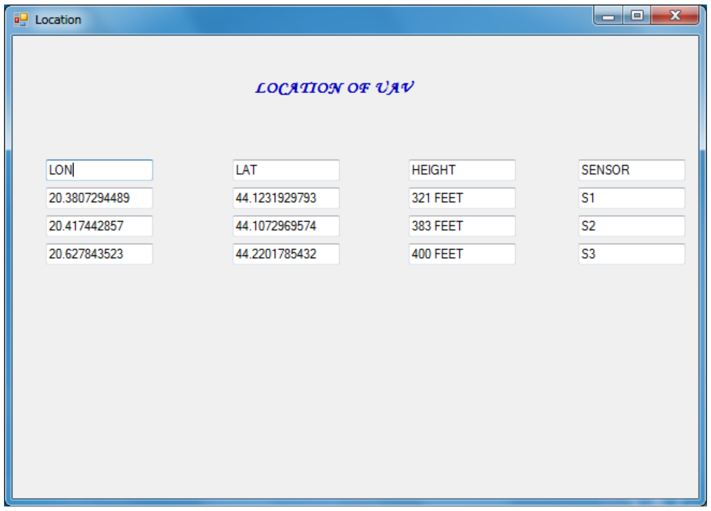


Fig.8. Flight Condition Window for UAV System

The UAV1 image is first displayed according to the timer_UAV1 activation and then the others are activated based on the situation of each timer. The result only shows the flight condition of UAV with GUI application.



LOCATION OF UAV

LON	LAT	HEIGHT	SENSOR
20.3807294489	44.1231929793	321 FEET	S1
20.417442857	44.1072969574	383 FEET	S2
20.627843523	44.2201785432	400 FEET	S3

Fig.9. Location Window for UAV System

The location window for UAV system is illustrated in Fig.9. There are three data for UAV location information to control the UAV. They are longitude, latitude and height or altitudes which were depend on the sensor operations to control the stability via human operator. The values of longitude, latitude and altitude are typical range for proposed UAV model.

XVII. Simulation of the Response Window

This window shows the two response curves for airfoil and lift coefficient for UAV flight condition which are based on HCI with GUI system. The two responses are comparing with the conventional situation and modification stage of UAV flight condition. According to these responses, the UAV can be controlled easily via HCI based GUI system for control of UAV. It is shown in Fig.10.

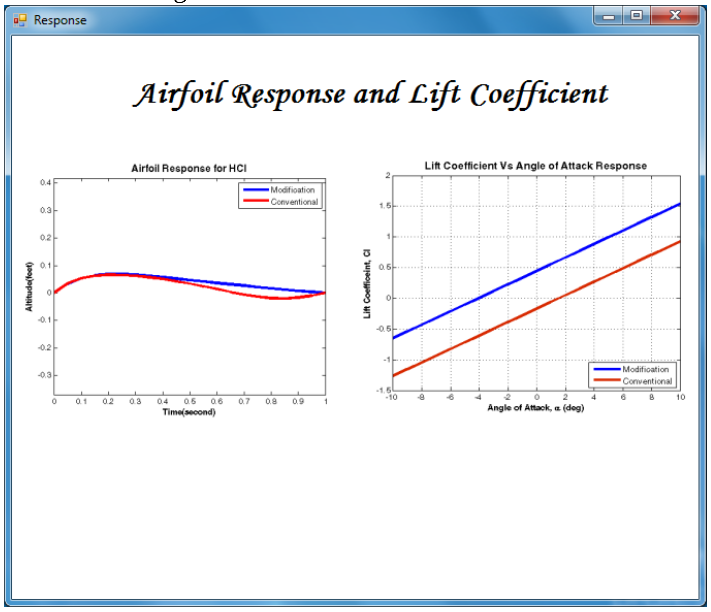
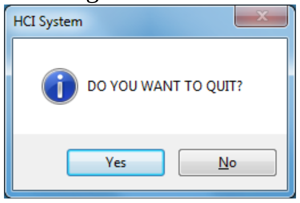


Fig.10. Airfoil Response and Lift Coefficient For UAV Flight Condition

XVIII. Simulation of the Exit Window

This window shows the decision making for the user to run or stop of the current condition of the overall system. It uses the message box including yes or no decision that is if it is yes, it will exit from the system otherwise the process will continue the operation. It is shown in Fig.11.



DO YOU WANT TO QUIT?

Fig.11. Exit Window

XIX. Conclusion

Though this research limited the number of vehicles to three, it is possible to accommodate as many vehicles as desired, since all codes created to implement algorithms used in this research including trajectory generation and coordination variables and coordination functions are modular except for the GUI. The proposed GUI was designed as a prototype, and for simplicity the display was limited to three robots. However, principles of GUI design discovered in this research are applicable to the control of a variable number of robots. Specific modular GUI designs were not studied, but changing the proposed GUI to allow the control of more than three vehicles is simple if all design parameters such as the maximum number of vehicles and the size of GUI are known a priori. As the total number of vehicles increases, methods of representing vehicle data and controls in the GUIs should be reconsidered.

References

[1] Bowman, D. & Hodges, L. (1997). An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments, 1997 ACM Symposium on Interactive 3-D Graphics, pp. 35 -38.

[2] Eberly, D. (2000). 3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics. Morgan Kaufmann, ISBN: 1558605932.

[3] Faugeras, O., Vieville, T., Theron, E., Vuillemin, J., Hotz, B., Zhang, Z., Moll, L., Bertin, P., Mathieu, H. , Fua, P., Berry, G. & Proy, C (1993). Real-time Correlation-Based Stereo: Algorithm, Implementations and Applications. INRIA Technical Report RR-2013.

[4] Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1995). Computer Graphics: Principles and Practice in C (2nd Edition). Addison-Wesley, IBSN: 0201848406.

[5] Hand, C. (1997) A Survey of 3-D Interaction Techniques. Computer Graphics Forum, 16(5), 269-281.

[6] Hearn, D., and Baker., M. (1996). Computer Graphics, C Version (2nd Edition). Prentice Hall, ISBN: 0135309247.

[7] Lindeman, R., Sibert, J., & Hahn, J. (1999) Hand-Held Windows: Towards Effective 2D Interaction in Immersive Virtual Environments. IEEE Virtual Reality.

[8] Mine, M., Brooks, F., & Sequin, C. (1997) Moving Objects in Space: Exploiting Proprioception in Virtual-Environment Interaction. Proceedings of SIGGRAPH 97.

[9] Sherman, W. & Craig, A. (2003) Understanding Virtual Reality: Interace, Application, and Design. Morgan Kaufmann, ISBN: 1-55860-353-0.

[10] Watt, A., & Watt (1993). 3D Computer Graphics, 2nd Edition. Addison-Wesley, ASIN: 0201154420.

[11] Woo, M., Neider, J., Davis, T., & Shreiner, D. (1999). OpenGL® Programming Guide: The Official Guide to Learning OpenGL, Version 1.2 (3rd Edition). Addison-Wesley, ISBN: 0201604582.