CRANFIELD UNIVERSITY

OGUZ KAGAN ISIK

MANNED MACHINE INTERFACE (MMI) FOR COMMANDING AUTONOMOUS WINGMAN

SCHOOL OF AEROSPACE,TRANSPORT AND MANUFACTURING
Autonomous Vehicle Dynamics and Control

MASTER OF SCIENCE
Academic Year: 2017 - 2018

Supervisor: Prof. Rafal Zbikowski
Associate Supervisor: Dr. Ivan Petrunin
August 2018

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE,TRANSPORT AND MANUFACTURING
Autonomous Vehicle Dynamics and Control

MASTER OF SCIENCE

Academic Year 2017- 2018

OGUZ KAGAN ISIK

MANNED MACHINE INTERFACE (MMI) FOR COMMANDING AUTONOMOUS WINGMAN

Supervisor: Prof. Rafal Zbikowski
Associate Supervisor: Dr. Ivan Petrunin
August 2018

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science
(NB. This section can be removed if the award of the degree is based solely on examination of the thesis)

ABSTRACT

With development of 5th generation fighter jets, especially F-35, and improving technologies and capabilities of unmanned systems, manned-unmanned teaming has been very much in the foreground recently. Studies of interaction between manned and unmanned systems has been carried on for a long time. This interaction influenced by various topics, such as autonomy, communication and human factors. Manned machine interface (MMI) design is a key aspect to manage the interaction and build a team spirit among manned and unmanned units.

As air warfare increases in sophistication pilots are increasingly fulfilling a command, control and information (C2I) analysis role. One such scenario may be a pilot teamed with an autonomous wingman to support certain missions. This thesis aims to design an optimal, seamless but simple, MMI for a 5th generation fighter jet pilot in order to he/she is able to maintain his/her C2 role. Thus, criteria of MMI design was discussed for optimal design based on its functionality and simplicity and according to the criteria, a virtual interface model and an example simulation in MATLAB environment were presented in this thesis.

Keywords:

Manned-Unmanned Teaming, HMI.

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Prof. Rafal Zbikowski and Dr. Ivan Petrunin for their time, knowledge and patience throughout the process of writing this thesis.

I would also like to thank Paul Tremelling for his support and precious advice on design issues.

I am grateful to the Ministry of Education of Turkey for their financial support during all my MSc study.

Last but not the least, I would like to express my special appreciation to my wife Kubra and my daughter Azra for their patience, understanding and motivation which helped me to do my best.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

ROA	Remotely Operated Aircraft
UAV	Unmanned Aerial Vehicle
DoD	Department of Defence
FAA	Federal Aviation Administration
UGV	Unmanned Ground Vehicle
UMV	Unmanned Maritime Vehicle
MUM	Manned Unmanned Teaming
C2I	Command, Control and Information
ISR	Intelligence, Surveillance and Reconnaissance
MMI	Manned Machine Interface
TDL	Tactical Data Link
NCS	Net Control Station
UCAV	Unmanned Combat Air Vehicle
CNI	Communication, Navigation and Identification
DTDMA	Distributed Time Division Multiple Access
TDMA	Time Division Multiple Access
LOS	Line of Sight
BLOS	Beyond Line of Sight
NATO	North Atlantic Treaty Organization
HF	High Frequency
UHF	Ultra-High Frequency
MIDS	Multifunctional Information Distribution System
JTIDS	Joint Tactical Information Distribution
PCD	Panoramic Cockpit Display
SAM	Surface to Air Missile
PACT	Pilot Authorisation and Control Task
SAS	Situational Awareness Screen

1 INTRODUCTION

1.1 Background and Motivation

It is hard to find a consensus for the definition of unmanned aerial vehicle (UAV). There are several different definitions from different organisations for UAVs. Formerly, these types of aircrafts were named “remotely piloted vehicles” or “remotely operated aircrafts”. The term UAV comes from military authorities, and its use has become increasingly popular. Meanwhile, “uninhabited” or “unoccupied” are terms already used by civilian authorities instead of “unmanned”. The most useful and updated definition, which will be used in this paper, is mentioned by the DoD as

> a powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or nonlethal payload. Ballistic or semi-ballistic vehicles, cruise missiles, and artillery projectiles are not considered unmanned aerial vehicles. Also called UAV. (DoD, 2001)

Contrary to popular belief, UAVs have been actively used for only a few decades. The UAV idea is based on Tesla’s patent (Tesla, 1898). Tesla expected that if autonomous vehicles and drones were commonly used, it could decrease the possibility of war and encourage peace among countries. This idea is arguable, but it is clear that research into UAVs was triggered by this patent at the beginning of the 1900s.

Since the beginning of UAV research, military purposes have been at the forefront. The first example of a military UAV, the “Kettering Bug”, was used during WW1 by the United States, but it was not remotely controllable. With the development of the first remotely controllable aircraft in 1920, human operators became involved in UAV operations. The effects of human–machine interaction on UAV operations will be discussed in Section 2. Although interest in UAVs decreased during the peace between WW1 and WWII, UAVs were used for target practice thanks to their expendability. After this peaceful era, during WWII, a UAV

was used in a military operation in coordination with other units. This UAV was the V-1 bomber, designed by Germans, and it was capable of autonomous control. The V-1 bomber demonstrated the importance of UAVs and autonomy on the battlefield and led to development of modern military UAVs (Degarmo, 2004).

Modern UAVs, which have similar operational capabilities to those of recent types, can be classified in three generations. The first generation of military UAVs was seen during the Vietnam War. The UAVs used in Afghanistan and Bosnia such as the Gnat, Predator and Global Hawk are considered second-generation UAVs. The third generation of UAVs has been used in both the military and civilian areas for the last 10 years (Goraj, 2005).

As seen from the historical background of the development of UAVs, the UAV idea is not new, but it was not popular before 1990s. There might be different reasons for this, but the most significant one is the lack of technology. Developments in technologies such as autonomy, artificial intelligence and control systems over the last 30 years have reached the necessary level to develop more useful unmanned systems for use in real warfare environments.

Various classes of unmanned systems have been produced for different types of military and civilian missions, and their individual operational capabilities almost meet all requirements of these missions. However, unmanned systems should be able to work in cooperation with other unmanned (UAV, UGV, UMV) and manned systems during military missions. Thus, the individual capabilities of particular unmanned system classes should be used with a team spirit. This significant issue is called manned–unmanned (MUM) teaming, and Figure 1-1 illustrates the MUM teaming roadmap. According to this roadmap, in the next 10 years, researchers will be focused on controlling UAVs from other aircraft, the unmanned wingman and unmanned scout concepts, to increase MUM teaming performance. Similarly, controlling an autonomous wingman from another aircraft is also one of the key goals of this research.

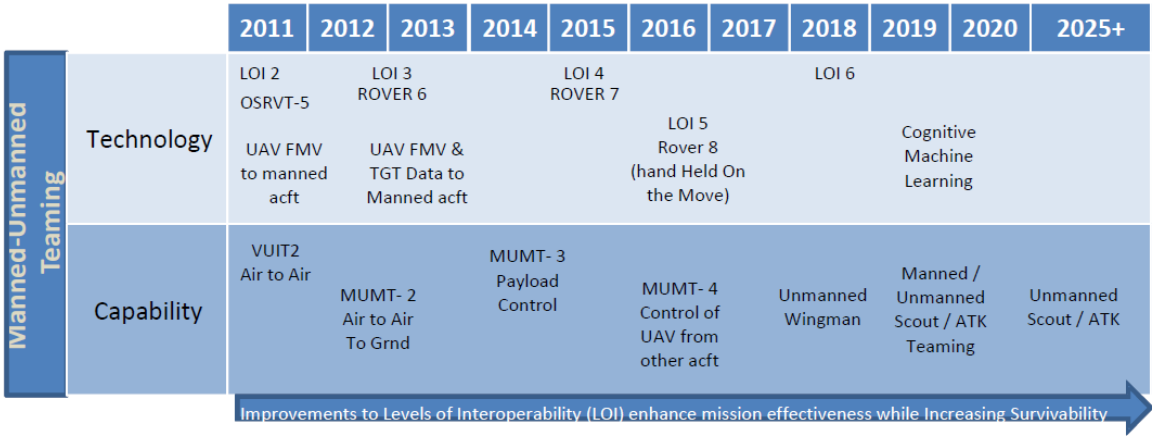


Figure 1-1 Manned Unmanned Teaming Roadmap.

Researchers face several challenges while working on MUM teaming. One of these challenges is communication. Due to the increasing number of unmanned systems used in missions, it is hard to ensure communication between all units. In addition, another more complicated challenge is controlling multiple unmanned units through one human operator. To deal with these challenges, man–machine interaction should be examined carefully. Man–machine interaction is directly related to the level of autonomy, communication protocols between units, human factors and user interface design. In this thesis, manned–machine interface design criteria are studied to find an optimal solution to maintaining command and control of an autonomous wingman.

1.2 Aims and Objectives

As air warfare increases in sophistication, pilots will increasingly fulfil a command, control and information (C2I) analysis role. One such scenario will be a pilot teamed with an autonomous wingman to support certain missions. Data exchange between the autonomous aircraft and the fighter is necessary so that the pilot can maintain control of the overall mission objectives. The main objective in this thesis is to provide an optimal man–machine interface design that is seamless but simple enough for a fifth-generation fighter jet (e.g., F-35) pilot. The level of the wingman's autonomy should be predictable and intuitive for the pilot, for example, the wingman should know to "stay in a certain formation" on the host aircraft until told otherwise or "execute an ISR task in a particular area of

responsibility". When automating tasks, it will be important to consider the different prioritisation of information the pilot requires depending on where the pilot is in the mission and what other information is required to be exchanged.

Key requirements for manned–machine interface design include the following:

- **Optimality:** Interface design should be optimal and based on workload, situational awareness and spending time.
- **Simplicity:** Interface design should be seamless but simple. Complex and distracting visual elements may decrease the pilot's situational awareness.
- **Functionality:** Interface design should be as functional as needed—not more or less. Extraneous design increases complexity, and underdeveloped design decreases the pilot's control over the wingman.

1.3 5th Generation Fighter Jets

The capabilities of fighter jets have been increasing since fighter jets were first designed. The first generation of fighter jets suitable for subsonic flight date to 1945. Ten years later, the second-generation F-100 fighter jet made supersonic flight possible and used the first radar systems. At the beginning of the 1960s, the third-generation fighter jets, of which the F-4 is the most famous, were developed. It can be said that the third generation is the first representative of the modern type of fighter jets, according to its role on the battlefield. The most critical improvements of this generation were its multirole capability and use of missiles. In just 15 years, fighter jets improved rapidly. After the fourth-generation fighter jets, this pace decreased. Fourth-generation jets have been used since 1970 and were the main power of technologically modern air forces until the release of the fifth-generation fighter jet in 2005. Fourth-generation fighter jets came into prominence because of their low observability, advanced avionics and guided weapons. Based on their aerodynamic capabilities, they do not provide less performance than fifth-generation fighter jets, but they have greater dependence on avionics and sensor fusion capabilities than the later generation. Fifth-generation fighter jets have increased fighter performance over the last generation, and they are able to build an operational network and provide sensor fusion and advanced sustainment (North, 2016).

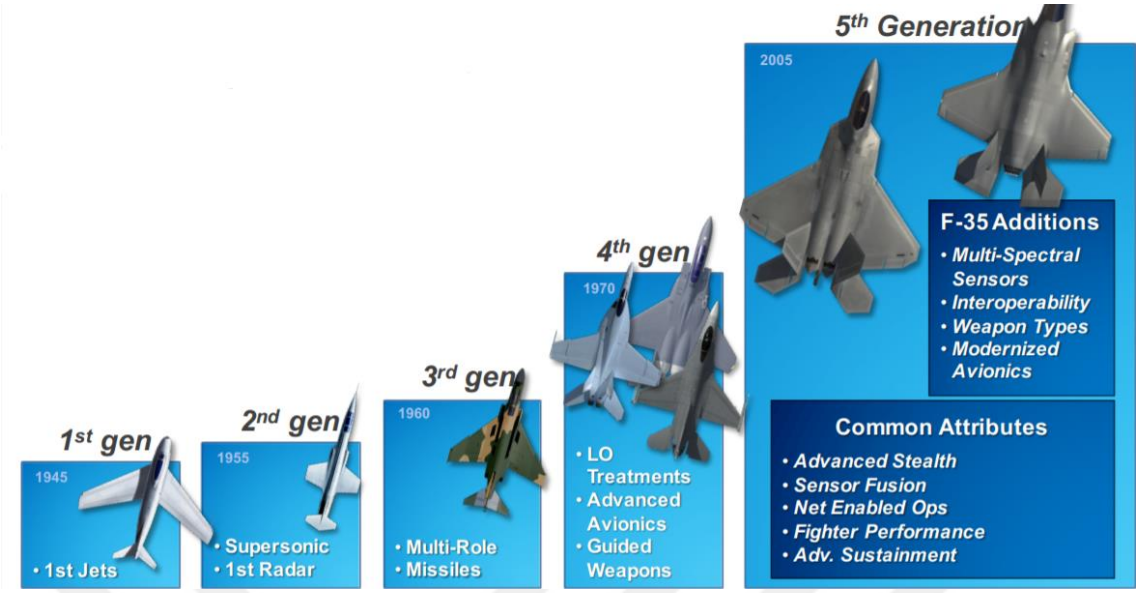


Figure 1-2 Development of Fighter Jets. (North, 2016)

A key issue that emerged during these developments is stealth technology. After the third-generation fighter jets had been released, the first attempts at stealth technology integration began. Although fast and high-altitude flight capability was stealthy in itself, developing radar systems necessitated night flights and non-manoeuvrable flights. Although these methods work for hiding from ground stations, they were not able to hide from other aircraft's radar systems. Thus, low observable aerodynamic design was introduced to solve this problem, as seen on the famous F-117A stealth aircraft. The F-117A's design provided good performance and near invisibility from radar, but it decreased agility and manoeuvrability. One of the aims of fifth-generation aircraft was to combine the capabilities of a fourth-generation fighter jet and a stealth aircraft. As a result of this aim, the most recent fifth-generation design, the F-35, looks like a combination of an F-16 and an F-117A. Its aerodynamic design makes it as manoeuvrable as an F-16 and as invisible as an F-117A for all types of operations and all kinds of weather conditions (Plessas, 2017).

Figure 1-3 Development of Stealth Technology (North, 2016)

When we look more closely at a fifth-generation fighter jet, the most significant difference from earlier generations is the role of the pilot. Lockheed Martin defines the design philosophy of F-35 as “return the pilot to the role of tactician.” (Skaff, 2010). In fourth-generation fighter jets, pilots had to manage sensor and system data, which can cause pilot overload. However, thanks to the sensor fusion engine of the F-35, pilots do not have to struggle with time-wasting activities and are able to focus on operational decisions (Skaff, 2010). Gen. Ellen Pawlikowski said, “I can see a scenario where you’ve got an F-35 orchestrating an attack with 20 RPAs that are weapons-equipped, and that F-35, with all its sensors and communications, is essentially an orchestrator” (Clark, 2014).Thus, it can be said that a MMI design that is adaptable to the sensor and communication capabilities of a fifth-generation fighter jet to control crowded swarms is a vital topic. In addition, it can be expected that the sixth generation of fighter jets will be almost completely autonomous.

1.4 Thesis Structure

This thesis consists of five chapters:

In Chapter 1, the introduction, background and motivation of the main and related topics were mentioned; aims and objectives were stated; the fifth-generation fighter jet concept was introduced as a complimentary topic; and the chapter thesis structure was given.

In Chapter 2, Man–Machine Interaction, autonomy and communication issues and key concepts were outlined to detail the interaction between manned and unmanned systems. First, the basic terms were defined and the main issues surrounding automation were addressed. After levels of autonomy and their effects on man–machine interaction were introduced, trade-offs between autonomy level, workload and unpredictability in interface design were explained. In the second part of the chapter, TDLs were analysed according to their capabilities and communication challenges, such as bandwidth, timeliness and LOS range.

In Chapter 3, Interface Design Criteria, the cockpit design for the fifth-generation fighter jet F-35 was analysed to obtain more realistic design criteria. After this, general MMI design criteria were assessed and compared to the F-35 cockpit to analyse the feasibility of my design.

In Chapter 4, MMI Design, the suggested MMI design, which was produced with MATLAB Graphical User Interface tools, was explained section by section through validating design criteria. Related frames of MMI were also presented.

In Chapter 5, Conclusions and Recommendations, a brief conclusion of the findings and recommendations for future works were provided.

2 Man-Machine Interaction

Man–machine interaction is a complex and sensitive topic due to bitter failures in UAV history. So many accidents and unsuccessful missions have occurred in UAV operations because of interaction problems. Although some of these problems were directly caused by machines, human operators have also been a major problem in the operational loop. Therefore, man–machine interaction can pose a problem if the hierarchy and responsibilities are not determined clearly. The level of autonomy, or what kind of work the machine performs, and the communication procedure between MUM systems will be touched on in this section to analyse their effects on man–machine interaction.

2.1 Autonomy

Autonomy has been widely analysed in different areas such as aircraft, space, traffic control, medical and business. In the military area, especially after the popularity of unmanned systems increased, autonomy has become one of the most crucial design concepts because of its decisive effect on man–machine interaction.

The main aim of the use of autonomy is to make reflexive works automated to decrease workload and increase intelligence to deal with complex works. At this point, oft-confused terms like autonomy, automation and intelligence should be defined.

Autonomy: The ability or opportunity to make your own decisions without being controlled by anyone else (Longman Dictionary of Contemporary English [LDOCE], 2018).

Automation: The use of computers and machines instead of people to do a job (LDOCE, 2018).

Intelligence: The ability to learn, understand and think about things. (LDOCE, 2018).

As understood from these definitions, an autonomous system is somewhere between completely automatic and completely intelligent. So, two questions should be asked to determine the scope of automation in a system:

- What kind of responsibilities should be undertaken and what kind of decisions should be made by the machine or a human operator?
- How is the level of automation assessed?

Classifying the type of decisions made by the human operator is important to defining the optimal interaction level between operator and machine. According to Rasmussen (1976), there are three types of cognitive behaviours: skill-based, rule-based and knowledge-based. Skill-based behaviours can be defined as reflexive and routine behaviours, which cause a majority of the workload but can be easily automated, like cycling and swimming. Rule-based behaviours have their specific requirements based on procedures. If the conditions and procedures are defined clearly, machines can easily follow the steps based on predefined rules. Tactical definitions and missions may be examples of rule-based behaviour. However, knowledge-based behaviours are hard to automate because they need much more intelligence based on experience. Additionally, this type of decision can be critical for mission success, so it is best performed by a human.

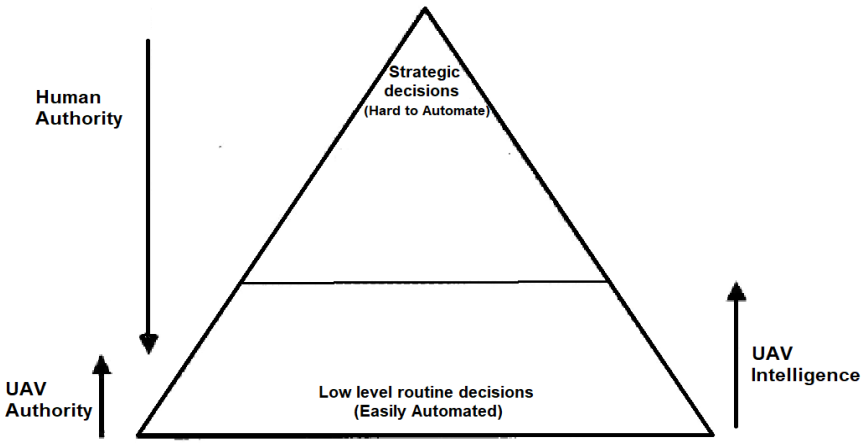


Figure 2-1 Authority – Intelligence Relationship (White, 2003)

Level of Autonomy	Description of Man-Machine Interaction
Level 10	Computer does whole job if it decides it should be done and tells human if it decides he should be told.
Level 9	Computer does whole job and tells human what it did, and the computer decides he should be told.
Level 8	Computer does whole job and tells human what it did only if human explicitly asks.
Level 7	Computer does whole job and necessarily tells human what it did
Level 6	Computer selects action, informs human in plenty of time to stop it
Level 5	Computer selects action and implements it if human approves
Level 4	Computer selects action and human may or may not do it
Level 3	Computer helps determine the options and suggest one, which human need not follow.
Level 2	Computer helps by determining the options
Level 1	Human does whole job up to the point of turning it over to the computer to implement

Table 2-1 Levels of Autonomy in Man-Machine Interaction

Figure 2-1 illustrates how responsibilities are shared based on intelligence level (White, 2003). Although the top represents the decisions, which need higher-level intelligence, the bottom represents routine decisions. To obtain an optimal and functional design, the key issue is specifying the boundary between operator and machine decisions. If too much responsibility is given to an operator, the workload is increased, which decreases the operator's situational awareness. Meanwhile, if the machine is responsible for the critical and higher-level decisions, it would be risky and unpredictable.

To optimally split the responsibilities, the second question should be answered. Levels of autonomy provide clues to decide what kind of decision should or should not be autonomous. A model for levels of autonomy has been suggested by (Sheridan and Verplank, 1978) on a 10-point-scale, where level 1 shows the human-only behaviour and level 10 shows fully autonomous behaviour, as seen in Table 2-1. At the lower levels, the computer only provides some advisory information that the operator needs to analyse before making a decision. With an increasing level of autonomy, the operator's analysing role decreases, and, instead, the operator only makes decisions. At the top levels, the operator only executes the operator's tactician role.

An adapted version of these levels of autonomy and autonomous control of UAVs and a prediction about the future of the autonomy levels in UAV technology is illustrated in Figure 2-2.

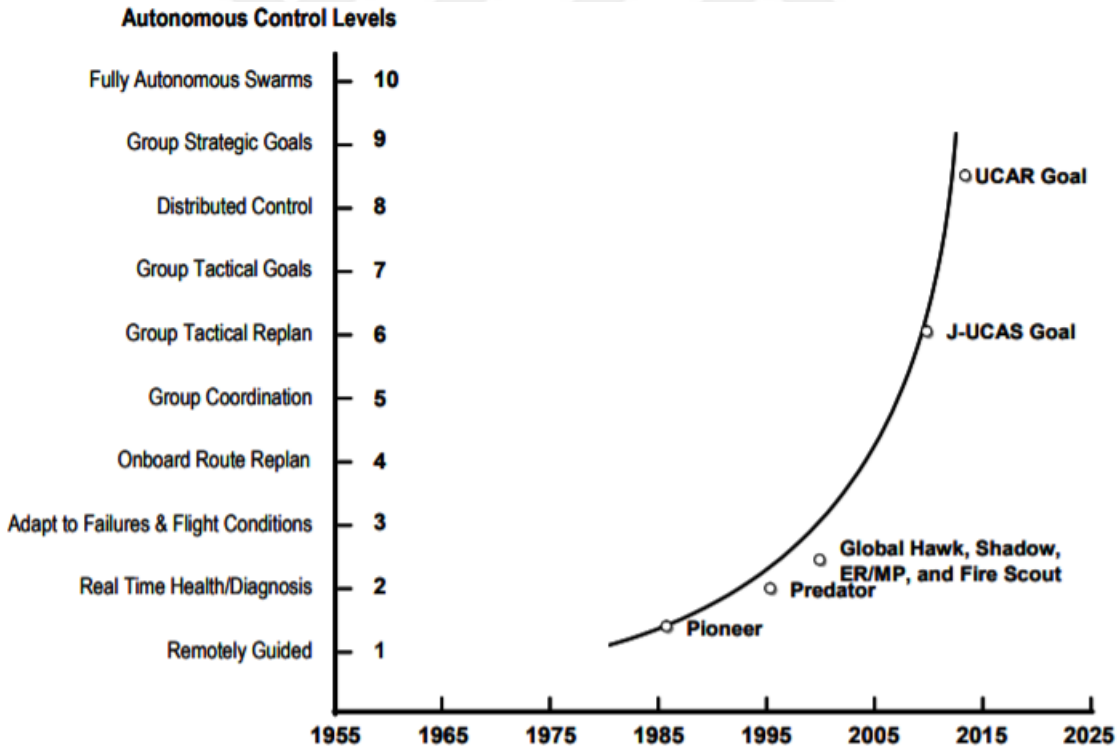


Figure 2-2 Trend in UAV Autonomy (of the Secretary of Defense, 2005)

After defining the type of human behaviours and levels of autonomy, the next topic involves how the level of autonomy effects an interface design to control a

wingman or a remotely piloted aircraft. (Miller, Pelican and Goldman, 2000) conducted a trade-off study between workload, unpredictability and adaptiveness as a result of level of autonomy. As seen in Figure 2-3, workload and unpredictability cannot be decreased simultaneously. Although lower autonomy causes a higher workload, higher autonomy causes higher unpredictability, and, in both cases, adaptiveness of design increases dramatically. Because of this, the optimal adaptiveness depends on the balance between workload and unpredictability.

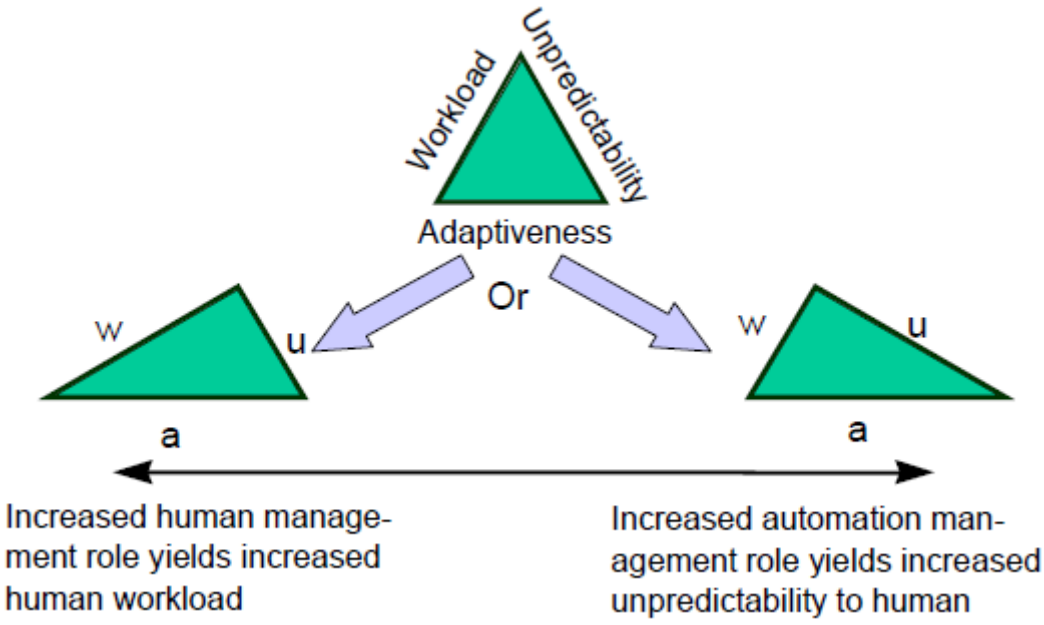


Figure 2-3 Authority – Intelligence Relationship (Miller, Pelican and Goldman, 2000)

It is possible to use different autonomy levels for different functions instead of defining all interface functions at the same autonomy level. According to (Parasuraman, Sheridan and Wickens, 2000), a four-stage model of the human information process can be defined with the following stages: each stage may have its own autonomy level to meet the interface design requirements.

a) Information acquisition
b) Information analysis
c) Decision and action selection
d) Action implementation

With this approach, different combinations of autonomy levels can be described for collecting or analysing data from sensors or for decision-making and action selection. For example, if you have a large amount of information from various types of sensors, higher-level acquisition and information automation is better to decrease operator workload. However, if the mission is critical, action automation should be at a lower level so that the human takes the responsibility, according to the operator's knowledge and experience.

2.2 Communication

To provide sufficient communication among manned or unmanned military units, TDLs have been widely used for a long time. As a basic definition, "data link" involves system, application and data protocol components inside of a communication system, and a TDL is a data link used for supporting military action (Stoica et al., 2016). Since the beginning of World War II, TDLs have been renovated, and different types of TDLs have been designed to meet different types of operational requirements. The first communication issue in military operations was to determine an aircraft as a friend or foe, and the first data links focused on solving this issue. After adding more tactical information, such as airspeed, location, direction and altitude in data link messages, the first TDL was born (Sorroche, 2012).

The main aim of all sorts of TDLs is to provide a successful command, control and communication (C3) system in a high-density, rapidly changing warfare environment to effectively manage military operations (Golliday, 1985). Every tactical data link has unique specifications to meet different requirements. For example, although some of them provide point-to-point communication, others use netted communication, which consists of various types of tactical units. Every type has its own advantages and disadvantages.

Although some TDLs are obsolete, almost all of them are already actively used, even if their applications are limited. Data link evolution can be divided into two generations. The first generation of TDLs, comprising Link 1, Link 4, Link 11, Link11B and Link14, was developed in the 1950s and 1960s, and it had a limited data rate and capacity. The second generation of TDLs, Link 16 and Link 22, are

more functional and have higher data rates and capacity. Characteristics of some basic type of TDLs are illustrated in Table 2-2. Specifications vary according to characteristics such as modulation technique, data rate, message and protocol standards (Stoica et al., 2016).

Characteristics	Link 1	Link 11	Link 16	Link 22
Frequency	Point-to-point land line	HF/ UHF	UHF/ Spread	HF/UHF Spread
Speed (bit/sec)	1,200	1,800	> 57,600	-
ECM resistance	-	-	x	x
Crypto-secure	-	x	x	x
Nodeless	-	-	x	x
Extended LOS	-	-	x	x
Antijam	-	-	x	-
Data rate (kbps)	1.2	1.3 ÷ 2.25	2.8 ÷ 115.2	2.4
Standard message	S-series	M series	J series	J series
Participants		4 ÷ 8	> 128	40
Voice circuits	-	-	2	-
Architecture	Duplex digital	Radio Broadcast	TDMA	DTDMA

Table 2-2 Characteristics of TDL Networks

Link 1 and Link 11 have lower data rates than Link 16 and Link 22. Link 1 is a point-to-point data link and can connect only two units at the same time. Link 1 uses a fixed data set and has limited usage. It connects some fixed points to share some fundamental information for situational air awareness (Golliday, 1985).

Links 11, 16 and 22 are netted data links. Netted data links use a common channel with a common frequency so that a unit can send data to many other units. Maintaining a netted network is harder than maintaining a point-to-point network, and they need to be managed by more sophisticated protocols. Link 11 uses a polling protocol managed by a net control station (Golliday, 1985), which is to interrogate the units sequentially and to make sure that only one user talks during a specific period. Link 11 also uses a secure data link, but it is not robust against jamming. Link 11 works in HF or UHF bands and provides beyond-line-of-sight (BLOS) communication (Zhao, Kou and Wu, 2012).

Link 16 is the most common TDL for all kinds of tactical air, naval or ground units for LOS communication these days. Unlike Link 11, Link 16 does not need to use

an NGS unit to manage data transfer thanks to the time division multiple access (TDMA) technique. With TDMA, users are allocated a 7.8125 ms time slot to send a message to the other users in the network. There are 128 time slots/second for data transfer and several different information can be transmitted in every time slot (Golliday, 1985). Hence, Link 16 is a high-capacity data link. The Joint Tactical Information Distribution System terminals and Multifunctional Information Distribution System are also used by Link 16 (Northrop Grumman, 2014).All these functions make Link 16 robust against jamming and provide nearly real-time data exchange between joint units.

Finally, Link 22 can be considered an updated version of Link 11. It uses HF and UHF bands for BLOS communication in up to a 500 km radius with an omnidirectional antenna. The main purpose of Link 22 was to develop a resistant data exchange network for NATO allied forces. Although its data rate is much lower than Link 16, Link 22 is able to operate even in bad transmission conditions. Due to the distributed time division multiple access protocol, the whole system is never affected by a specific unit failure (Stoica et al., 2016).

One of the most crucial challenges of TDLs is timeliness when the TDL is used to track the position of a high-speed target. Because the target-position data cannot be sent continuously, a small time delay between two consecutive position data points may easily affect timeliness performance. Because users expect nearly real-time data from the data link, delays can jeopardise the operation. The most common reporting delays are queuing delays, which are caused by intense data traffic. Especially in Link 16, the combination of frequency hopping and TDMA provides enough data slots to tolerate the intense traffic. However, in the future, because of the increasing number of unmanned aerial systems in tactical military missions, it can be predicted that Link 16 will not be enough to manage this data traffic.

Another critical challenge of TDLs is interoperability. As seen from the TDL characteristics, every data link has its own protocol. If different TDLs are used in the same operation, some units cannot understand each other because they do not use the same transmission protocol. There are two ways to achieve

interoperability (Golliday, 1985): producing a general data link used by all units or using a translator unit between different data links. Although the first solution appears suitable at first, it is extremely hard to integrate a common TDL for every NATO ally. Additionally, conversion of existing systems to a new TDL would be quite costly. However, using a translator is also not effective due to some extra data that should be transferred on the data links. Some translator and multilink processor solutions are suggested by (Ozturk, Esen and Sahin, 2014) and (Zhao, Kou and Wu, 2012).

Because of the development of fifth-generation fighter jets and unmanned combat air vehicles, MUM teaming issues have become more prominent. For a successful team operation, communication is a key challenge. To provide safe and quick communication, the Multifunctional Advanced Data Link (MADL) is integrated into fifth-generation fighter jet avionics. One of the most functional integrated communication, navigation and identification systems which is linked to MADL is AN/ASQ-242, which was designed by Northrop Grumman (Northrop Grumman, 2014).This system provides simultaneous operation and high sensor fusion capability to pilots thanks to its powerful data analysis engine and high-speed communication protocols.

3 Interface Design Criteria

In this section, design criteria and the requirements of a man–machine interface will be assessed. First, it is worth analysing the cockpit and interface design of a real fifth-generation aircraft, the F-35, to specify realistic requirements. Second, general MMI design requirements will be examined based on the literature. According to (Howitt and Richards, 2003), designing an effective MMI to control a UAV requires examining three concepts:

- Interaction requirements
- Information requirements
- Human cognitive behaviour

This sequence will be followed in this chapter as a reference to present design criteria and requirements.

3.1 F-35 Cockpit Specifications

Undoubtedly, the best way to decide design criteria of the wingman control interface is to look at a real fifth-generation fighter jet. The F-35 is the second fifth-generation fighter jet and is produced by Lockheed Martin, Northrop Grumman and BAE Systems, after the F-22 Raptor, but there are significant differences, depending on their cockpit design.

The design philosophy of the F-35 cockpit, defined by Lockheed Martin, is to “return the pilot to the tactician role” (Skaff, 2010). In fourth-generation fighter jets, pilots spend a remarkable amount of time managing data and information, so they have very high workload in addition to flying and fighting. However, in fifth-generation fighter jets, with their coherent design philosophy, pilots are able to focus on tactical and intelligence-based decisions while computers do routine and time-intensive work. The F-35 provides these abilities to pilots thanks to its cutting-edge cockpit design. The cockpit consists of three elements: panoramic cockpit display (PCD), active sidestick and active throttle. Conventional physical switches are not required for operational functions; instead of these, touch, cursor hooking and voice recognition are used as control inputs. Physical switches are

preferred for safety-critical functions such as engine start/stop or landing gear. The cockpit view can be seen in Figure 3-1:

Figure 3-1 Cockpit of F-35

Instead of physical switches and buttons, PCD meets all necessary operational needs through its two 10-inch by 8-inch, multifunctional, LED touch screens. These two LED panels are supplied with different energy sources and different computers, so if one of the panels is lost, the pilot can control the aircraft using the other panel. In addition to this, it provides effective use of energy and computational capability. More than 20 graphical user interfaces, such as situational awareness and weapon interfaces, can be selected and displayed, at a maximum of four at the same time, on this LED panel. It is similar to choosing an application from a smartphone. The size of the interfaces is also adjustable and can be changed by the pilot to settings such as 5 by 5 inches, 5 by 8 inches or 10 by 8 inches. Thus, pilots can easily arrange the size of their user interfaces according to their importance to the active mission (Skaff, 2010).

There are 10 switches on the active sidestick, which is placed on the right-hand side, and there are 12 switches on active throttle, which is placed on the left-hand side. Some of the most used and time-critical functions are defined on these switches. For example, there is a switch for the cursor hooking function on the throttle, and there is a target designation switch on the sidestick (Skaff, 2010).

Human factors also were heavily considered in the design process. The size of the cockpit and the ejection seat are comfortable enough on long missions for pilots of many sizes. Furthermore, the size of the touch screen buttons and sidestick switches are appropriate to use while wearing a chemical-biological protection glove. Visual signs on the interfaces were designed with optimum colour, contrast and brightness specifications to provide maximum situational awareness for the pilot (Skaff, 2010).

3.2 Interaction Requirements

Interaction between the pilot and autonomous wingman is the key element for collaboration. For building an optimal pilot–UAV interaction, both the UAV and pilot should know their role in the mission, share the same goals, have enough knowledge, have access to appropriate information and be able to communicate with other related units (White, 2003).

First, the pilot and wingman should both understand their role in regard to the primary goal and auxiliary goals in a mission. For example, destroying a critical enemy building may be the goal of the mission. However, the area may be defended by surface-to-air missile (SAM) launchers, and their positions unknown. Thus, detecting the position of the SAM launchers could be an auxiliary goal. Mission planners can predict possible sub-goals according to the mission scenario.

After defining the goals, it is important to specify the role of each team member to successfully accomplish the mission. Detecting SAM launchers is a risky job for a pilot, but it can be undertaken by a wingman. Therefore, the pilot is able to fulfil his or her main objective in safety. Additionally, all off-duty roles should be specified. For example, the wingman should be able to fly by itself and draw its

flight path without operator control if the operator is a fighter jet pilot. Nevertheless, the use of weapons should be under the control of the pilot, except when avoiding enemy fire. These role definitions can be expanded according to needs. While defining these roles, it should be considered that whereas an operator is able to deal with knowledge-based and tactical-strategic decisions, a computer shows better performance on routine analysis and reflexive decisions.

White (2003) defined two approaches to finding optimal balance of authority between operator and UAV. The aim of the first approach is to minimise autonomy in interaction to give more control and authority to the operator. In contrast, the second aim is to minimise operator presence in the operational loop so the operator workload can be decreased. Balancing these two approaches is the best way to finding an optimal solution. The MMI design gives enough authority to the pilot while using an appropriate level of autonomy to decrease pilot workload. Every function in the interaction process needs a different level of support. These support levels have been defined in the Pilot Authorisation and Control of Tasks (PACT) system, which is presented in Figure 3-2. This system provides five support levels as a more applicable military version of the level of autonomy definitions given in Table 2-1.

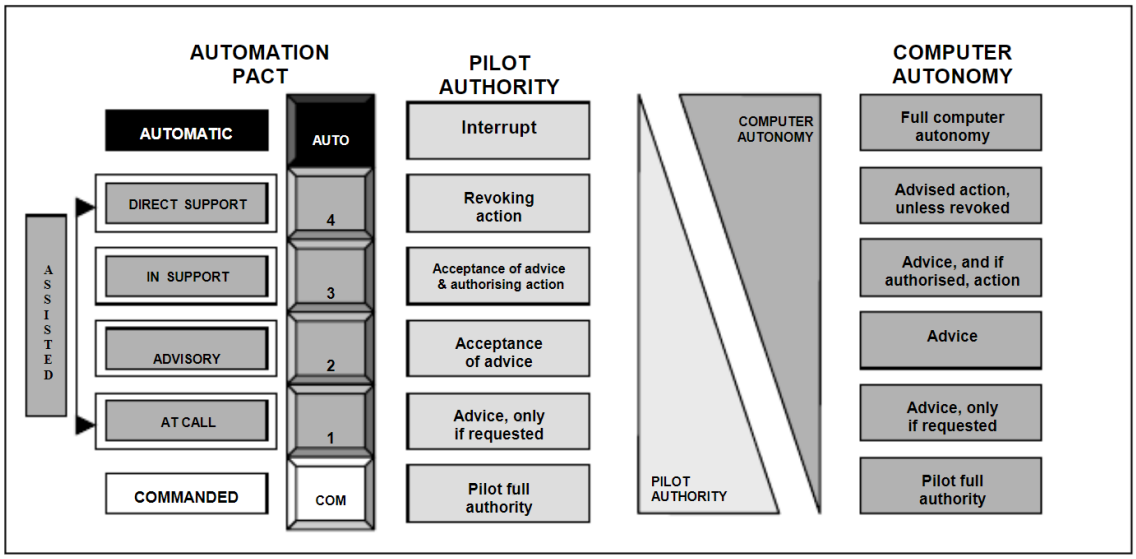


Figure 3-2 Pilot Authority & Computer Autonomy (PACT) System

By describing functions based on the PACT system, it is easier to illustrate the pilot–wingman interaction in the MMI design. Although high PACT levels are suitable for urgent actions, such as self-defence, UAV functions or iterative and computational actions including target detection, low PACT levels are suitable for pilot-dependent actions, such as the use of armaments and strategic decisions. It is harder to decide the PACT level of some tactical functions because of their interactive nature; this type of function can be assigned between levels 2 and 4 (assisted), depending on the MMI design.

3.3 Information Requirements

After defining interaction requirements, use of an appropriate information exchange procedure between pilot and wingman and the definition of its infrastructure are crucial for a successful MMI design. Provided information should be enough to be maintained operational environment by pilot. However, if a large amount of information is presented to the pilot simultaneously, the pilot faces a high workload, which decreases pilot performance. Thus, relevant information should be presented in every step of the operation in a logic order and cognisable form.

The amount of exchanged information directly affects the autonomy level of MMI design (Howitt and Platts, 2002). With increasing autonomy level, the pilot needs less information to manage the wingman's behaviour. So, data traffic should be designed between manned and unmanned units according to their intelligence. Hence, more functional and optimal data communication can be provided. If the autonomous wingman can fly by itself, presenting flight data such as air speed and roll and pitch angles in MMI is not necessary. However, tactical information such as coordinates, mission status and safety information like fuel/battery level should be provided to the pilot. (Miller and Parasuraman, 2007) discussed required information types for a military application, and (Ruiz et al., 2016) discussed them for a civilian interface application.

Another related issue that affects data flow between units is data link limitations. TDLs have a bandwidth limit, and because of increasing data traffic from an increasing number of units, these bandwidth limits may be exceeded. Because

of this, some information can be lost or delayed even if received. Another limitation is coverage area. Recent TDLs provide safe and fast communication in line-of-sight (LOS) range, which is approximately 100 nautical miles. If operational units are out of this range or a mountainous environment obstructs the LOS communication, units have to use satellite communication, which causes communication delays. In consideration of these limitations, information requirements should be specified optimally for MMI design.

3.4 Human Cognitive Behaviour

Although interaction and information requirements are more related to system-based specifications, human cognitive behaviour directly represents physical and psychological interactions between operator and interface. Hence, the most critical design requirements are based on human cognitive behaviour. The main goal when displaying information in a graphical interface is to maximise situational awareness and maintain minimum workload. Situational awareness is "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" (Endsley, 1995). According to this definition, situational awareness consists of three levels: perception, comprehension and projection.

A large variety of input and output methods can be used for pilot interface interaction. As mentioned before, in the F-35 cockpit, touch, cursor hooking and voice recognition are used as input methods. In addition to these, gesture or body movements can be used as inputs (Popov et al., 2016); (Soto-Gerrero and Ramrez-Torres, 2016). Additionally, a flat-screen display, a head-mounted display, sound or vibrations can be used as an MMI output. I mainly focus on a multifunctional touch screen similar to the one used in the F-35 cockpit for both inputs and outputs. Every display unit should be designed according to size, location, colour, grouping and distinctiveness to increase the pilot's situational awareness when using the touch screen (Howitt and Richards, 2003).

Most of the time, the interface size is limited because there is limited space in the cockpit for interface design. However, thanks to improving touch screen technology and usage of multifunctional touch screens, different interfaces can

be displayed on the same screen panel with adjustable sizes. Placement of user interfaces in the F-35 cockpit can be seen in Figure 3-3.

Figure 3-3 Multi-functional touch screen of F-35.

The size of every sub-element, such as the icon, text or button, is as important as the size of the interface. Unlike the size of the interface, designers have more freedom to choose the size of sub-elements. Text and icons should be comprehensible to increase pilots' awareness, but buttons are also interactive input units and their size directly affects the speed and accuracy of pilot performance. (Conradi, Busch and Alexander, 2015) experimented to find the optimal touch-screen interface size. This experiment was organised with different touch button sizes, 5 x 5 mm, 8 x 8 mm, 11 x 11 mm and 14 x 14 mm, and aimed to understand the effect of the button size on user performance based on reaction time and accuracy. The results demonstrated that there is a dramatic performance increment between 5 mm and 8 mm buttons. However, between 8, 11 and 14 mm buttons, there is almost no difference based on reaction time, but there is a very small decrease in accuracy. So according to this research, 11 mm may be considered the optimal touch button size.

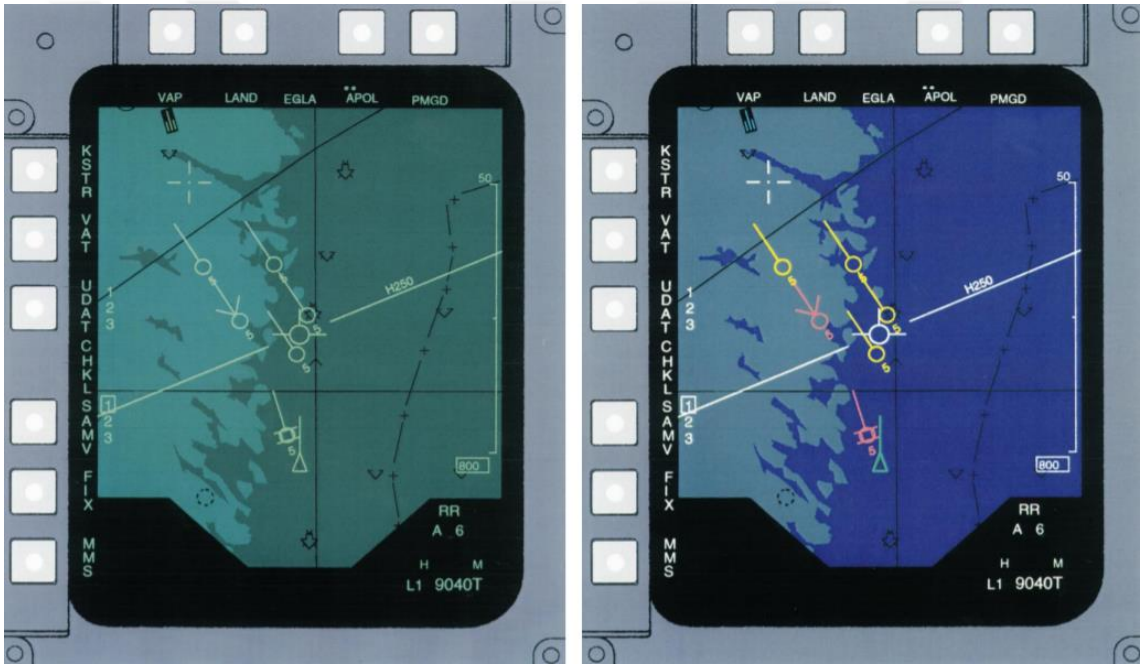
Another experiment on buttons made by (Schedlbauer, 2007) aimed to understand effect of button size and gap size between the buttons on user performance with three methods of input: finger, stylus and trackball. According to the results, although button size has a small effect on accuracy, gaps have almost no effect on either time or accuracy. In addition, the best speed performance was measured with finger touch, and the best accuracy was measured for the trackball; therefore, if the space for user interface is limited, gaps can be minimised. This study also concluded that if a touch-screen function needs quick action, touch input is the best way, but if it needs accuracy, using a cursor is a better solution. The F-35 cockpit uses both methods.

The location of the user interface is another MMI design issue. The location of the interface and its elements should consider visibility and reachability. The best form is an adjustable location, but this is not always possible. In terms of visibility, no objects should block the screen, including the pilot's hand. For example, if the user interface is just in front of the pilot, and the frequently used buttons are placed on the top, the pilot's hand would block the information screen at every touch. Maximising reachability requires an analysis of which hand or hands the pilot uses to touch the screen. (Lewis, 2015) mentioned that especially in military aircraft cockpits and ground control stations for UAV commanding, both hands have different responsibilities. The right hand is generally used to control the joystick, which controls the jet's flight. The left hand maintains the throttle and interface. This type of cockpit design can be seen in Figure 3-1 for the F-35 fighter jet and Figure 3-4 for the Ikhana Ground Control Station.

Figure 3-4 Operator in Ikhana Ground Control Station

Grouping together the related information is a supplementary topic to finding an optimal location for the interface elements. Human nature is inclined to search for related information, so grouping related icons or texts together increases the pilot's perceptive and comprehensive capabilities and decreases the reaction time. Drawing a line between two groups of information and leaving an appropriate amount of blank space are two of the most common ways to separate one group from another.

Colour is one of the most effective factors for increasing situational awareness on a crowded information screen. Although colours can be used as an information classification tool, they are also useful for presenting alerts alongside voice stimulus. When polychrome interfaces are compared with monochrome interfaces, it is clear that multicolour interfaces provide much better situational awareness to the pilot (Derefeldt et al., 1999). Figure 3-5 illustrates a monochrome and a polychrome interface comparison. In addition to these, colour coding, which is also used in F-35 interfaces, is beneficial to increasing pilot performance. Colour coding is the designation of colours to a specific function or meaning.



a) Monochrome Interface **b) Polychrome Interface**

Figure 3-5 Comparison of Monochrome and Polychrome Interfaces

Lastly, human cognitive behaviour is also affected by the distinctiveness of icons or texts. Icons are generally more meaningful than texts and can provide a great amount of information quickly. Furthermore, icons are more attractive if they feature appropriate colour selection. However, icons can be more complex and confusing than simple text in certain situations.

4 Suggested MMI Design

In this section, design assumptions and limitations, such as communication and the intelligence capabilities of the wingman, will be defined according to Chapter 2. In addition, some assumptions about physical limitations of the interface design, such as its size, will be mentioned based on the requirements in Chapter 3. After defining these assumptions, the user interface design will be presented section by section with a discussion of their validity based on functionality. At the end, an example simulation scenario will be explained to show how the design works.

4.1 Assumptions and Limitations

Man–machine interface design is not only influenced by man–screen interaction. The whole system consists of sensors, communication protocols and the units' intelligence level; the complete interaction between man and autonomous system affects the MMI design. Hence, assumptions and limitations have direct influence on the design.

It is assumed that every wingman in this MMI design has homogeneous capabilities regarding sensors, armaments, manoeuvrability and intelligence. Every wingman in the team is able to maintain a task with sufficient performance, and the pilot does not need to be worried about the wingman's capability.

Link 16 TDL is used as the LOS communication protocol in the UHF band, and every wingman is able to communicate not only with its leader but also with one another to exchange relevant information to achieve mutual goals. It is also possible to build a satellite communication for beyond LOS communication. In addition to these, the communication line has enough bandwidth to provide seamless data transfer for various data types.

Based on the wingman's levels of autonomy and intelligence, it has enough intelligence and knowledge about the tasks to make specific decisions about achieving the mission goal. Accordingly, the pilot works as a task planner and can start, stop or revise a task. The flow of the task is pursued by MMI, and

relevant information is presented to the pilot via graphical interface. The pilot also does not need to take care of sensor and data management. The autonomy level of the suggested MMI design, based on Miller, Pelican and Goldman's four-stage model, is specified as:

- Information acquisition : Level 9
- Information analysis : Level 9
- Decision and action selection : Level 3
- Action implementation : Level 6

Thanks to a high level of autonomy at the information acquisition and analysis stages, the pilot workload decreases because the MMI is responsible for sensor and data management. However, the pilot takes most of the responsibility for tactical decisions. At the implementation level, the MMI can maintain the given task and decide to complete it. Aside from the four-stage model, based on the PACT system, the suggested MMI design can be defined as Level 3 (in support), which means MMI advises and, if authorised, executes an action. However, the pilot is responsible for authorising and accepting the advice.

A realistic MMI design should account for F-35 cockpit specifications to specify the physical and functional limitations of the MMI design. A multifunctional touch screen is used for interaction and input methods through touch, cursor hooking and voice, but voice control is not used in this project. More than 20 different interfaces work on the F-35 touch screen, and in this project, one more interface is suggested for this system. So, this suggested MMI design should be applicable for real cockpit systems.

One of the most crucial physical limitations is interface size, but designing a fixed-size interface is not possible for the F-35 because of its adjustable screen size, as mentioned in Section 3. Hence, the suggested interface has been designed for three different sizes: 5 by 5 inches, 5 by 7 inches and 10 by 7 inches. The general MMI view in different sizes can be seen in Figures 4-1 to 4-3. Pilots can change the MMI size with a touch according to their needs. Although the smallest size consists of just a situational awareness screen, the other two also have an

information display screen that illustrates useful information about the wingmen, such as fuel level.

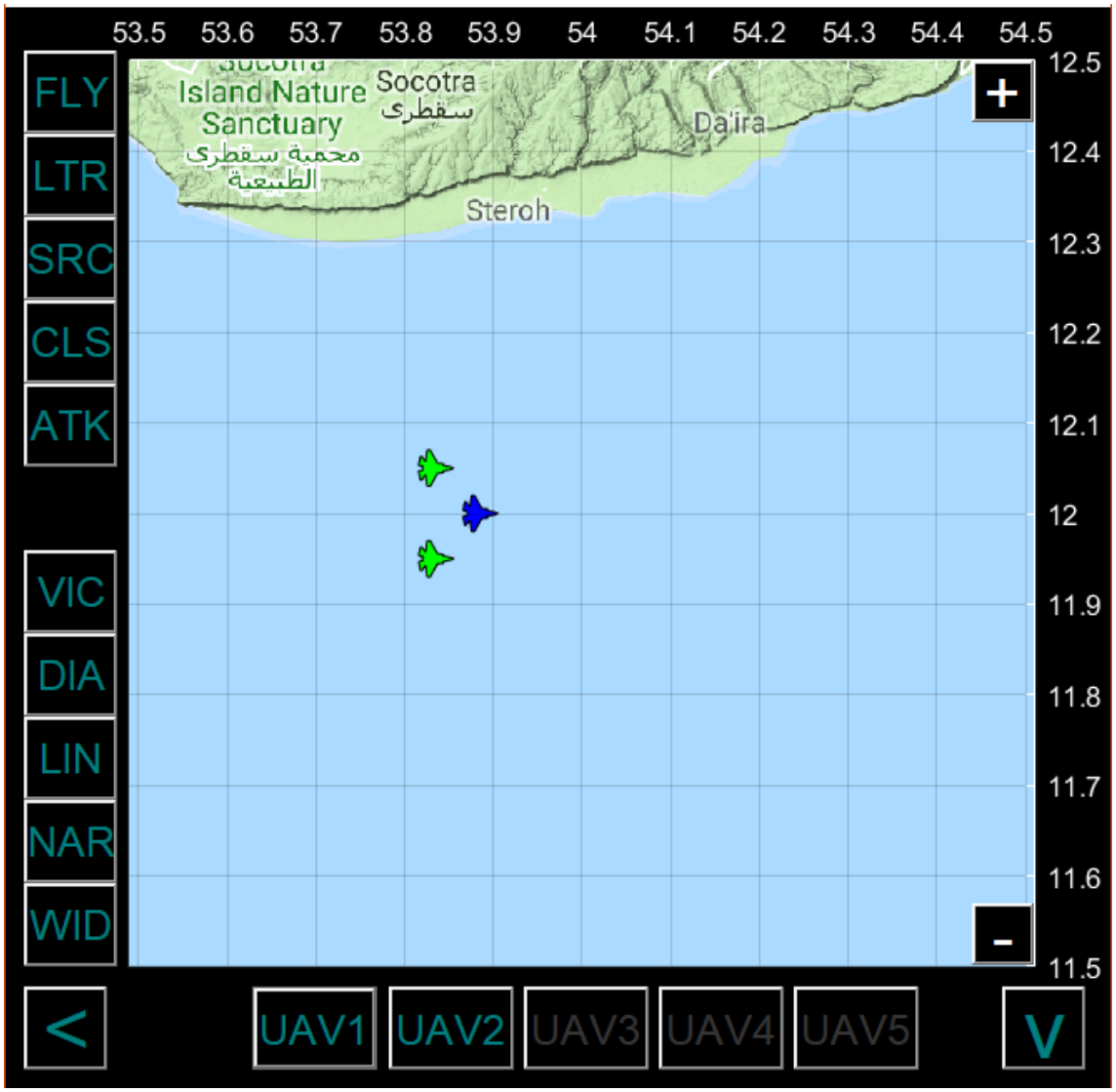


Figure 4-1 MMI Design 5 x 5 Inches

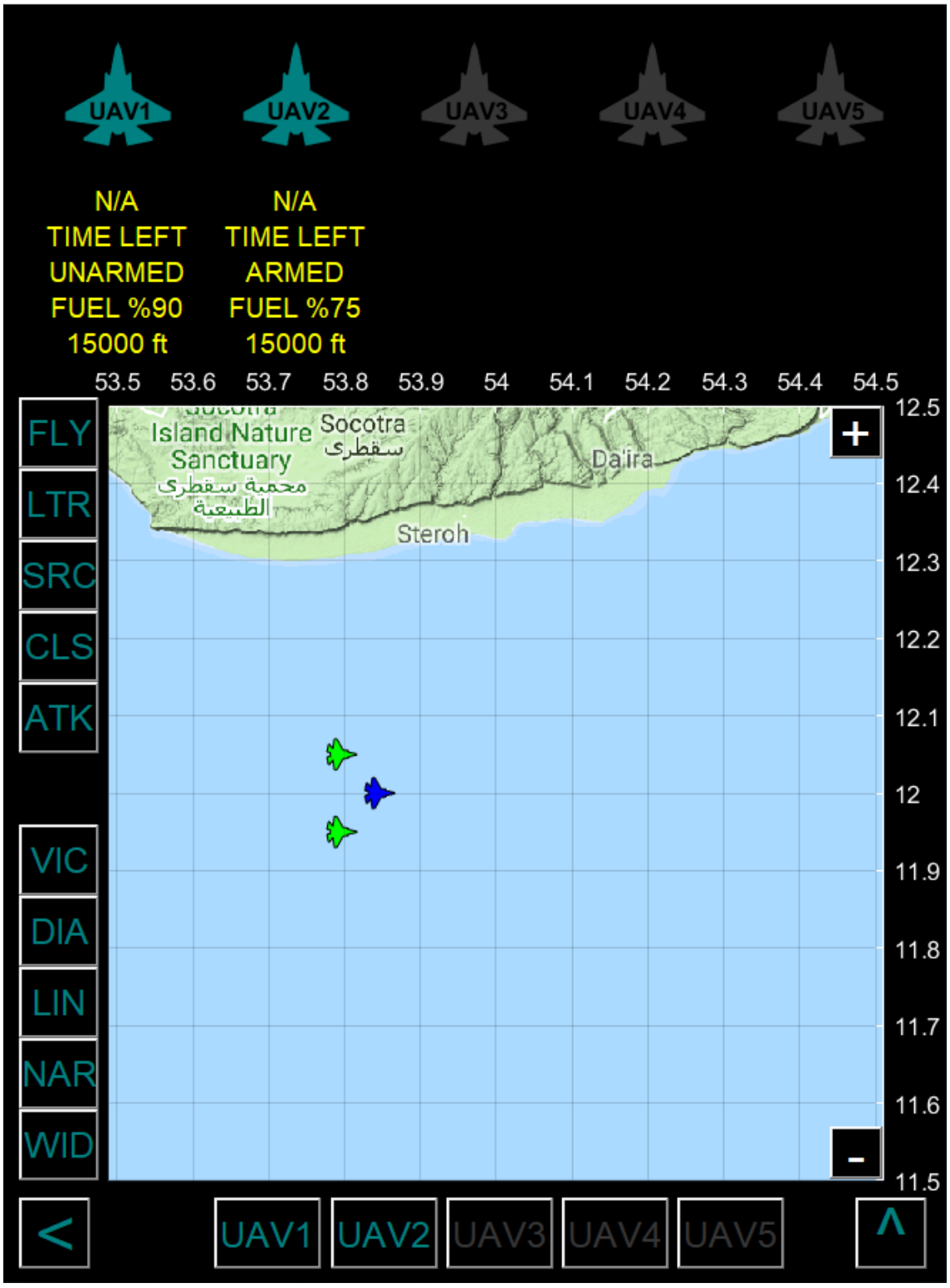


Figure 4-2 MMI Design 5 x 7 Inches

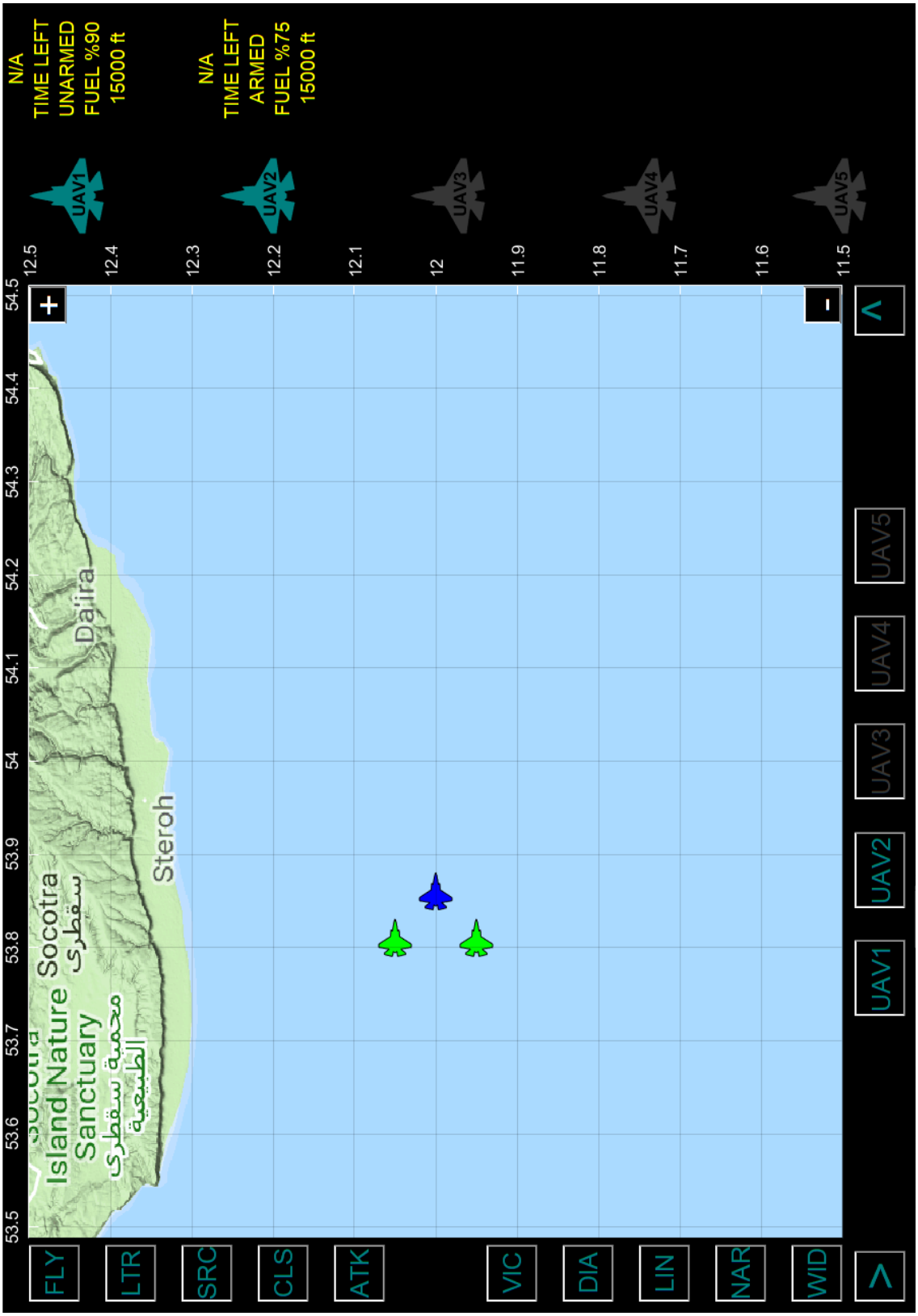


Figure 4-3 MMI Design 10 x 7 Inches

Colour preference in the MMI is also another assumption. Colour coding, which is commonly used in military displays, is a useful method to increase pilot situational awareness. Using the same colours for the same kind of icons or texts provides fast recognition based on human cognitive behaviour and decreases reaction time. The meaning of the colours in the suggested design are as follows:


Leader aircraft


Friend unit / good information


Foe unit / bad information

Selected unit / neutral information


Active object or function


Passive object or function


Background

4.2 Sections

The suggested MMI design consists of three sections:

- Situational awareness screen
- Information display
- Buttons

4.2.1 Situational Awareness Screen

The situational awareness screen (SAS) is the main component of the interface design. All actions can be tracked on the SAS with a real coordinate system. It presents dynamic information from the operational environment, such as the position of friend and foe units and terrain features. In addition to its informative function, SAS is also used as a task planning tool. The area for a searching task or a target point can be selected on it. During the design of the SAS, I faced two issues: map dimension and type.

A 2D map was preferred over a 3D map in the SAS design because of its advantages in regard to tactical usage. 3D maps provide a better visualisation of the environment, but they are generally more useful for remotely controlled aircrafts. Although a 3D view gives more detail in a narrow area, a 2D view provides the complete picture of the battlefield area. Tactically, a pilot should be able to see a wide area to analyse the battlefield environment effectively. It is easiest to track both friend and foe units from a 2D map with a bird's eye view.

The radar screen is also useful for understanding what is happening around the pilot, but in the cockpit of the F-35, there is another radar display already, so a second radar screen would be redundant. In addition to these, UAV operations terrain features are important for task planning. Numerous types of maps can be used as an SAS background map, but the most useful one is a terrain map because of its cognisable colour contrast and detailed terrain view.

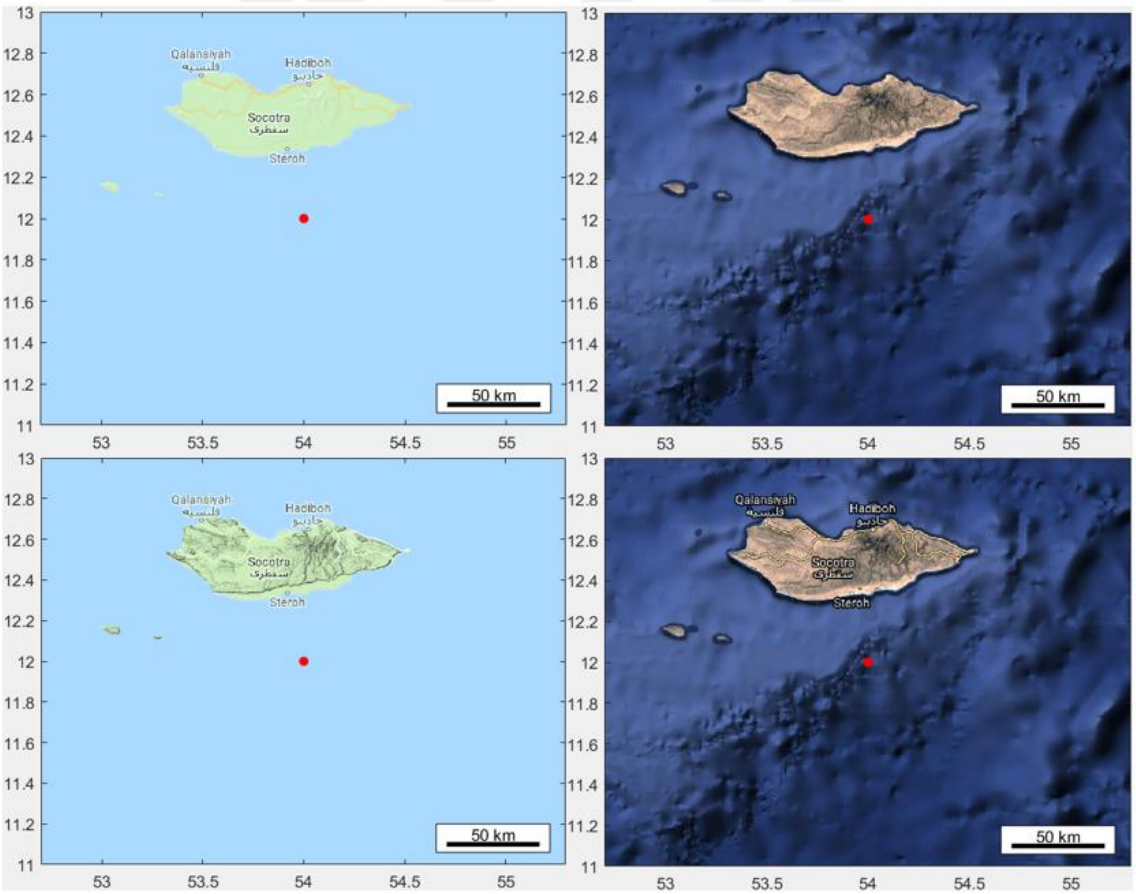


Figure 4-4 Map Types

4.2.2 Information Display

The information display is used to illustrate critical information related to the wingman status that cannot be presented on the SAS. Pilots need to know the current task and its remaining completion time because this information is useful for tracking the operational steps. If pilots know how much time is left to complete the task, they can prepare for the next move. Armament status is also important for wingman capability. Pilots need to know the armament status of the wingman before starting an attack mission. Due to the use of a 2D map display, the SAS does not show the altitude of wingmen, so altitude is given as complementary information. Finally, as safety-critical information, the fuel level is given on the information display. Figure 4-5 illustrates the design of an information display, illustrating up to five wingmen. Faded aircraft icons represent inactive wingmen, and two wingmen are controlled by the pilot.

Figure 4-5 Information Display

Because of the lack of space on the 5 by 5 inches display, the information screen is absent in this smallest size. Since the information screen just plays an informative role, this absence does not affect the functional performance of the interface. In the 5 by 7 inches design, the information display is placed at the top of the screen, and, in the 10 by 7 inches design, it is placed at the right-hand side. The top or right side of the screen is used because pilots must use their left hand to interact with the touch screen. As mentioned in Chapter 3, in military aircraft cockpits, the right hand is used for steering the aircraft, and the left hand interacts with buttons or the touch screen. Hence, in suggested MMI design, although

functional tools are located at the left and bottom, display tools are located at the right and top. Thus, the pilot's left hand never blocks the screen.

4.2.3 Buttons

Buttons can be classified into two groups. The first group involves interface management buttons, which consist of the size extension and zoom in and zoom out buttons. Although zoom in and zoom out buttons are placed at the top and bottom right-hand corners of the SAS, the extension buttons are placed at the bottom corners of the interface. Because, zoom in and zoom out buttons functional in SAS.

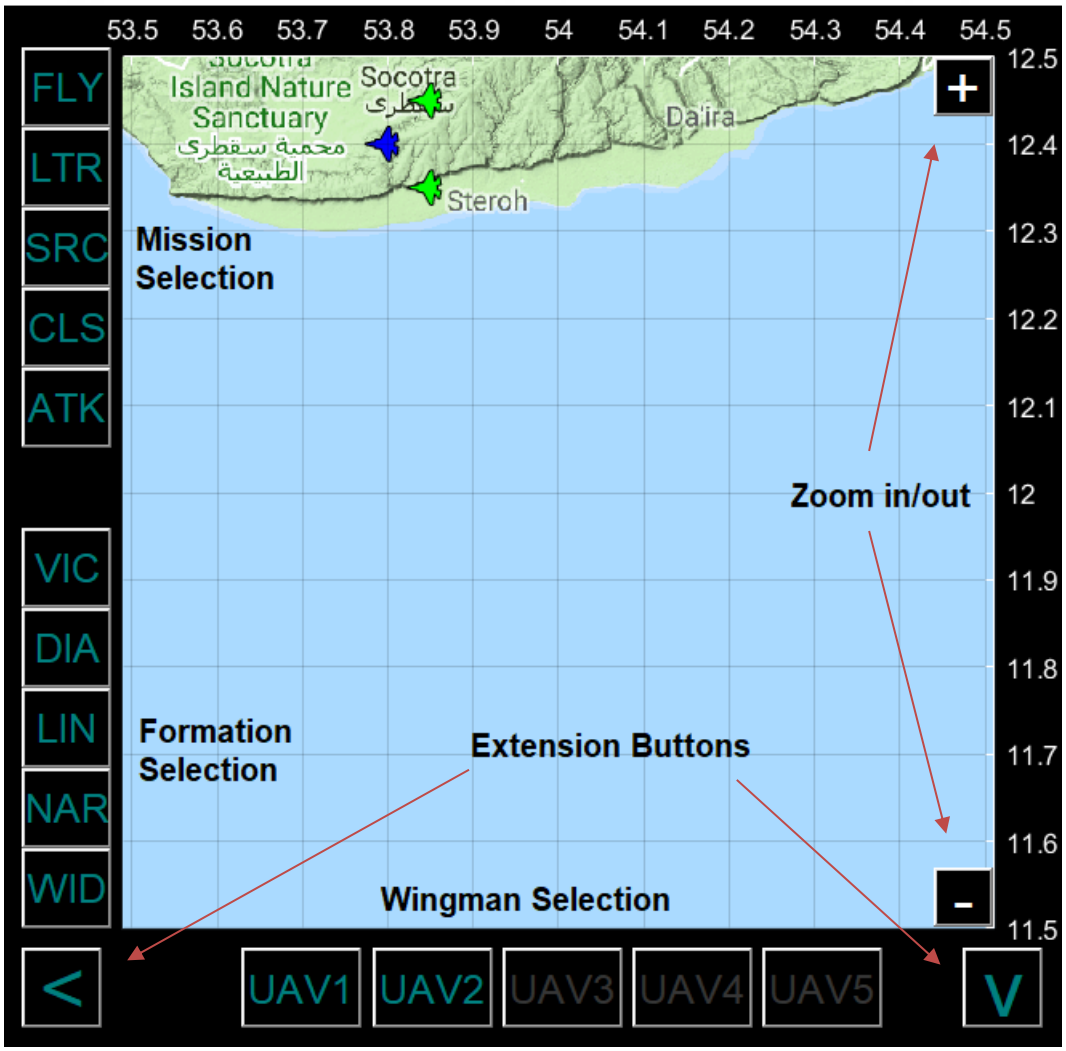


Figure 4-6 Buttons

The second group is the commanding buttons. The mission selection, formation selection and wingman selection buttons belong to this group. Grouping the buttons according to their functions makes interface usage easier. Hence, the mission selection buttons are located at the top-left corner, the formation selection buttons are located at the bottom-left corner and the wingman selection buttons are located at the bottom.

For optimal touch performance, the size of the commanding buttons is 10 by 15 mm, and interface management buttons is 10 by 10 mm based on the discussion in Chapter 3.4. In addition, the names of the functions are abbreviated to the 3-letter form to inform the pilot about the button's function. Colour codes are also used for buttons. When a button is touched, its colour turns to yellow to show an active selection.

4.3 Simulation

In this section, an example mission is simulated to explain the command and control procedure of the suggested MMI design. Five mission options are presented to the pilot as follows:

- FLY : Fly to X
- LTR : Loiter on X
- SRC : Search Area
- CLS : Classify X
- ATK : Attack to X

Also, five formation options are presented to the pilot:

- VIC : Vic Formation
- DIA : Diamond Formation
- LIN : Line Formation
- NAR : Become Narrow
- WID : Become Wide

In the simulation, a leader and two wingmen fly in Vic formation, and the pilot commands a search mission. One wingman alone or two wingmen together can

perform this search mission. Furthermore, two separate search functions can be maintained at the same time if the pilot demands it.

The commanding procedure is explained by the following steps:

1) The pilot selects one of the wingmen or both together. If the pilot does not select a wingman, the MMI will automatically select the closest wingman in the next step.

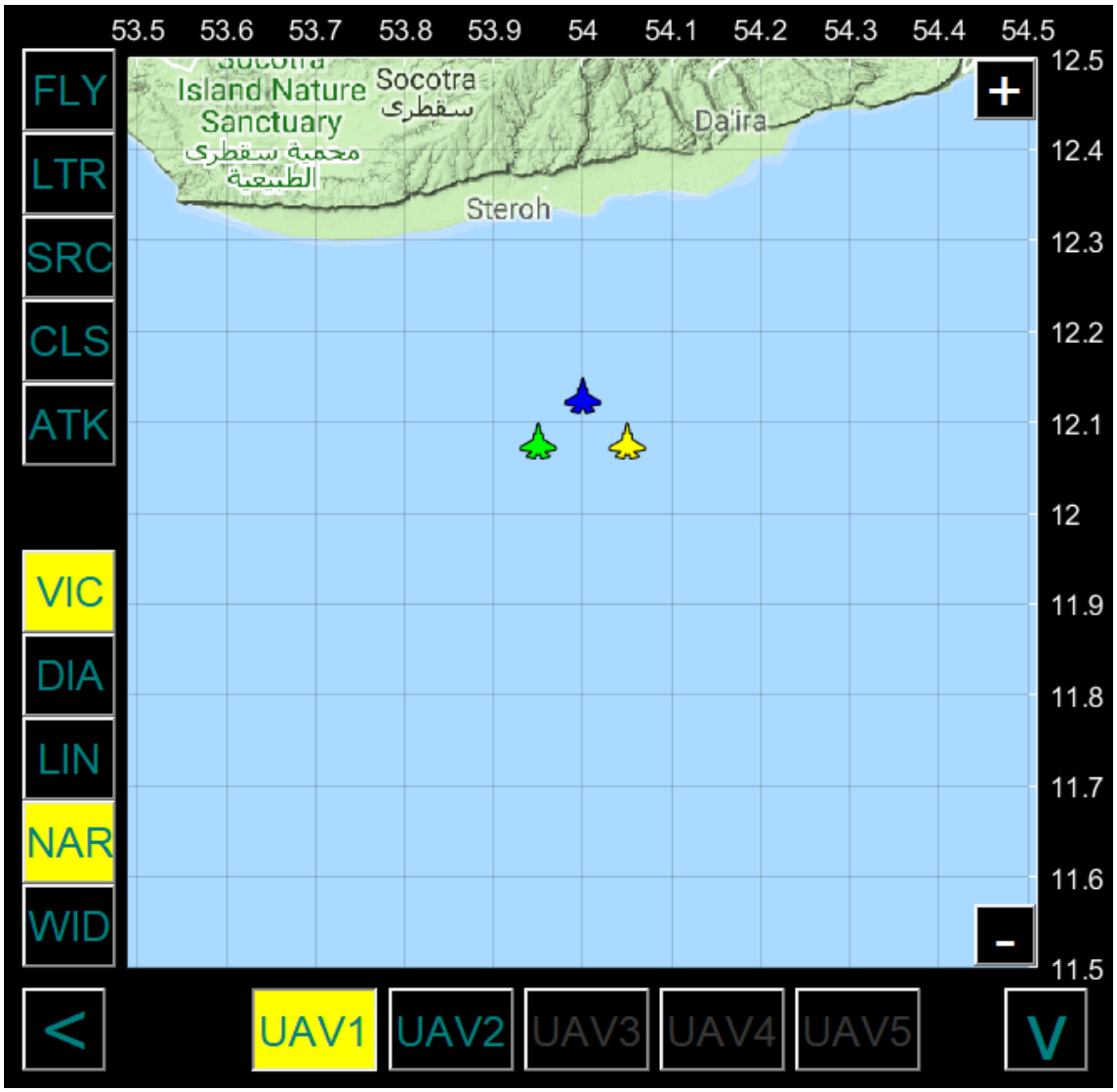


Figure 4-7 Simulation Step 1

After selecting the wingman or wingmen, both the selection button and wingman icon in the SAS become yellow, which represents a selected unit according to the colour code. The VIC and NAR formation buttons are also yellow at this time.

2) The pilot selects a mission from the mission selection menu. After selecting a mission, the screen is activated to select a target point or area according to the mission type. For the area search mission, the MMI allows pilots to draw a rectangle with their finger, which gives them the freedom to define the area size.

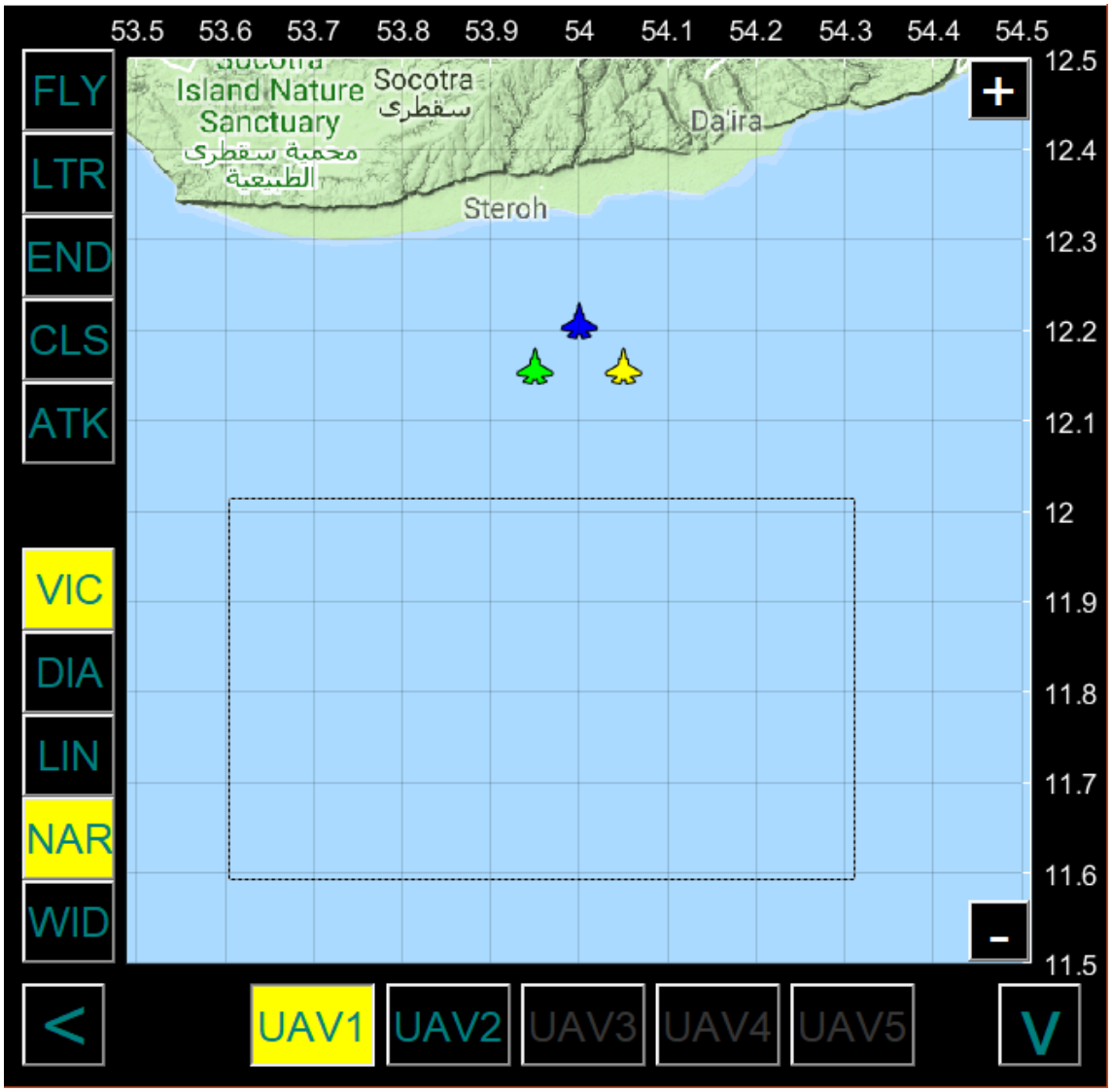


Figure 4-8 Simulation Step 2

After defining the area, the wingman is authorised to execute a search mission in the defined area.

3) The wingman calculates the optimal flight path to search the defined area and leaves the team to complete its mission. The information screen also illustrates the current mission and remaining time information.

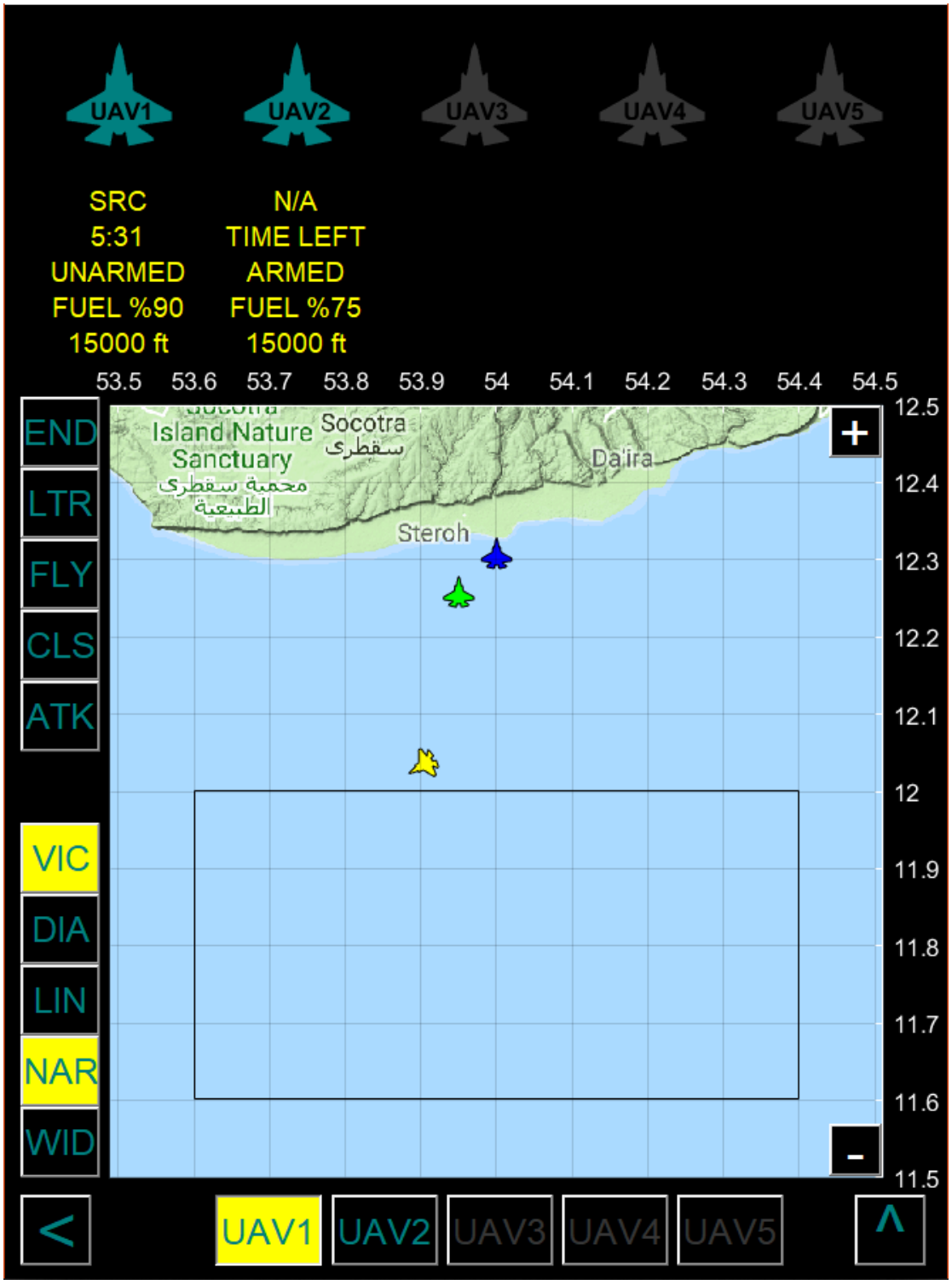


Figure 4-9 Simulation Step 3

4) The wingman searches the defined area. After searching, it returns to the team.

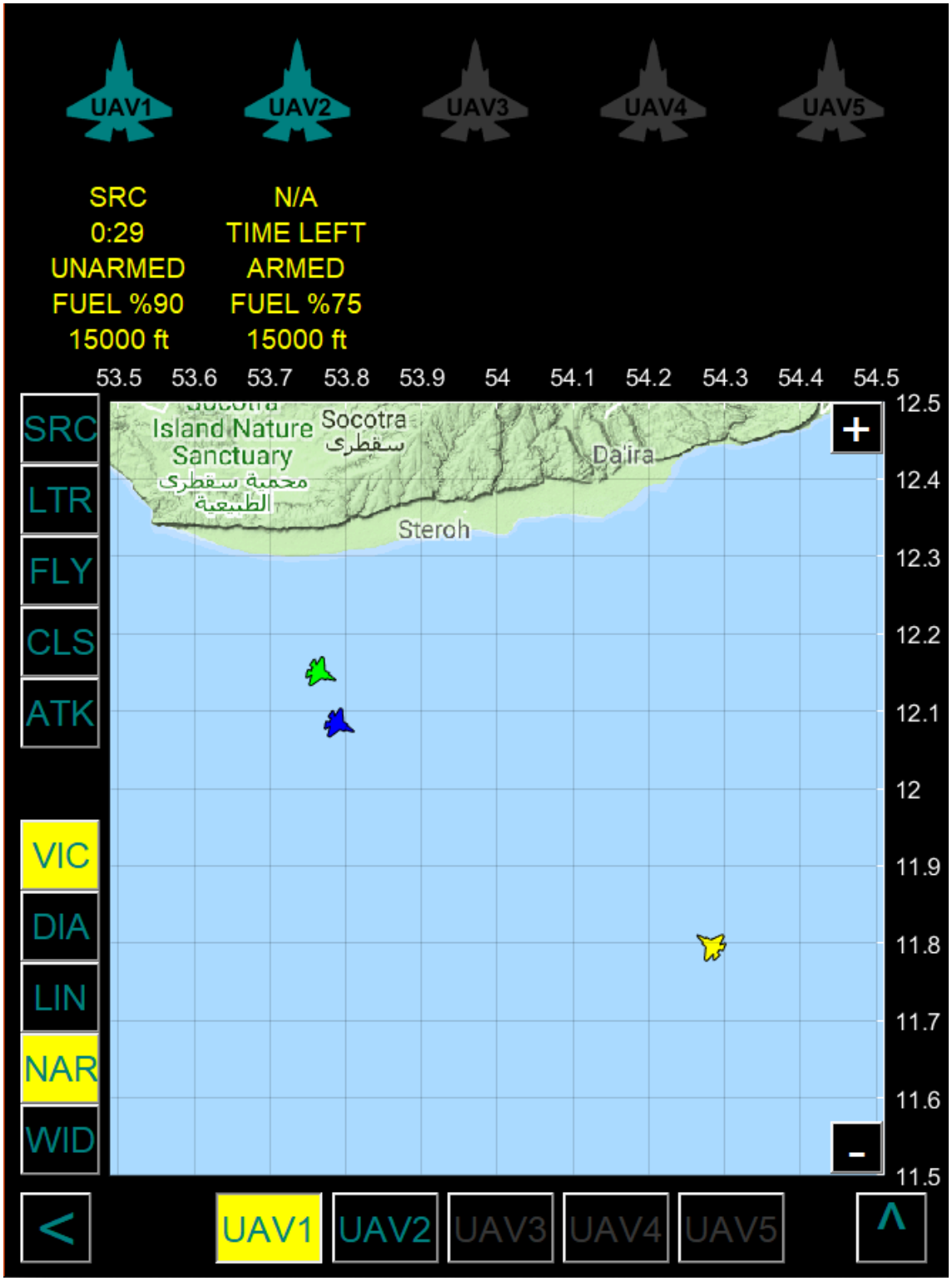


Figure 4-10 Simulation Step 4

4.4 Discussion of Design Parameters

Although there are some certain limitations and assumptions, a lot of design parameters have more than one option to obtain an optimal design. Hence, optimality of design varies according to designer perspective and preferences. In Table 4-1, design preferences are assessed with comparing other alternatives.

Parameter	Preference	Advantage	Alternatives
Button Sizes	Minimum possible sizes were chosen: 10x10 mm for interface management buttons. 10x15 mm for commanding buttons	- Provide effective usage of limited space of interface screen. - Provide acceptable performance based on reaction time and error.	- Larger buttons may provide better touch performance - Smaller buttons may provide more empty space for larger screen or information display.
Colour	Colour Coding	- Provide easy recognition and comprehension	- Colour preferences may be changed but it has no significant effect on performance
Map Type	Terrain Map	- Provide better contrast to recognize terrain features.	- Satellite map provides more real view, but it can be confusing for pilot - Radar map is good for tactical decisions but cannot present terrain features.
Dimensions	2D	- 2D map provides better tactical and general view in large area.	- 3D map may be preferred for more detailed view for a narrow area.
Number of UAV	5	- This is the maximum number to present data with selected size preferences.	- With decreasing the size of some components, it may be possible to add some extra UAV slots.
Mission Definition	Pre-defined	- Provide quick selection but mission variety is limited	- Adding a pop-up menu for mission selection may be a solution to increase mission variety.
Level of Autonomy	PACT Level 3	- Pilot gets enough assistance on routine tasks and stay in the control loop for critical decisions	- With increasing autonomy pilot's workload can be decreased but increasing unpredictability causes design challenges

Table 4-1 Design Preferences

5 Conclusion and Recommendation

Due to research and developments in fighter jets and unmanned systems, MUM teaming has begun to gain significance for military applications. Rather than tens of manned aircrafts, just a few manned aircrafts will be able to command and control a huge squadron in the near future. According to General Pawlikowski (Clark, 2014) from the US Air Force, F-35 fighter jets can form a team with 20 remotely piloted aircraft with their technological infrastructure. Furthermore, the upcoming sixth-generation fighter jets will likely become completely autonomous. It means importance of MMI design maintain its increase in next years.

In this thesis, the factors that influence MMI design have been assessed, and the design criteria for an MMI design have been specified based on the F-35 fifth-generation fighter jet. Autonomy and communication issues are very effective on MMI design. As it assessed in Chapter 2, both increases and decreases in level of autonomy cause inconvenience in design process. Because increasing autonomy increases unpredictability and decreasing autonomy increases workload of the pilot. Again, from Chapter 2 communication is also crucial and limitations are bandwidth, LOS range and amount of data.

In Chapter 3, I examined a real MMI design from a fifth generation fighter jet airplane and also presented some design criteria both from F-35 and literature. It can be said that the most crucial design criteria are interaction requirements and information exchange requirement between pilot and interface. Furthermore cognitive factors, such as button size, screen features and information presenting style, is also effective on physical design.

According to Chapter 2 and 3 a suggested design has presented in Chapter 4. Design was arranged for a maximum of five wingmen. Although some of the design criteria are subjective, a near optimal design has been suggested that considers simplicity and functionality. A discussion about advantages and disadvantages of chosen design parameters was given at the end of the Chapter 4.

Future works can focus on following ideas:

- Design can be adjustable for different number of wingmen and different number of missions. Because the number of unmanned system will increase in the future
- To understand the optimality level of design, a questionnaire can be arranged based on different design criteria
- F-35 fighter jet has already a situational awareness so merging main situational awareness screen with our wingmen commanding interface may be a good idea for functionality.
- Voice recognition can be added to MMI design as an input tool.

REFERENCES

Clark, C (2014) 'Pawlikowski on Air Force offset strategy: F-35s flying drone fleets', *Breaking Defense.* Available at: https://breakingdefense.com/2014/12/pawlikowski-on-air-force-offset-strategy-f-35s-flying-drone-fleets/ (Accessed: 9 August 2018).

Conradi, J., Busch, O. and Alexander, T. (2015) 'Optimal touch button size for the use of mobile devices while walking', *Procedia Manufacturing*, 3 (Ahfe) Elsevier B.V., pp. 387–394.

Degarmo, M.T. (2004) *Issues concerning integration of unmanned aerial vehicles in civil airspace*.

Derefeldt, G., Skinnars, Ö., Alfredson, J., Eriksson, L., Andersson, P., Westlund, J., Berggrund, U., Holmberg, J. and Santesson, R. (1999) 'Improvement of tactical situation awareness with colour-coded horizontal-situation displays in combat aircraft', *Displays.*

Endsley, M.R. (1995) 'Toward a theory of situation awareness in dynamic systems', *Human Factors: The Journal of the Human Factors and Ergonomics Society.*

Golliday, C. (1985) 'Data link communications in tactical air command and control systems', *IEEE Journal on Selected Areas in Communications*, 3(5), pp. 779–791.

Goraj, Z. (2005) 'Design challenges associated with development of a new generation UAV', Aircraft Engineering and Aerospace Technology, 77(5), pp. 361–368.

Howitt, S.L. and Platts, J.T. (2002) 'Real-time deep strike mission simulation using air-launched UAVs', AIAA's 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles. Virginia, pp. 1–10. Available at: 10.2514/6.2002-3480 (Accessed: 16 August 2018).

Howitt, S.L. and Richards, D. (2003) 'The human machine interface for airborne

control of UAVs', 2nd AIAA Unmanned Systems, Technologies, and Operations - Aerospace, Land, and Sea Conference and Workshop, (September), pp. 1–10.

JP1-02 DoD (2001) *JP 1-02, Department of Defense Dictionary of Military and Associated Terms, Department of Defense Dictionary of Military and Associated Terms* - Available at: http://www.bits.de/NRANEU/others/jp-doctrine/jp1_02%2805%29.pdf (Accessed: 3 August 2018).

Lewis, R. (2015) '*The effect of handedness on use of touch screen versus touch pad*', *ATACCS*. Toulouse, pp. 115–120.

Longman Dictionary of Contemporary English. https://www.ldoceonline.com Accessed: 2018-08-22

18-05-2014.Miller, C., Pelican, M. and Goldman, R. (2000) ''Tasking interfaces for flexible interaction with automation: Keeping the operator in control', *Conference on Human Interaction with Complex Systems*,(1), Redondo Beach, CA, pp. 123–128.

Miller, C.A. and Parasuraman, R. (2007) 'Designing for flexible interaction between humans and automation: Delegation interfaces for supervisory control', *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 49(1), pp. 57–75.

North, G. (2016) *Proud to support the IAF mission since 1948*. Available at: http://www.fisher.org.il/2016/Adir Powepoint/GaryNorth.pdf#page=7 (Accessed: 9 August 2018).

Northrop Grumman (2014) '*Understanding voice and data link networking*', (December)

Office of the Secretary of Defense (2005) *Unmanned aircraft systems roadmap*, Available at: https://fas.org/irp/program/collect/uav_roadmap2005.pdf (Accessed: 13 August 2018).

Parasuraman, R., Sheridan, T.B. and Wickens, C.D. (2000) 'A model for types and levels of human interaction', *IEEE Transactions on Systems Man and*

Cybernetics Part A Systems and Humans, 30(3), pp. 286–297.

Plessas, D. (2017) Transitioning from 4th to 5th generation fighter aircraft innovation and rationalization in the new economic environment. Available at: https://www.aerosociety.com/media/7520/9-plessas-transitioning-from-4th-to-5th-generation-fighter-aircraft.pdf (Accessed: 9 August 2018).

Popov, V.L., Shiev, K.B., Topalov, A.V., Shakev, N.G. and Ahmed, S.A. (2016) 'Control of the flight of a small quadrotor using gestural interface', *2016 IEEE 8th International Conference on Intelligent Systems, IS 2016 - Proceedings*, Sofia, pp. 622–628.

Rasmussen, J. (1976) 'Outlines of a hybrid model of the process plant operator', Monitoring Behavior and Supervisory Control, NATO Conference Series, vol 1. Springer, Boston, MA pp. 371–383.

Ruiz, J.J., Escalera, M.A., Viguria, A. and Ollero, A. (2016) 'A simulation framework to validate the use of head-mounted displays and tablets for information exchange with the UAV safety pilot', *2015 Workshop on Research', Education and Development of Unmanned Aerial Systems, RED-UAS 2015*, Cancun, pp. 336–341.

Schedlbauer, M. (2007) *Effects of key size and spacing on the completion time and accuracy of input tasks on soft keypads using trackball and touch input*. Available at: http://www.cs.uml.edu/~mschedlb/mte. (Accessed: 17 August 2018).

Sheridan, T.B. and Verplank, W.L. (1978) 'Human and computer control of undersea teleoperators', *ManMachine Systems Lab Department of Mechanical Engineering MIT Grant N0001477C0256.*

Skaff, M. (2010) 'F-35 lightning II cockpit vision', *SAE International J. Passeng. Cars – Electron. Electr. Syst.*

Sorroche, J. (2012) 'Modeling tactical data links', *Engineering Principles of Combat Modeling and Distributed Simulation*, pp. 537–578.

Soto-Gerrero, J. and Ramrez-Torres, G. (2016) ‘A Human-Machine Interface with Unmanned Aerial Vehicles’, 37, pp. 233–242.

Stoica, A., Militaru, D., Moldoveanu, D. and Popa, A. (2016) ‘Tactical data link – From link 1 To link 22’, Scientific Bulletin of Naval Academy, 19(2), pp. 317–322.

Taylor, R.M. (2001) ‘Cognitive cockpit systems engineering: pilot authorisation and control of tasks’, *CSAPC’01 Proceedings of the 8th Conference on Cognitive Science Approaches to Process Control*. , Farnborough.

Tesla, N. (1898) ‘Method of and apparatus for controlling mechanism of moving vessels or vehicles’, *Google Patents*, (68), pp. 1–9.

White, A.D. (2003) ‘The human-machine partnership in UCAV operations’, Aeronautical Journal, 107(1068 SPEC.), pp. 111–116.

Winnefeld, J.A. and Kendall, F. (2010) ‘*Unmanned systems integrated roadmap: FY 2011-2036*’, pp. 1–74.

Wordless Tech (2012) Lockheed Martin’s F-35 Lightning II most advanced cockpit | wordlessTech. Available at: https://wordlesstech.com/lockheed-martins-f-35-lightning-ii-most-advanced-cockpit/#post/0 (Accessed: 17 August 2018).

Zhao, C., Kou, M. and Wu, J. (2012) ‘Task allocation for integrated tactical data links’, *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, Williamsburg pp. 14–18.

APPENDICES

Appendix A MATLAB Codes (Use Heading 7)

```
function OKI_UI()
clear all
clc;

% Figure definition
fig = figure('units','pixels',...
                'name','Wingmen Control Screen',...
                'menubar','none',...
                'numbertitle','off',...
                'position',[510 200 530 520],...
                'color','k',...
                'busyaction','cancel',...
                'renderer','opengl');

% UAV1 Button
btn_uav1 = uicontrol('Style',
'togglebutton','FontUnits','pixels',...

'FontSize',20,'String','UAV1','Units','pixel','Position',[120 10
60 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@selection_uav1);

%UAV2 Button
btn_uav2 = uicontrol('Style',
'togglebutton','FontUnits','pixels',...

'FontSize',20,'String','UAV2','Units','pixel','Position',[185 10
60 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@selection_uav2);

%UAV3 Button
btn_uav3 = uicontrol('Style',
'togglebutton','FontUnits','pixels',...

'FontSize',20,'String','UAV3','Units','pixel','Position',[250 10
60 40],...

'BackgroundColor','k','ForegroundColor',[0.211,0.211,0.211],'Cal
lback', @selection_uav3);

%UAV4 Button
btn_uav4 = uicontrol('Style',
'togglebutton','FontUnits','pixels',...
```

```
'FontSize',20,'String','UAV4','Units','pixel','Position',[315 10
60 40],...

'BackgroundColor','k','ForegroundColor',[0.211,0.211,0.211],'Cal
lback', @selection_uav4);

%UAV5 Button
btn_uav5 = uicontrol('Style',
'togglebutton','FontUnits','pixels',...

'FontSize',20,'String','UAV5','Units','pixel','Position',[380 10
60 40],...

'BackgroundColor','k','ForegroundColor',[0.211,0.211,0.211],'Cal
lback', @selection_uav5);

% Extension Button 1
btn_extension1 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',40,'String','v','Units','pixel','Position',[480
10 40 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@extension1);

% Extension Button 2
btn_extension2 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',40,'String','<','Units','pixel','Position',[10 10
40 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@extension2);

% Zoom in button
btn_zoom_in = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',30,'String','+','Units','pixel','Position',[465
465 30 30],...

'BackgroundColor','k','ForegroundColor','w','Callback',@zoom_in)
;

% Zoom out button
btn_zoom_out = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',30,'String','-','Units','pixel','Position',[465
60 30 30],...

'BackgroundColor','k','ForegroundColor','w','Callback',@zoom_out
);
```

```
% Mission Button 1
btn_mission1 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','SRC','Units','pixel','Position',[10
380 45 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@mission_ss);

% Mission Button 2
btn_mission2 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','LTR','Units','pixel','Position',[10
420 45 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@mission_ss);

% Mission Button 3
btn_mission3 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','FLY','Units','pixel','Position',[10
460 45 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@mission_ss);

% Mission Button 4
btn_mission4 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','CLS','Units','pixel','Position',[10
340 45 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@mission_ss);

% Mission Button 5
btn_mission5 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','ATK','Units','pixel','Position',[10
300 45 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@mission_ss);

% Formation Button 1
btn_formation1 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','WID','Units','pixel','Position',[10
60 45 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@formation_ss);
```

```
% Formation Button 2
btn_formation2 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','NAR','Units','pixel','Position',[10
100 45 40],...

'BackgroundColor','y','ForegroundColor',[0,0.5,0.5],'Callback',
@formation_ss);

% Formation Button 3
btn_formation3 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','LIN','Units','pixel','Position',[10
140 45 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@formation_ss);

% Formation Button 4
btn_formation4 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','DIA','Units','pixel','Position',[10
180 45 40],...

'BackgroundColor','k','ForegroundColor',[0,0.5,0.5],'Callback',
@formation_ss);

% Formation Button 5
btn_formation5 = uicontrol('Style',
'pushbutton','FontUnits','pixels',...
    'FontSize',20,'String','VIC','Units','pixel','Position',[10
220 45 40],...

'BackgroundColor','y','ForegroundColor',[0,0.5,0.5],'Callback',
@formation_ss);


%% Initials
global c path_ac angle_ac
focus=1;
uav=0;
shf=0;
m=0;
s=0.001;
r=0.025;
ac_lon = 54;
ac_lat = 12;
ang_ac = pi/2;
pos_ac = [ac_lon,ac_lat];
pos_uav1 = [0 0];
ang_uav1 = 0;
pos_uav2=[0 0];
```

```
ang_uav2 = 0;
W1=[pos_ac+[0 r],ang_ac];

act_pos = pos_ac; % Active central position

% Main Screen
screen_h = axes('Units','pixel','Position',[60 60 435
435],'XAxisLocation','top',...

'XColor','w','YAxisLocation','right','Ycolor','w',...
                'XGrid','on','YGrid','on','GridColor','k');
pan on

% Aircraft figure

X_image = [-163 -163 -150 -209 -187 -171 -125 -105 -68  20  84
98  80  187 220 ;...
        0  -11  -19  -28  -117 -117 -63  -178 -178 -49 -46 -41 -
29 -12 0 ]';

leng_X = length(X_image(:,1)) ;

for i=1:leng_X
    X_image(leng_X+i,1) = X_image(leng_X-i+1,1);
    X_image(leng_X+i,2) = -X_image(leng_X-i+1,2);
end

X_image(:,1) = X_image(:,1)*0.04/( max(X_image(:,1))-
min(X_image(:,1)) ) ;
X_image(:,2) = X_image(:,2)*0.04/( max(X_image(:,2))-
min(X_image(:,2)) ) ;

% Handles
main_ac=hgtransform;
patch('XData',X_image(:,1),'YData',X_image(:,2),'FaceColor','b',
'Parent',main_ac,'ButtonDownFcn',@selection_ac); %Main_AC
uav_1=hgtransform;
patch('XData',X_image(:,1),'YData',X_image(:,2),'FaceColor','g',
'Parent',uav_1,'ButtonDownFcn',@selection_uav1); %UAV 1
uav_2=hgtransform;
patch('XData',X_image(:,1),'YData',X_image(:,2),'FaceColor','g',
'Parent',uav_2,'ButtonDownFcn',@selection_uav2); %UAV2

rect = rectangle('ButtonDownFcn',@re_size);

%% Information screens
        %Display1
        axes_1 = axes('Units','pixel','Position',[15 620 100
100],...
                'color','k','XColor','k','Ycolor','k');
        uav_1_im=hgtransform;
```

```
patch('XData',X_image(:,1),'YData',X_image(:,2),'FaceColor',[0,.
5,.5],'Parent',uav_1_im);
        uav_1_im.Matrix =
makehgtform('zrotate',pi/2,'scale',1.5);
        text(-.016,-
.008,'UAV1','FontWeight','bold','Color','k');

        % Display2
        axes_2 = axes('Units','pixel','Position',[115 620 100
100],...
                'color','k','XColor','k','Ycolor','k');
        uav_2_im=hgtransform;

patch('XData',X_image(:,1),'YData',X_image(:,2),'FaceColor',[0,.
5,.5],'Parent',uav_2_im);
        uav_2_im.Matrix =
makehgtform('zrotate',pi/2,'scale',1.5);
        text(-.016,-
.008,'UAV2','FontWeight','bold','Color','k');

        % Display3
        axes_3 = axes('Units','pixel','Position',[215 620 100
100],...
                'color','k','XColor','k','Ycolor','k');
        uav_3_im=hgtransform;

patch('XData',X_image(:,1),'YData',X_image(:,2),'FaceColor',[0.2
11,0.211,0.211],'Parent',uav_3_im);
        uav_3_im.Matrix =
makehgtform('zrotate',pi/2,'scale',1.5);
        text(-.016,-
.008,'UAV3','FontWeight','bold','Color','k');

        % Display4
        axes_4 = axes('Units','pixel','Position',[315 620 100
100],...
                'color','k','XColor','k','Ycolor','k');
        uav_4_im=hgtransform;

patch('XData',X_image(:,1),'YData',X_image(:,2),'FaceColor',[0.2
11,0.211,0.211],'Parent',uav_4_im);
        uav_4_im.Matrix =
makehgtform('zrotate',pi/2,'scale',1.5);
        text(-.016,-
.008,'UAV4','FontWeight','bold','Color','k');

        % Display5
        axes_5 = axes('Units','pixel','Position',[415 620 100
100],...
                'color','k','XColor','k','Ycolor','k');
        uav_5_im=hgtransform;
```

```
patch('XData',X_image(:,1),'YData',X_image(:,2),'FaceColor',[0.2
11,0.211,0.211],'Parent',uav_5_im);
        uav_5_im.Matrix =
makehgtform('zrotate',pi/2,'scale',1.5);
        text(-.016,-
.008,'UAV5','FontWeight','bold','Color','k');

        info11 = uicontrol('Style',
'text','FontUnits','pixels',...

'FontSize',15,'String','SRC','Units','pixel','Position',[15 600
100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info12 = uicontrol('Style',
'text','FontUnits','pixels',...

'FontSize',15,'String','0:29','Units','pixel','Position',[15 580
100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info13 = uicontrol('Style',
'text','FontUnits','pixels',...

'FontSize',15,'String','UNARMED','Units','pixel','Position',[15
560 100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info14 = uicontrol('Style',
'text','FontUnits','pixels',...
        'FontSize',15,'String','FUEL
%90','Units','pixel','Position',[15 540 100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info15 = uicontrol('Style',
'text','FontUnits','pixels',...
        'FontSize',15,'String','15000
ft','Units','pixel','Position',[15 520 100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info21 = uicontrol('Style',
'text','FontUnits','pixels',...
```

```
'FontSize',15,'String','N/A','Units','pixel','Position',[115 600
100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info22 = uicontrol('Style',
'text','FontUnits','pixels',...
        'FontSize',15,'String','TIME
LEFT','Units','pixel','Position',[115 580 100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info23 = uicontrol('Style',
'text','FontUnits','pixels',...

'FontSize',15,'String','ARMED','Units','pixel','Position',[115
560 100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info24 = uicontrol('Style',
'text','FontUnits','pixels',...
        'FontSize',15,'String','FUEL
%75','Units','pixel','Position',[115 540 100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

        info25 = uicontrol('Style',
'text','FontUnits','pixels',...
        'FontSize',15,'String','15000
ft','Units','pixel','Position',[115 520 100 20],...

'HorizontalAlignment','center','BackgroundColor','k','Foreground
Color','y');

%% Simulation
axes(screen_h);

map_update;

free_mission;

    function map_update(varargin)

        lon_min = -0.5*focus+act_pos(1);
        lon_max =  0.5*focus+act_pos(1);
        lat_min = -0.5*focus+act_pos(2);
        lat_max =  0.5*focus+act_pos(2);
```

```
        axis([lon_min lon_max lat_min lat_max]);

% plot_google_map.m , Created by Zohar Bar-Yehuda, availlable
at: http://www.mathworks.com/matlabcentral/fileexchange/24113
        p_map = plot_google_map('MapType','terrain');

    end

    function mission_ss(varargin)
        switch get(btn_mission1,'string')
            case 'SRC'
                set(btn_mission1,'string','END')
                m=1;
                mission
            case 'END'
                set(btn_mission1,'string','SRC')
                delete(rect)
                m=0;
                free_mission
        end
    end

    function free_mission(varargin)
        %Way points
        W1=[pos_ac+[0 r],ang_ac];
        W2=[pos_ac+[0-r 0.4],ang_ac+pi/2];
        W3=[pos_ac+[-0.4 0.4-r],ang_ac+2*pi/2];
        W4=[pos_ac+[-0.4+r 0],ang_ac+3*pi/2];

        %Path
% dubins_curve.m , Created by Ewing Kang, availlable at:
https://uk.mathworks.com/matlabcentral/fileexchange/63308-ewingkang-
dubins-curve-for-matlab

        path_ac_1 = dubins_curve(W1,W2,r,s,'quiet');
        path_ac_2 = dubins_curve(W2,W3,r,s,'quiet');
        path_ac_3 = dubins_curve(W3,W4,r,s,'quiet');
        path_ac_4 = dubins_curve(W4,W1,r,s,'quiet');
        path_ac =
[path_ac_1(:,1:2);path_ac_2(:,1:2);path_ac_3(:,1:2);path_ac_4(:,
1:2)];
        angle_ac =
[path_ac_1(:,3);path_ac_2(:,3);path_ac_3(:,3);path_ac_4(:,3)];

        c=1;
        while c<=length(path_ac)
            main_ac.Matrix =
makehgtform('translate',[path_ac(c,:),0],'zrotate',angle_ac(c),'
scale',focus);
            uav_1.Matrix =
makehgtform('translate',[path_ac(c,:),0]+([cos(angle_ac(c)) -
sin(angle_ac(c)) 0;...
                sin(angle_ac(c)) cos(angle_ac(c)) 0;0 0 0]*[-
0.05,-0.05,0]')','zrotate',angle_ac(c),'scale',focus);
```

```
            uav_2.Matrix =
makehgtform('translate',[path_ac(c,:),0]+([cos(angle_ac(c)) -
sin(angle_ac(c)) 0;...
                sin(angle_ac(c)) cos(angle_ac(c)) 0;0 0 0]*[-
0.05, 0.05,0]')','zrotate',angle_ac(c),'scale',focus);
            pos_ac = path_ac(c,:);
            ang_ac = angle_ac(c);

            drawnow
            if m==1
                break
            end
            if c==length(path_ac)
                c=1;
            end
            c=c+1;
        end

    end

    function mission(varargin)
         tmp=getrect();
        set(rect,'Position',tmp);
        m=1;
        pan on

        switch uav
            case 1
            %Path UAV
            pos_uav1 = [pos_ac]+([cos(ang_ac) -sin(ang_ac) ;...
                    sin(ang_ac) cos(ang_ac)]*[-0.05,-0.05]')';
            ang_uav1 = ang_ac;

            U1 =[pos_uav1,ang_uav1];
            lx=rect.Position(3);
            ly=rect.Position(4);

            U2 = [rect.Position(1)+0.1, rect.Position(2)+ly-
0.1,-pi/2];
            path_uav1 = dubins_curve(U1,U2,r,s,'quiet');

            rep = round((lx-0.1)/0.2);
            U(1,:)=U2;
            cons = 0.2;
            for n=2:2:2*rep
                U(n,:)= [U(n-1,1),U(n-1,2)-ly+cons,U(n-1,3)];
                U(n+1,:)= [U(n,1)+0.2,U(n,2),-U(n,3)];
                cons=-cons;
                ly=-ly;
            end
            U(2*rep+2,:) = [U(2*rep+1,1),U(2*rep+1,2)-
ly+cons,U(2*rep+1,3)];
            for k=1:length(U)-1
```

```
                path= dubins_curve(U(k,:),U(k+1,:),r,s,'quiet');
                path_uav1=[path_uav1;path];
            end


            cc=1;
            while c<=length(path_ac)

                uav_1.Matrix =
makehgtform('translate',[path_uav1(cc,1:2),0],'zrotate',path_uav
1(cc,3),'scale',focus);
                main_ac.Matrix =
makehgtform('translate',[path_ac(c,:),0],'zrotate',angle_ac(c),'
scale',focus);
                uav_2.Matrix =
makehgtform('translate',[path_ac(c,:),0]+([cos(angle_ac(c)) -
sin(angle_ac(c)) 0;...
                    sin(angle_ac(c)) cos(angle_ac(c)) 0;0 0
0]*[-0.05, 0.05,0]')','zrotate',angle_ac(c),'scale',focus);
                pos_ac = path_ac(c,:);
                ang_ac = angle_ac(c);
                drawnow
                if m==0
                    break
                end
                if c==length(path_ac)
                    c=1;
                end
                if cc==length(path_uav1)
                    delete(rect)
                    m=0;
                    set(btn_mission1,'string','SRC')
                    rect = rectangle('ButtonDownFcn',@re_size);
                    W0_ac=[pos_ac,ang_ac];
                    W0_uav1=path_uav1(cc,:);
                    ac_dis=norm(W1(1:2)-W0_ac(1:2),2);
                    uav1_dis=norm(W1(1:2)-W0_uav1(1:2),2);
                    path_ac_in =
dubins_curve(W0_ac,W1,r,s,'quiet');
                    path_uav_1 = dubins_curve(W0_uav1,W1+[0.05 -
0.05 0],r,s,'quiet');
                    path_uav1_in = dubins_curve(W0_uav1,W1+[0.05
-0.05 0],r,s*length(path_uav_1)/length(path_ac_in),'quiet');

ll=min(length(path_ac_in),length(path_uav1_in));
                    for l=1:ll
                        uav_1.Matrix =
makehgtform('translate',[path_uav1_in(l,1:2),0],'zrotate',path_u
av1_in(l,3),'scale',focus);
                        main_ac.Matrix =
makehgtform('translate',[path_ac_in(l,1:2),0],'zrotate',path_ac_
in(l,3),'scale',focus);
```

```
                        uav_2.Matrix =
makehgtform('translate',[path_ac_in(l,1:2),0]+([cos(path_ac_in(l
,3)) -sin(path_ac_in(l,3)) 0;...
                    sin(path_ac_in(l,3)) cos(path_ac_in(l,3))
0;0 0 0]*[-0.05,
0.05,0]')','zrotate',path_ac_in(l,3),'scale',focus);
                        drawnow
                    end
                    pos_ac = path_ac_in(ll,1:2);
                    ang_ac = path_ac_in(ll,3);
                    free_mission
                    break
                end
                c=c+1;
                cc=cc+1;
            end
            case 2
                %Path UAV
                pos_uav2 = [pos_ac]+([cos(ang_ac) -sin(ang_ac)
;...
                        sin(ang_ac) cos(ang_ac)]*[-
0.05,0.05]')';
                ang_uav2 = ang_ac;

                U1 =[pos_uav2,ang_uav2]
                lx=rect.Position(3);
                ly=rect.Position(4);

                U2 = [rect.Position(1)+0.1, rect.Position(2)+ly-
0.1,-pi/2]
                path_uav2 = dubins_curve(U1,U2,r,s,'quiet');

                rep = round((lx-0.1)/0.2);
                U(1,:)=U2;
                cons = 0.2;
                for n=2:2:2*rep
                    U(n,:)= [U(n-1,1),U(n-1,2)-ly+cons,U(n-
1,3)];
                    U(n+1,:)= [U(n,1)+0.2,U(n,2),-U(n,3)];
                    cons=-cons;
                    ly=-ly;
                end
                U(2*rep+2,:) = [U(2*rep+1,1),U(2*rep+1,2)-
ly+cons,U(2*rep+1,3)];
                for k=1:length(U)-1
                    path=
dubins_curve(U(k,:),U(k+1,:),r,s,'quiet');
                    path_uav2=[path_uav2;path];
                end


                cc=1;
                while c<=length(path_ac)
```

```
                    uav_2.Matrix =
makehgtform('translate',[path_uav2(cc,1:2),0],'zrotate',path_uav
2(cc,3),'scale',focus);
                    main_ac.Matrix =
makehgtform('translate',[path_ac(c,:),0],'zrotate',angle_ac(c),'
scale',focus);
                    uav_1.Matrix =
makehgtform('translate',[path_ac(c,:),0]+([cos(angle_ac(c)) -
sin(angle_ac(c)) 0;...
                        sin(angle_ac(c)) cos(angle_ac(c)) 0;0 0
0]*[-0.05, -0.05,0]')','zrotate',angle_ac(c),'scale',focus);
                    pos_ac = path_ac(c,:);
                    ang_ac = angle_ac(c);
                    drawnow
                    if m==0
                        break
                    end
                    if c==length(path_ac)
                        c=1;
                    end
                    if cc==length(path_uav2)
                        delete(rect)
                        m=0;
                        set(btn_mission1,'string','SRC')
                        rect =
rectangle('ButtonDownFcn',@re_size);
                        W0_ac=[pos_ac,ang_ac];
                        W0_uav2=path_uav2(cc,:);
                        ac_dis=norm(W1(1:2)-W0_ac(1:2),2);
                        uav1_dis=norm(W1(1:2)-W0_uav2(1:2),2);
                        path_ac_in =
dubins_curve(W0_ac,W1,r,s,'quiet');
                        path_uav_2 = dubins_curve(W0_uav2,W1+[-
0.05 -0.05 0],r,s,'quiet');
                        path_uav2_in =
dubins_curve(W0_uav2,W1+[-0.05 -0.05
0],r,s*length(path_uav_2)/length(path_ac_in),'quiet');

ll=min(length(path_ac_in),length(path_uav2_in));
                        for l=1:ll
                            uav_2.Matrix =
makehgtform('translate',[path_uav2_in(l,1:2),0],'zrotate',path_u
av2_in(l,3),'scale',focus);
                            main_ac.Matrix =
makehgtform('translate',[path_ac_in(l,1:2),0],'zrotate',path_ac_
in(l,3),'scale',focus);
                            uav_1.Matrix =
makehgtform('translate',[path_ac_in(l,1:2),0]+([cos(path_ac_in(l
,3)) -sin(path_ac_in(l,3)) 0;...
                        sin(path_ac_in(l,3))
cos(path_ac_in(l,3)) 0;0 0 0]*[-0.05, -
0.05,0]')','zrotate',path_ac_in(l,3),'scale',focus);
```

```
                        drawnow
                    end
                    pos_ac = path_ac_in(ll,1:2);
                    ang_ac = path_ac_in(ll,3);
                    free_mission
                    break
                end
                c=c+1;
                cc=cc+1;
            end
        case 3
            %Path UAV1
            pos_uav1 = [pos_ac]+([cos(ang_ac) -sin(ang_ac)
;...
                    sin(ang_ac) cos(ang_ac)]*[-0.05,-
0.05]')';
            ang_uav1 = ang_ac;

            U1 =[pos_uav1,ang_uav1];
            lx=rect.Position(3);
            ly=rect.Position(4);

            U2 = [rect.Position(1)+lx/2+0.1,
rect.Position(2)+ly-0.1,-pi/2];
            path_uav1 = dubins_curve(U1,U2,r,s,'quiet');
            l1=length(path_uav1);

            rep = round((lx-0.1)/0.2);
            U(1,:)=U2;
            cons = 0.2;
            for n=2:2:2*rep/2
                U(n,:)= [U(n-1,1),U(n-1,2)-ly+cons,U(n-
1,3)];
                U(n+1,:)= [U(n,1)+0.2,U(n,2),-U(n,3)];
                cons=-cons;
                ly=-ly;
            end
            U(end+1,:) = [U(end,1),U(end,2)-
ly+cons,U(end,3)];
            for k=1:length(U)-1
                path=
dubins_curve(U(k,:),U(k+1,:),r,s,'quiet');
                path_uav1=[path_uav1;path];
            end

            %Path UAV2
            pos_uav2 = [pos_ac]+([cos(ang_ac) -sin(ang_ac)
;...
                    sin(ang_ac) cos(ang_ac)]*[-
0.05,0.05]')';
            ang_uav2 = ang_ac;

            U11 =[pos_uav2,ang_uav2];
```

```
                lx=rect.Position(3);
                ly=rect.Position(4);

                U22 = [rect.Position(1)+0.1,
rect.Position(2)+ly-0.1,-pi/2];
                path_uav_2 = dubins_curve(U11,U22,r,s,'quiet');
                l2=length(path_uav_2);
                path_uav2 =
dubins_curve(U11,U22,r,s*l2/l1,'quiet');

                UU(1,:)=U22;
                cons = 0.2;
                for n=2:2:2*rep/2
                    UU(n,:)= [UU(n-1,1),UU(n-1,2)-ly+cons,UU(n-
1,3)];
                    UU(n+1,:)= [UU(n,1)+0.2,UU(n,2),-UU(n,3)];
                    cons=-cons;
                    ly=-ly;
                end

                UU(end+1,:) = [UU(end,1),UU(end,2)-
ly+cons,UU(end,3)];
                for k=1:length(UU)-1
                    path2=
dubins_curve(UU(k,:),UU(k+1,:),r,s,'quiet');
                    path_uav2=[path_uav2;path2];
                end
                l_min=min(length(path_uav1),length(path_uav2));
                cc=1;
            while c<=length(path_ac)

                uav_1.Matrix =
makehgtform('translate',[path_uav1(cc,1:2),0],'zrotate',path_uav
1(cc,3),'scale',focus);
                main_ac.Matrix =
makehgtform('translate',[path_ac(c,:),0],'zrotate',angle_ac(c),'
scale',focus);
                uav_2.Matrix =
makehgtform('translate',[path_uav2(cc,1:2),0],'zrotate',path_uav
2(cc,3),'scale',focus);

                pos_ac = path_ac(c,:);
                ang_ac = angle_ac(c);
                drawnow
                if m==0
                    break
                end
                if c==length(path_ac)
                    c=1;
                end
                if cc==l_min
                    delete(rect)
                    m=0;
```

```
                    set(btn_mission1,'string','SRC')
                    rect = rectangle('ButtonDownFcn',@re_size);
                    W0_ac=[pos_ac,ang_ac];
                    W0_uav1=path_uav1(cc,:);
                    W0_uav2=path_uav2(cc,:);
                    ac_dis=norm(W1(1:2)-W0_ac(1:2),2);
                    uav1_dis=norm(W1(1:2)-W0_uav1(1:2),2);
                    path_ac_in =
dubins_curve(W0_ac,W1,r,s,'quiet');
                    path_uav_1 = dubins_curve(W0_uav1,W1+[0.05 -
0.05 0],r,s,'quiet');
                    path_uav1_in = dubins_curve(W0_uav1,W1+[0.05
-0.05 0],r,s*length(path_uav_1)/length(path_ac_in),'quiet');
                    path_uav_2 = dubins_curve(W0_uav2,W1+[-0.05
-0.05 0],r,s,'quiet');
                    path_uav2_in = dubins_curve(W0_uav2,W1+[-
0.05 -0.05
0],r,s*length(path_uav_2)/length(path_ac_in),'quiet');

ll=min(length(path_uav1_in),length(path_uav2_in));
                    for l=1:ll
                        uav_1.Matrix =
makehgtform('translate',[path_uav1_in(l,1:2),0],'zrotate',path_u
av1_in(l,3),'scale',focus);
                        main_ac.Matrix =
makehgtform('translate',[path_ac_in(l,1:2),0],'zrotate',path_ac_
in(l,3),'scale',focus);
                        uav_2.Matrix =
makehgtform('translate',[path_uav2_in(l,1:2),0],'zrotate',path_u
av2_in(l,3),'scale',focus);

                        drawnow
                    end
                    pos_ac = path_ac_in(ll,1:2);
                    ang_ac = path_ac_in(ll,3);
                    free_mission
                    break
                end
                c=c+1;
                cc=cc+1;
            end



        end
    end

    function selection_ac(varargin)
        tmp = main_ac.Matrix(13:14);
        act_pos=tmp;
        uav=0;
        map_update;
    end
```

```
function selection_uav1(varargin)
    button_state = get(varargin{1},'Value');
    if button_state == 1
        set(varargin{1},'BackgroundColor','y');
        uav_1.Children.FaceColor=[1 1 0];
        uav=uav+1;
    elseif button_state == 0
        set(varargin{1},'BackgroundColor','k');
        uav_1.Children.FaceColor=[0 1 0];
        uav=uav-1;
    end
    pan on
end

function selection_uav2(varargin)
   button_state = get(varargin{1},'Value');
    if button_state == 1
        set(varargin{1},'BackgroundColor','y');
        uav_2.Children.FaceColor=[1 1 0];
        uav=uav+2;
    elseif button_state == 0
        set(varargin{1},'BackgroundColor','k');
        uav_2.Children.FaceColor=[0 1 0];
        uav=uav-2;
    end
    pan on
 end

function re_size(varargin)
    m=0;
    mission
end

function zoom_in(varargin)
    focus=focus/2;
    map_update;
end

function zoom_out(varargin)
    focus=focus*2;
    map_update;
end

function up_shift(varargin)
   act_pos(2) = act_pos(2)+0.5*focus;
   shf=0;
   map_update;
end

function down_shift(varargin)
   act_pos(2)= act_pos(2)-0.5*focus;
   shf=0;
```

```
        map_update;
    end

    function right_shift(varargin)
        act_pos(1) = act_pos(1)+0.5*focus;
        shf=0;
        map_update;
    end

    function left_shift(varargin)
        act_pos(1)=act_pos(1)-0.5*focus;
        shf=0;
        map_update;
    end

    function pan_on(varargin)
        pan on
    end

    function pan_off(varargin)
        pan off
    end

    function extension1(varargin)
        set(fig,'position',[510 0 530 720])

        set(screen_h,'position',[60 60 435 435]);
        set(axes_1,'position',[15 620 100 100]);
        set(axes_2,'position',[115 620 100 100]);
        set(axes_3,'position',[215 620 100 100]);
        set(axes_4,'position',[315 620 100 100]);
        set(axes_5,'position',[415 620 100 100]);
        set(info11,'position',[15 600 100  20]);
        set(info12,'position',[15 580 100  20]);
        set(info13,'position',[15 560 100  20]);
        set(info14,'position',[15 540 100  20]);
        set(info15,'position',[15 520 100  20]);
        set(info21,'position',[115 600 100  20]);
        set(info22,'position',[115 580 100  20]);
        set(info23,'position',[115 560 100  20]);
        set(info24,'position',[115 540 100  20]);
        set(info25,'position',[115 520 100  20]);
        set(btn_zoom_in,'position',[465 465 30 30]);
        set(btn_zoom_out,'position',[465 60 30 30]);
        set(btn_extension1,'position',[480 10 40
40],'string','^','Callback',@shrink1);
        set(btn_extension2,'string','<','Callback',@extension2);
        set(btn_mission1,'position',[10 460 45 40]);
        set(btn_mission2,'position',[10 420 45 40]);
        set(btn_mission3,'position',[10 380 45 40]);
        set(btn_mission4,'position',[10 340 45 40]);
        set(btn_mission5,'position',[10 300 45 40]);
        set(btn_formation1,'position',[10 60 45 40]);
```

```
        set(btn_formation2,'position',[10 100 45 40]);
        set(btn_formation3,'position',[10 140 45 40]);
        set(btn_formation4,'position',[10 180 45 40]);
        set(btn_formation5,'position',[10 220 45 40]);
        set(btn_uav1,'position',[120 10 60 40]);
        set(btn_uav2,'position',[185 10 60 40]);
        set(btn_uav3,'position',[250 10 60 40]);
        set(btn_uav4,'position',[315 10 60 40]);
        set(btn_uav5,'position',[380 10 60 40]);

    end

    function shrink1(varargin)
        set(btn_extension1,'string','v','Callback',@extension1);
        set(fig,'position',[510 200 530 520]);

        set(screen_h,'position',[60 60 435 435]);
        set(axes_1,'position',[15 620 100 100]);
        set(axes_2,'position',[115 620 100 100]);
        set(axes_3,'position',[215 620 100 100]);
        set(axes_4,'position',[315 620 100 100]);
        set(axes_5,'position',[415 620 100 100]);
        set(info11,'position',[15 600 100  20]);
        set(info12,'position',[15 580 100  20]);
        set(info13,'position',[15 560 100  20]);
        set(info14,'position',[15 540 100  20]);
        set(info15,'position',[15 520 100  20]);
        set(info21,'position',[115 600 100  20]);
        set(info22,'position',[115 580 100  20]);
        set(info23,'position',[115 560 100  20]);
        set(info24,'position',[115 540 100  20]);
        set(info25,'position',[115 520 100  20]);
        set(btn_zoom_in,'position',[465 465 30 30]);
        set(btn_zoom_out,'position',[465 60 30 30]);
        set(btn_extension1,'position',[480 10 40
40],'string','v','Callback',@extension1);
        set(btn_extension2,'string','<','Callback',@extension2);
        set(btn_mission1,'position',[10 460 45 40]);
        set(btn_mission2,'position',[10 420 45 40]);
        set(btn_mission3,'position',[10 380 45 40]);
        set(btn_mission4,'position',[10 340 45 40]);
        set(btn_mission5,'position',[10 300 45 40]);
        set(btn_formation1,'position',[10 60 45 40]);
        set(btn_formation2,'position',[10 100 45 40]);
        set(btn_formation3,'position',[10 140 45 40]);
        set(btn_formation4,'position',[10 180 45 40]);
        set(btn_formation5,'position',[10 220 45 40]);
        set(btn_uav1,'position',[120 10 60 40]);
        set(btn_uav2,'position',[185 10 60 40]);
        set(btn_uav3,'position',[250 10 60 40]);
        set(btn_uav4,'position',[315 10 60 40]);
        set(btn_uav5,'position',[380 10 60 40]);
    end
```

```
    function extension2(varargin)
        set(fig,'position',[10 0 1030 720]);
        set(screen_h,'position',[60 60 750 640]);
        set(axes_1,'position',[830 620 100 100]);
        set(axes_2,'position',[830 470 100 100]);
        set(axes_3,'position',[830 320 100 100]);
        set(axes_4,'position',[830 170 100 100]);
        set(axes_5,'position',[830  20 100 100]);
        set(info11,'position',[930 700 100  20]);
        set(info12,'position',[930 680 100  20]);
        set(info13,'position',[930 660 100  20]);
        set(info14,'position',[930 640 100  20]);
        set(info15,'position',[930 620 100  20]);
        set(info21,'position',[930 550 100  20]);
        set(info22,'position',[930 530 100  20]);
        set(info23,'position',[930 510 100  20]);
        set(info24,'position',[930 490 100  20]);
        set(info25,'position',[930 470 100  20]);
        set(btn_zoom_in,'position',[780 670 30 30]);
        set(btn_zoom_out,'position',[780 60 30 30]);
        set(btn_extension1,'position',[770 10 40
40],'string','^','Callback',@shrink1);
        set(btn_extension2,'string','>','Callback',@shrink2);
        set(btn_mission1,'position',[10 660 45 40]);
        set(btn_mission2,'position',[10 600 45 40]);
        set(btn_mission3,'position',[10 540 45 40]);
        set(btn_mission4,'position',[10 480 45 40]);
        set(btn_mission5,'position',[10 420 45 40]);
        set(btn_formation1,'position',[10 60 45 40]);
        set(btn_formation2,'position',[10 120 45 40]);
        set(btn_formation3,'position',[10 180 45 40]);
        set(btn_formation4,'position',[10 240 45 40]);
        set(btn_formation5,'position',[10 300 45 40]);
        set(btn_uav1,'position',[235 10 60 40]);
        set(btn_uav2,'position',[320 10 60 40]);
        set(btn_uav3,'position',[405 10 60 40]);
        set(btn_uav4,'position',[490 10 60 40]);
        set(btn_uav5,'position',[575 10 60 40]);

    end

    function shrink2(varargin)
        set(fig,'position',[510 0 530 720])

        set(screen_h,'position',[60 60 435 435]);
        set(axes_1,'position',[15 620 100 100]);
        set(axes_2,'position',[115 620 100 100]);
        set(axes_3,'position',[215 620 100 100]);
        set(axes_4,'position',[315 620 100 100]);
        set(axes_5,'position',[415 620 100 100]);
        set(info11,'position',[15 600 100  20]);
        set(info12,'position',[15 580 100  20]);
```

```
        set(info13,'position',[15 560 100  20]);
        set(info14,'position',[15 540 100  20]);
        set(info15,'position',[15 520 100  20]);
        set(info21,'position',[115 600 100  20]);
        set(info22,'position',[115 580 100  20]);
        set(info23,'position',[115 560 100  20]);
        set(info24,'position',[115 540 100  20]);
        set(info25,'position',[115 520 100  20]);
        set(btn_zoom_in,'position',[465 465 30 30]);
        set(btn_zoom_out,'position',[465 60 30 30]);
        set(btn_extension1,'position',[480 10 40
40],'string','^','Callback',@shrink1);
        set(btn_extension2,'string','<','Callback',@extension2);
        set(btn_mission1,'position',[10 460 45 40]);
        set(btn_mission2,'position',[10 420 45 40]);
        set(btn_mission3,'position',[10 380 45 40]);
        set(btn_mission4,'position',[10 340 45 40]);
        set(btn_mission5,'position',[10 300 45 40]);
        set(btn_formation1,'position',[10 60 45 40]);
        set(btn_formation2,'position',[10 100 45 40]);
        set(btn_formation3,'position',[10 140 45 40]);
        set(btn_formation4,'position',[10 180 45 40]);
        set(btn_formation5,'position',[10 220 45 40]);
        set(btn_uav1,'position',[120 10 60 40]);
        set(btn_uav2,'position',[185 10 60 40]);
        set(btn_uav3,'position',[250 10 60 40]);
        set(btn_uav4,'position',[315 10 60 40]);
        set(btn_uav5,'position',[380 10 60 40]);
    end

end
```