SOFTWARE SUPPORT FOR GROUND CONTROL STATION FOR UNMANNED AERIAL VEHICLE

Mladen Jovanović∗
School of Electrical Engineering, Department for
Computer Science

Dušan Starčević
Faculty of Organizational Sciences, Laboratory for
Multimedia Communications

Zoran Jovanović
School of Electrical Engineering, Department for
Computer Science
University of Belgrade, Serbia

ABSTRACT
Uninhabited vehicles can be used in many applications and domains, particularly in environments that humans cannot enter (e.g. deep sea) or prefer not to enter (e.g. war zones). The promise of relatively low cost, highly reliable and effective assets that are not subject to the physical, psychological or training constraints of human pilots has led to much research effort across the world. Due to technological advances and increasing investment, interest in Unmanned Aerial Vehicles (UAVs) as a practical, deployable technological component in many civil applications is rapidly increasing and becoming a reality, as are their capabilities and availability. UAV platforms also offer a unique experimental environment for developing, integrating and experimenting with many other technologies such as automated planners, knowledge representation systems, chronicle recognition systems, etc. UAV performs various kinds of missions such as mobile tactical reconnaissance, surveillance, law enforcement, search and rescue, land management, environmental monitoring, disaster management. UAV is a complex and challenging system to develop. It operates autonomously in unknown and dynamically changing environment. This requires different types of subsystems to cooperate. In order to realize all functionalities of the UAV, the software part becomes very complex real-time system expected to execute real-time tasks concurrently. This paper describes proposed software architecture for visualizing and controlling UAV’s data acquisition operations. The rest of the paper is organized as follows: In Section 2 we give a brief survey of related developments. Section 3 outlines the engineering and system attributes of our particular UAV platform. Then, in Section 4 design goals and overall system architecture are discussed. Section 5 emphasizes software aspect of the architecture. Section 6 gives an overview of the developed system. In section 7, results of the experimental analysis of the system are described. Finally, the last section gives a brief discussion of the presented work and conclusions.

1 INTRODUCTION
UAVs are defined as “powered, aerial vehicles that do not carry a human operator, use aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or non-lethal payload” [1]. The Unmanned Aerial Vehicles are becoming more widely used in all types of applications. Aside from the military use of UAVs, from targeting and reconnaissance to actual weapons delivery [2], there are an increasing number of civilian uses for UAVs. These applications include search and rescue operations, land and environmental surveying, agricultural spraying and even a proposal to use high-flying UAV as stand-in for communications satellites. Realization of a typical UAV system requires complex hardware components and software components and highly time constrained coordination between these components. Especially, the software part consists of three grouped concurrent tasks (hard real-time tasks, soft real-time tasks and non-real-time tasks) and these are always run concurrently [3].

In this paper we present software architecture for visualizing and controlling UAV’s data acquisition operations. The rest of the paper is organized as follows: In Section 2 we give a brief survey of related developments. Section 3 outlines the engineering and system attributes of our particular UAV platform. Then, in Section 4 design goals and overall system architecture are discussed. Section 5 emphasizes software aspect of the architecture. Section 6 gives an overview of the developed system. In section 7, results of the experimental analysis of the system are described. Finally, the last section gives a brief discussion of the presented work and conclusions.

2 BACKGROUND
This section briefly references known usages of UAVs, as well as issues considering some existing software solutions for manning them.

2.1 Unmanned systems application
Uninhabited vehicles can be used in many applications and domains, particularly in environments that humans cannot
The promise of relatively low cost, highly reliable and effective assets that are not subject to the physical, psychological or training constraints of human pilots has led to much research effort across the world [4]. UAV platforms also offer a unique experimental environment for developing, integrating and experimenting with many Artificial Intelligence technologies such as automated planners, knowledge representation systems, chronic recognition systems, etc. [5]. The integration of ground wireless ad hoc networks and airborne UAVs is a promising solution for real-time data collection in wireless sensor networks. Such networks play a vital role in many mission critical tasks such as forest fire-fighting and battlefield support. The use of UAVs as mobile base stations is to close the gap between the sensor network and the command center. The UAVs maintain high bandwidth links with the command center, and communicate with the sensor nodes as it flies over these nodes to collect sensor data [6]. Henkel et al. [7] present a method for reliably collecting data events from sensors and forwarding the data via a mobile ad hoc network (MANET) to sensor monitoring stations located on an external network. At the core is a MANET concept that consists of ground and unmanned aircraft nodes. Unmanned aircraft enable a model whereby widely-spaced sensors are intermittently connected to the network and data is sent in stages as connections become available along each stage. Timely information about highway traffic conditions is very important, especially when traffic incidents or accidents occur. An aerial view is the best for traffic situations, particularly over highways. Unmanned aircraft equipped with video cameras and/or other sensors may be able to deliver the necessary information through video images with relatively low operational costs and risks to human life. Srinivasan et al. [8] developed a system for ATSS (Airborne Traffic Surveillance System) from the ground up, using unmanned aerial vehicles, digital video encoding, and transmission of data and multimedia video streams over microwave IP networks.

2.2 Software support

Wide range of research concerning UAVs is being conducted nowadays. Each of them encompasses specific areas of the UAVs applicability. Modeling is becoming increasingly important since models are used as “filters” or “placeholders” for not-yet-developed systems elements allowing system elements to come “on-line” when they are ready, rather than waiting for the entire system(s) maturation [9]. Theunissen et al. [9] introduced a concept to facilitate the integration of UAVs into controlled airspace using a datalink dialog with civil ATC (Air Traffic Control) system. To realize the desired integration, three development activities were performed: 1) the integration of the UAV simulation environment, an ATC simulator and an airspace command and control system into a common simulation environment; 2) the implementation of datalink interfaces to allow the exchange of the messages between the different simulators and 3) the development of the associated functions and user interfaces that will improve completion of the associated tasks. A wide range of simulators is provided by NASA Ames Simulation Laboratories including simulation with control of multiple UAVs [10]. In this context, The US Army conducted a manned-unmanned teaming (MUT) simulation in NASA Ames’ Vertical Motion Simulator (VMS) utilizing an attack helicopter flight model networked with a tactical Unpiloted Aerial Vehicle (UAV) ground control station (GCS). Aircraft control relationships and display concepts for multiple UAVs controlled from a single manned aircraft were investigated. CAE has developed the synthetic environment as a practical representation of the real world for the UAV simulation [11]. A ground control station is being used to operate a simulated UAV in a synthetic environment. The synthetic environment denotes practical representation of the real world [11]. In this instance, it comprises the UAV air vehicle simulation, payload simulation ,and comprehensive tactical environment. The Georgia Institute of Technology (GIT), designed and developed an UAV multi-sensor system that provides terrain navigation information [12]. Rasmussen et al. [13] developed MultiUAV, a simulation that is capable of simulating multiple unmanned aircraft vehicles which cooperate to accomplish a predefined mission. The simulation was constructed using the Mathwork’s Simulink simulation software. Construction of the simulation satisfied the need for a simulation environment that researchers can use to implement and analyze cooperative control algorithms. The simulation is implemented in a hierarchical manner with inter-vehicle communication explicitly modeled. During construction of MultiUAV, issues concerning memory usage and functional encapsulation were addressed. MultiUAV includes plotting tools and links to an external program for post-simulation analysis. Each of the vehicle simulations include six-degree-of-freedom dynamics and embedded flight software. The embedded flight software consists of a collection of managers (agents) that control situational awareness and responses of the vehicles. Managers included in the simulation are: Tactical Maneuvering, Sensor, Target, Cooperation, Route and Weapons. Hong et al. [14] proposed a hierarchical architecture for Unmanned Autonomous Helicopter System that guarantees the real-time performance of hard real-time tasks and the re-configurability of soft real-time or non-real-time tasks under the RT-Linux. This software architecture has four layers: hardware, execution, service agent and remote user interface layer according to the reactivity level for external events. In addition, the layered separation of concurrent tasks makes different kinds of mission reconfiguration possible in the system. An Unmanned autonomous helicopter system was implemented using Kyosho RC Helicopter to test and evaluate the performance of the proposed systems.

3 UAV PLATFORM

Before discussing the issues considering design of software platform, it is worthwhile to briefly discuss the engineering and system attributes of our particular UAV platform. Flying platform is proposed as an experimental lightweight UAV, purpose-built for small-scale reconnaissance and surveillance missions. The platform has a wing span of approximately 2.4 meters, while fuselage length amounts approximately 2.2 meters. It can be rapidly disassembled and reassembled, launched by catapult system (ie. thrown in the air for take-off), and easily transportable in a back-pack. Aircraft weighs 10 kg, while it can carry payload up to 8 kg in weight. The propulsion system consists of a 12V/5Ah accumulator battery and a small two-cycle 50cc gas engine driving a fullyfolding
propeller. Flight endurance is up to 240 minutes, depending on payload amount. The aircraft can fly with maximum air speed of 130 kph, while loiter speed amounts from 90 kph to 115 kph. It can operate in a vertical flight range from 100m to 2500m and within a 20-kilometres horizontal radius. Recovery (landing) is by a parachute that deploys on command from a cavity above the fuselage centre-section. Payload vary from a video camera; radio telecommand receiver link with system for manning the aircraft which includes servos for controlling the flying surfaces; separated video link and telemetry flight data link for transmission of video streams and flight data, respectively. The aircraft includes on-board GPS for basic navigation; an airspeed indicator and barometric altimeter; fuel level measurer, engine temperature measurer and accumulator voltage measurer.

The ground part of the UAV platform includes radio telecommand system for sending control data to flying platform; separated video link and telemetry flight data link for reception of video streams and flight data, respectively; computing system for processing, managing and storage of received flight data; video receiver for receiving and decoding incoming video streams.

Applications of the aircraft can be diverse: mobile tactical reconnaissance, surveillance, intelligence gathering, law enforcement (such as policing and security operations), search and rescue, land management, environmental monitoring, disaster management, and stock and station control. The relatively small size makes it cost effective, energy efficient, easily transportable, flexible and scalable.

4 UAV SYSTEM ARCHITECTURE

The overall architecture of the UAV system is shown in Figure 1.

During autonomous flight, a ground station operator is responsible for

- A primary mission task, in which the operator tracks the UAV waypoints and reports on targets at known coordinates.
- A secondary mission task, in which the operator navigates the aircraft through the environment.
- A system monitoring task, in which the operator monitors aircraft instruments and detects possible system failures.

In addition, operator monitors the video link and records and identifies ground targets at each waypoint. The GCS operator can also change the waypoints that the UAV is being directed to visit, and can change the control parameters.

5 SOFTWARE ASPECT OF THE ARCHITECTURE

The overview of the proposed software architecture is shown in Figure 2.

![Figure 2. Overview of the proposed software architecture.](image)

Environment state data and flight platform data are produced as messages complying with the specific formats and are processed by corresponding modules to make them suitable for visual presentation. Terrain and navigational modules visualize terrain environment and waypoints tracking. User also performs monitoring task and identifies failure when it occurs. This is achieved with aircraft instrument presentation module. Through the network interface system receives data from the aircraft. These data includes environment state and flight platform data, as well as real-time video stream incoming from the aircraft. Planning and plan execution tasks base its decisions on information derived from the camera vision subsystem. In addition, terrain view visualizes aircraft’s environment generated from digital terrain model as well as vehicle’s exact position and orientation.

Figure 3 gives more detailed view of software architecture in terms of software modules and their layered organization. In order to get more flexibility and portability from our solution, it is divided into three layers: Application layer – consists of software modules which make the design of our system; Middleware – comprised of corresponding implementation platform libraries; System layer – includes system level software and libraries.
Figure 4 presents a Unified Modelling Language (UML) deployment diagram which describes possible physical deployment of developed software modules. Proposed solution can be executed in a local environment and in network environment as well. Part of the communication infrastructure for receiving data from flying platform includes video receiver which converts wireless analog video signal into PAL/NTSC format; video encoder which digitally encodes the video and flight data receiver. Central computer is connected with these devices and contains corresponding software modules (Datasource, Instruments, Map, Camera, 3DNavigation). Produced messages and video streams can be further passed to interested clients through local computer network.

6 DESIGN DETAILS

The user first performs mission selection operation by specifying waypoints to track and targets of interest to zoom and inspect when found. After that he performs the take-off operation. During the flight, the user performs the monitoring task which includes information about vehicle’s state and status of the execution of the primary flight task, i.e., the mission completion degree. We are also relying on use of design patterns [15] to increase the efficiency and reusability of our design decisions. Design patterns have already found application in design of reusable avionics software [16]. Real Time Specification for Java (RTSJ) [17] was designed with the ability to combine plain Java components with the real-time ones in a type-safe and higher performance manner. The Ovm virtual machine [18] presents an open source implementation of the RTSJ virtual machine. It implements some core VM features such as thread scheduling and memory management and is used as a core execution environment in the developed system. Following subsections give a closer look at some important aspects of software design.

6.1 Aircraft Instruments

UAV operators can incur a high workload due to the fact that today’s modern aircrafts produce vast amount of data which has to be presented in real-time. In situations where operator must react in a limited period of time and avoid hazardous situations, it is very important to present flight data in a form that can be easily interpreted and processed having in mind throughput of human sensory and perceptual apparatus.

Requirements for aircraft instruments’ design are as follows:

- Cognitive goal. In order to decrease cognitive fatigue, controls should operate in a way that represents an operator’s intuitive understanding.
- Response goal. This concerns minimizing UAV response time and is achieved by underlying implementation technology.

In order to streamline and optimize operator-vehicle interface we have decided to classify instrument types and to model the structure of an instrument as is shown in Figure 5.

AbstractInstrument represents an abstraction of common properties for all types of instruments. VisualInstrument and ComplexInstrument are derived from base class.
VisualInstrument is further derived according to specific kinds of instruments which can be found in aircraft cockpits. Introduction of ComplexInstrument type enables modeling of composite instrument containing other instrument types. This kind of realization of instruments hierarchy presents an example of Composite design pattern [15]. Presented approach to modeling aircraft instruments enables construction of instrument tables of random complexity and layout of instruments. The process of creation of instruments in runtime (Fig. 6) is realized using Factory Method design pattern [15].

Figure 6. Simplified UML illustration of structure for creating instruments.

An example of concrete instrument table implemented from underlying model is shown in Figure 7. Presented view operates in a way that represents an operator’s intuitive understanding. Controls that have different functions are distinguishable from one another in order to clearly assess flight status data. Instruments and controls with related functions are grouped together in a logical arrangement which helps reduce instrument scan time and lowers operator’s workload. An approach to design of user interfaces is presented in [19].

Figure 7. Concrete realization of an instrument table.

6.2 Mission Navigation

UAV missions are monitored and controlled by two-dimensional navigational view and three-dimensional terrain view. Figure 8 describes a simplified UML description of module for controlling and navigating UAV missions. Module’s main frame (MapMainFrame) includes instances of three views: route navigation view (MapVerticalViewPanel), terrain profile view (MapProfilePanel) and mission data control view (MapDataPanel). Map connection thread (MapConnectionThread) is responsible for communicating with Datasource module and keeping the corresponding views up to date. Since planning of missions is also allowed, communication between views must be established. In our case, it is realized using Mediator design pattern [15]. Process of rendering of route navigation view and terrain profile view is realized relying on Observer design pattern [15]. Classes and interfaces which realize rendering of corresponding views are shown in Figure 9.

Figure 8. Simplified UML description of UAV mission navigation.

Figure 9. The structure of mission navigation rendering mechanism.
Figure 10 shows an example of mission control navigation display realized according to previously described models. Navigational view comprises terrain profile view (bottom), route navigation view (top) and data control view (left). Each mission consists of number of routes containing waypoints that need to be visited and reported. Operator manually enters waypoints positions. Based on their positions planned routes are calculated. During the flight aircraft can deviate from planned route and operator is allowed to correct aircraft’s position and orientation through manual control console.

Figure 10. Mission control navigational display.

Terrain description is built from elevation data on a regularly spaced grid. These data are tiled with triangle strip arrays. Figure 11 gives an example of rendered terrain environment. Java3D [20] presents an open source graphics library with the built-in mechanism for rendering this triangle mesh more effective. Each elevation point is depicted with nine parameters describing color, normal and coordinates. Parameters are contained by reference and shared between user and core graphics routines which allow more efficient memory usage. Rendering level of details (LOD) is implemented allowing operator to choose the precision of the rendered three-dimensional terrain model.

Figure 11. Generated terrain model.

6.3 Network Interface

One of the fundamental capabilities of the proposed architecture is to receive messages provided by the UAV’s sensors as well as real-time video stream from the camera mounted on the aircraft. Figure 12. gives an overview of the flight data message formats.

Figure 12. Flight data messages formats.

Network communication software is encapsulated in Datasource module and has two parts. One part is a set of Java-based network daemon threads that is able to read messages incoming from flight data receiver and distribute them to interested modules which can be deployed in a local or remote environment. Specialized Java classes are designed to be called by an application that maintains the state of the environment and the vehicle. This description is updated as messages from the aircraft are being received. Communication between Datasource and other modules is realized using Remote Proxy design pattern [15] as is shown in Figure 13. ServerConnectionThread maintains collection of ClientConnectionThread instances which present local representative of the remote client. In this instance, the remote client is Instruments module, that is InstrumentsConnectionThread which keeps instrument table up to date. Similar realization has already been introduced by Sharp [16] on the example of Boeing avionics software.

Figure 13. Structure of the communication between Datasource and other modules.

Figure 14. Dialogs of Datasource module.
Second part represents subsystem for processing video stream incoming from the vehicle. On the side of the UAV the onboard computer system is integrated with camera gimbal mounted beneath the aircraft fuselage. To support UAV operations, a video camera on the UAV produces a video stream that must be displayed with minimal real-time delay on display console. There are several steps to this process:

- Video feed from off-board source (UAV).
- Sending video to hosts on aircraft's network.
- Users’ hosts receive video and display it.
- Users analyze received data and send commands to UAV to control it.

Our prototype supports the first three of these steps. The fourth step, in which operator interacts with the UAV in real time, is manifested in a QoS requirement: at least on certain display consoles, images must appear a very short time after the camera records them; we cannot buffer or interrupt the stream in order to accommodate variable latency and periods of network congestion.

6.4 Camera Vision

JMF is an API defined jointly by Sun Microsystems, Inc. and IBM Corporation. The main aim of the product is the provision of real-time control of multimedia data (video and audio) by means of streaming techniques. The API can be used in applets and in stand-alone applications developed in Java. Apart from providing facilities for data processing (codecs, multiplexers and demultiplexers, renderers, ...), JMF allows transmission and reception of multimedia data using RTP in unicast and multicast modes, and the utilization of several capture devices like webcams. There are different versions of the API, the most recent is v2.1.1, although new improvements are being included [21]. Figure 15 presents an example of rendered video stream.

![Figure 15. Screenshot of the camera view](image)

7 EXPERIMENTAL EVALUATION

In this section we explain details and results of the experimental analysis performed upon developed prototype of software solution. In this context, we have been examining three critical aspects of execution of developed software system relevant for UAV successful task completion. These three include communication with the device for reception of flight data from flying platform, distribution and rendering of incoming video streams, and navigation of the vehicle through simulated three-dimensional terrain environment. In the rest of the section previously mentioned aspects are described respectively.
Module was tested using codification standards based on opposite tenets in regard to required bandwidth and processing and memory resources. Given results show that bit rate values are within the range defined by these standard. Variations of average bit rate values were small implicating that signal was stable.

With regard to navigation in generated three-dimensional terrain environment, we have been examining graphical rendering device’s frame rate dependency on size of the digital terrain model (DEM) division. Graphical terrain description is built from elevation data on a regularly spaced grid. This approach has already been introduced by McCarty et al. [22]. Digital terrain model is divided into grid of adjacent regions (Figure 18.).

DEM used in the experiment covers an area of 110591 meters in geographic length and 80174 meters in geographic width which is represented by 1201x1201 matrix. Thus, DEM includes 1442401 heights, while cell size amounts 92 x 67 meters. Experiment was performed on Intel Celeron 2.00GHz, 1.5 Gb RAM, GPU Intel 82945G 128 MB platform. Results are presented in tabular form (Table 1) and in the form of diagram (Figure 19).

Table 1. Frame rate dependency on terrain division size.

<table>
<thead>
<tr>
<th>Number of regions per degree of geographical coordinates</th>
<th>Region’s dimensions (km)</th>
<th>Number of frames rendered per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>40,087 x 55,995</td>
<td>15,28</td>
</tr>
<tr>
<td>3</td>
<td>26,724 x 36,863</td>
<td>21,67</td>
</tr>
<tr>
<td>4</td>
<td>20,043 x 27,847</td>
<td>33,77</td>
</tr>
<tr>
<td>5</td>
<td>16,034 x 22,118</td>
<td>44,70</td>
</tr>
<tr>
<td>6</td>
<td>13,362 x 18,431</td>
<td>53,74</td>
</tr>
<tr>
<td>7</td>
<td>11,453 x 15,798</td>
<td>76,09</td>
</tr>
<tr>
<td>8</td>
<td>10,021 x 13,823</td>
<td>79,22</td>
</tr>
<tr>
<td>9</td>
<td>8,908 x 12,287</td>
<td>98,93</td>
</tr>
<tr>
<td>10</td>
<td>8,017 x 11,06</td>
<td>102,47</td>
</tr>
<tr>
<td>11</td>
<td>7,288 x 10,053</td>
<td>131,45</td>
</tr>
<tr>
<td>12</td>
<td>6,681 x 9,215</td>
<td>133,83</td>
</tr>
<tr>
<td>13</td>
<td>6,167 x 8,507</td>
<td>124,79</td>
</tr>
<tr>
<td>14</td>
<td>5,726 x 7,899</td>
<td>133,05</td>
</tr>
<tr>
<td>15</td>
<td>5,344 x 7,372</td>
<td>136,50</td>
</tr>
<tr>
<td>16</td>
<td>5,01 x 6,91</td>
<td>139,40</td>
</tr>
<tr>
<td>17</td>
<td>4,716 x 6,505</td>
<td>146,87</td>
</tr>
<tr>
<td>18</td>
<td>4,454 x 6,143</td>
<td>147,27</td>
</tr>
<tr>
<td>19</td>
<td>4,219 x 5,820</td>
<td>142,17</td>
</tr>
<tr>
<td>20</td>
<td>4,008 x 5,529</td>
<td>147,68</td>
</tr>
</tbody>
</table>
Results indicated approximately linear frame rate increase according with increase of terrain division size. This is expected since regions are rendered independently with resolution that depends on viewer’s current position and orientation. Increase in number of segments allows LOD optimization mechanism to render higher number of segments with lower resolutions which further affects rendering performances. However, increase in number of regions that terrain is consisted of affects rendering quality since visible seams are appearing along the joints of adjacent regions. From the start of execution with an empty cache to the first rendered frame takes approximately 10 seconds. Using cached data cuts this time in half. Performances of the developed module are comparable to some similar existing solutions [23]. With respect to trade-off between rendering performance and quality, module achieves the most optimal results in extent from 5 to 10 regions per degree of geographic coordinates.

8 Conclusions and Future Work

In this paper, we have proposed software architecture for GCS for UAV. The operator observes the system through four separate views. First presents an instrument table for monitoring basic flight data and state of the vehicle. Second displays terrain map together with terrain profile and serves for mission tracking. Third view presents three-dimensional representation of the environment and visualizes vehicle’s exact position and orientation. Forth view renders real-time video stream from camera mounted on the aircraft. Based on operators’ abilities and preferences developed views can be deployed on a single or multiple displays. Work presented in this paper extends our previous efforts on designing simulation environment for UAV [24]. Based on the performance tests and results of the performed experiments within the developed simulator we have decided to conclude our work by building system for manning the real aircraft. With the proposed solution we can apply this to the actual unmanned aircraft and can confirm its exact operation. Also, the proposed solution can be applied to various unmanned control systems. Our future work includes adaptation of the system to mobile platforms, as well as integration with swarming intelligence technologies.

ACKNOWLEDGMENTS

The authors wish to express gratitude to Dragan Obredović from IMTEL Communications, Belgrade and Željko Obrenović from Technical University Eindhoven for their feedback and advice.

9 REFERENCES


