Czech Technical University in Prague
Faculty of Electrical Engineering

Doctoral Thesis

August, 2018

Martin Selecký

Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Computer Science

Mixed-Reality and High-Fidelity Simulation in Autonomous Unmanned Aerial Systems

Doctoral thesis

Martin Selecký

Prague, August 2018

Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Artificial Intelligence and Biocybernetics

Supervisor: Doc. Ing. Jan Faigl, Ph.D.
Supervisor-Specialist: Ing. Milan Rollo, Ph.D.

Acknowledgments

I would like to express my special thanks to my supervisor Jan Faigl for motivating and encouraging me to publishing papers and for his advice and guidance in my latest research and during the writing of this thesis. My special thanks also go to my supervisor specialist, Milan Rollo who introduced me to the area of unmanned aircraft systems, provided me with access to all the experimental hardware and helped me at many occasions when I was facing research and development-related or any other problems. I would also like to thank prof. Michal Pěchouček for his guidance and study-related advice and to my colleagues at AgentFly Technologies and Artificial Intelligence Center for their help during my research and their cooperation during all the flight experiments. Finally, I would like to thank my family, friends and my girlfriend Anička for their endless toleration and support during my research and writing.

The research activity published in this thesis was supported by the Czech Ministry of Defence grant OVCVUT2010001, the U.S. Army grant no. W911NF-11-1-0252, the AFRL contract FA8075-12-D-0001, Czech Technical University grant no. SGS14/143/OHK3/2T/13, by the Czech Science Foundation (GAČR) under research project No. 16-24206S, Ministry of Agriculture of the Czech Republic under contract No. QJ1520187, and by the Technology Agency of the Czech Republic under project No. TA01030847.

Abstract

This thesis addresses the problem of safe and efficient development of multi-robot systems, in particular, systems of multiple Unmanned Aerial Vehicles (UAVs). Since these Unmanned Aircraft Systems (UASs) operate in the real world, it is necessary for them to be robust with respect to the aspects of the real world, such as unreliable and delayed communication, noisy sensors, limited computational power of on-board computers, limited flight time, effects of the environment, and many others. These aspects are often connected and addressing them separately may lead to only partial solutions and unexpected problems during the development.

Recently, these systems have been developed using either classical simulations that employed computational models for each part of the system, hardware-in-the-loop (HIL) simulations, where only some parts of the aircraft were modeled, or using fully hardware deployed testbeds. The new emerging concept of Mixed-Reality (MR) simulations assumes simultaneous co-existence of entities on different levels of virtualization. This way, the MR simulations provide the fidelity of fully hardware deployed testbeds while keeping the risks and costs of the tests low and scalability of the simulation high. The realization of the MR simulations has specifics, that have not been described or addressed yet.

In this thesis, the issues and current approaches to the development of multi-UAV systems are presented. These approaches are extended by proposing a Mixed Reality simulation Architecture for development of Multi-Robot systems (MiRAMuR) together with incremental development method that allows efficient development of such systems. This is supported by the risk and cost analysis of the proposed methodology. This architecture is implemented within a framework for Command and Control (C2) of multi-UAV teams. In such teams, the communication and interaction with the environment is crucial and so these aspects of the MR simulations are addressed as well. When facing the interaction with real environment, wind effects showed to significantly affect the MR simulation results. Therefore, the thesis addresses the trajectory planning that produces flight plans executable by real UAVs in wind. Moreover, the problem of refinement of the wind model by persistent measurement is addressed that can further improve the fidelity of the MR simulation.

The proposed methodology has been validated in development of three various aviation systems which show the versatility of the methodology for UASs research and development. In particular, these are (i) a system for command and control of a heterogeneous team of autonomous unmanned aircraft; (ii) a system for verification of collision avoidance methods among multiple fixed wing UAVs and for training the system operators; and finally, (iii) a system for increased flight safety in a domain of light-sport aircraft employing optionally piloted aircraft.

Anotace

Tato práce se zabývá problematikou bezpečného a efektivního vývoje multi-robotických systémů, konkrétně systémů sestávajících z více bezpilotních strojů (UAV). Pro správnou funkci takovýchto bezpilotních systémů (UAS) v reálném prostředí je třeba, aby byly robustní vzhledem k vlivům reálného světa jako je nespolehlivá komunikace, nízké přenosové rychlosti, interference radiového signálu, senzorický šum, omezená výpočetní kapacita palubních počítačů, omezená doba letu, vlivy externího prostředí na let a podobně. Tyto jevy jsou často vzájemně provázány a jsou-li řešeny odděleně, může to vést k neočekávaným problémům a pouze částečným řešením.

V současné době se UAS vyvíjejí bud' (i) za pomoci klasických simulací využívajících výpočetní modely pro každou část systému; (ii) s využitím hardware-in-the-loop (HIL) simulací, kde jsou modelovány pouze některé komponenty UAV a zbylé jsou reprezentovány skutečnými hardwarovými prvky; nebo (iii) se používají hardwarové testovací systémy plně osazené všemi potřebnými senzory a zařízeními. Nově se za účelem vývoje UAS začíná využívat koncept Mixed-Reality (MR) simulací, který předpokládá současnou koexistenci virtuálních i hardwarových entit. MR simulace tak poskytují simulační věrnost srovnatelnou s hardwarovými testovacími systémy a zároveň zachovávají nízkou míru rizika a ceny experimentů spolu s vyšší škálovatelností výsledného systému. Samotná realizace MR simulace má však specifika, která zatím nebyla nikde řešena ani publikována.

V této práci jsou diskutovány problémy vývoje multi-UAV systémů spolu s existujícími přístupy k vývoji těchto systémů. Tyto přístupy jsou pak rozšířeny o námi navrženou architekturu pro vývoj multi-robotických systémů pomocí MR simulací (MiRAMuR) spojenou s metodou inkrementálního vývoje umožňující efektivní návrh a implementaci těchto systémů. Tato architektura je implementována v rámci systému pro řízení autonomních multi-UAV týmů, kde je kritické zabezpečit komunikaci mezi jednotlivými entitami a interakci s okolním prostředím. Proto se práce zabývá těmito aspekty MR simulací podrobněji.

Jelikož má okolní prostředí, konkrétně vítr, značný vliv na průběh simulace, je řešen i problém plánování trajektorií ve větru. Přesněji jde o hledání časově optimální trajektorie, kterou je možné proletět bezpilotním letounem s omezeným náklonem i při působení větru. Za účelem zvýšení věrnosti MR simulace je dále řešeno také zpřesňování modelu větru pomocí persistentního sběru dat z prostředí s překážkami.

Metodologie popsaná v této práci byla použita a validována v rámci tří různých leteckých systémů. Takto prezentované široké možnosti využití demonstrují univerzálnost našeho přístupu pro výzkum a vývoj v oblasti UAS. Konkrétně je prezentován (i) systém pro řízení heterogenního týmu autonomních UAV; (ii) systém pro verifikaci metod pro řešení kolizních situací mezi více UAV a pro školení UAV operátorů; a (iii) systém pro zvýšení bezpečnosti letu pro volitelně pilotovaná ultra-lehká letadla.

Contents

Abbreviations

BVLOS	Beyond Visual Line of Sight
CA	Collision Avoidance
C2	Command & Control
CFD	Computational Fluid Dynamics
CIP	Critical infrastructure protection
FDM	Flight Dynamics Model
GCS	Ground Control Station
GNSS	Global Navigation Satellite System
HART	Human, agent, and robot teams
HIL	Hardware-in-the-Loop
HMI	Human machine interface
HWE	Physical/Hardware Entity
IMU	Intertial Measurement Unit
ISTAR	Intelligence, Surveillance, Target Acquisition and Reconnaissance
MiRAMuR	Mixed Reality simulation Architecture for development of Multi-Robot systems
MR	Mixed-Reality
NFZ	No-Flight Zone
PCAS	Portable Collision Avoidance System
ROP	Repeated Orienteering Problem
ROS	Robot Operating System
RPAS	Remotely Piloted Aircraft System
SIL	Software-in-the-Loop
SWE	Virtual/Software Entity
TCAS	Traffic Collision Avoidance System
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take-Off and Landing

Chapter 1

Introduction

1.1 Unmanned Aerial Systems

The utilization of *Unmanned Aerial Vehicles* (UAVs) has been for years primarily in a domain of military applications. Their massive deployment began in the 1990s and their usage is steadily increasing. Their first missions were for execution of reconnaissance and monitoring flights, and later they were equipped with weapon systems. These were large and complex airplanes, remotely operated via satellite lines. Over the time, the UAVs were getting smaller and started to be utilized by wider range of military units—from aircraft with wingspan of units of meters operated *Beyond Visual Line of Sight* (BVLOS) of the pilot, to small mobile devices with the ability to be carried in backpacks that could be used for small-scale explorations. All the vehicles used in this sector shared a focus primarily not on their price or operating costs, but rather on their robustness and ability to be deployed in hostile conditions.

Since the turn of the millennium, an upward trend in the use of UAVs in the civilian sector can be observed. Due to the advances in miniaturization of electronic devices that have taken place in recent years, it is possible to construct small UAVs with sufficient power to perform tasks that have only been addressed by piloted aircraft in the past. Primarily, this was a progress in the development of embedded computers with high computing power, miniaturization of sensors, improved range and throughput of communication modems, and increased battery capacity. All this enabled operation of control algorithms directly on board UAVs and online transfer of data from aircraft to *Ground Control Stations* (GCSs). Progress in these areas also led to a reduction in the prices of key components, leading to lower costs of these systems compared to the military applications.

The operation of UAVs brings many advantages over piloted aircraft, the most notable being their lower acquisition cost and operating costs. Another advantage is the ability of UAVs to be deployed in areas where piloted aircraft cannot be used, for example indoor areas of buildings, areas with increased levels of radiation, or with chemical or temperature dangers. Usually, the aircraft, or a set of multiple aircraft, together with their corresponding GCS, communication

equipment, sensory payload, and other associated components are collectively labeled by the term *Unmanned Aerial/Aircraft System* (UAS). These systems are used for a number of applications such as in geodesy and cartography (where resources are used for mapping, terrain modeling, mass volumes calculation, soil erosion mapping, etc.), for aerial photogrammetry (e.g., for real estate agencies, insurance companies, tourism), film industry, inspection and monitoring of infrastructures (e.g., monitoring of photovoltaic power plants, high voltage power lines, bridge constructions), agriculture and forestry, and a number of other areas.

For some types of tasks, it may be convenient to use more aircraft at the same time to increase the speed of the mission execution or the overall team performance in area coverage tasks. These multi-UAV teams—either homogeneous, consisting of vehicles with the same capabilities, or heterogeneous, where individual aircraft have a different set of capabilities at their disposal—present a significant increment to the cognitive load of their pilots and operators. To decrease this cognitive load and to allow one operator to control more UAVs at the same time, the automation of the individual UAVs has to be improved. In result, the focus must shift from the development of a solitary intelligent aircraft to teams of cooperating intelligent UAVs in a complex unmanned aircraft systems. Sheridan and Verplanck [173] defined categorization of levels of automation as shown in Table 1.1. According to this categorization, the current automation level of the UASs is the lowest. A process that would allow development of complex multi-UAV systems with gradual transition of the UAS automation to the higher levels is required. Incremental development of complex multi-UAV systems, where teams of cooperating intelligent UAVs coordinate their actions to autonomously accomplish missions assigned by the system operator is the main motivation and subject of this thesis.

Table 1.1: Levels of automation as defined by Sheridan and Verplanck [173].

Automation Level	**Automation Description**
1	The computer offers no assistance: human must take all decision and actions.
2	The computer offers a complete set of decision/action alternatives, or
3	narrows the selection down to a few, or
4	suggests one alternative, and
5	executes that suggestion if the human approves, or
6	allows the human a restricted time to veto before automatic execution, or
7	executes automatically, then necessarily informs humans, and
8	informs the human only if asked, or
9	informs the human only if it, the computer, decides to.
10	The computer decides everything, acts autonomously, ignoring the human.

1.2 Development of Complex Multi-UAV Systems

Development and validation of distributed algorithms for control of heterogeneous robot teams are complicated tasks since there is a lot of complex issues that need to be taken into account [87]. Some of them, like complex interactions among the team members, can be modeled with the help of multi-agent simulations and software simulations [63], [101]. However, since the goal of the development process is to deploy the algorithms in the real-world, software agent simulations by themselves are not appropriate in these cases. Effects of the real-world phenomena like those of the weather, communication issues, sensor and actuator errors or limited computational resources are difficult to model on high-fidelity levels. The whole system needs to be verified using real hardware assets, which can be very costly if the verification is done at every stage of the system development and integration. Moreover, these real-world issues are often connected; so, dealing with only some of them in dedicated simulations may not produce correct results.

Recently, experiments have been conducted both in a virtual world and in the real-world and each of them has its limits and justifications [83]. Simulations are much easier to be set up and repeated, they can be easily used to model the basic functionality of the developed system and allow quick observation of results of interactions among entities and the environment. Simulations can be used to study effects of various environmental phenomena by modifying only specific environmental conditions and fixing others thus removing possible sources of noise. Experiments in virtual environments can significantly save costs in cases of malfunctions and accidents. This applies even more in the case of unmanned aerial vehicles, where the consequences of accidents are much more severe than in other types of robots. However, even with increasing computational resources, it is challenging for software simulations to incorporate all sources of inputs from the real-world for realistic modeling of the behavior of the developed system. That is why the real-world experiments should be obligatory step for obtaining realistic results in later stages of development of a robotic system and for validating the robotic software's robustness before it is deployed in real missions.

Mixed-reality (MR) [122], [40] simulations gain benefits of both these worlds. They present a world where both virtual and real objects and entities can co-exist and interact in real-time. This allows the system developers to get more insight into the behavior of the entities (e.g., by visualization of their inner states) and to perform much cheaper and safer experiments with part of the system being real and the other part virtualized [41], [83]. MR simulations also relieve simulators from recreating complete worlds since simulation occurs in a partially real world where certain phenomena, such as noise and complex physics, do not have to be modeled [87], [169]. In contrast to *hardware-in-the-loop* (HIL) simulations [48], [149], that have been used for a long time for efficient system development, the concept of MR simulations allows for easily executable co-existence and interaction of purely virtualized and fully hardware deployed entities.

This thesis elaborates on the concept of MR simulations and presents a methodology of

using MR simulations for incremental development of complex multi-robot systems that is, as is further shown, cost and risk efficient.

1.3 Motivation

The main motivation of this thesis is a lack of a methodology that would allow gradual increasing of the automation of multi-UAV systems. When studying the problem deeper, the following observations were made.

- **There was no system for command and control of team of autonomous UAVs with onboard intelligence**, at least to the best of our knowledge. Although there are various kinds of GCS software solutions used for control of UAVs with different autopilots, there is, to our knowledge, none that would be able to control a team of autonomous UAVs on a high-level, by an assignment of tasks to the team. Development of such a system was the primary motivation for the research behind this thesis and it led to an observation of the following findings.

- **There are only a few works dealing with a development methodology for complex multi-UAV systems**. Most of the works that describe the UAS development from the simulation to the real-world deployment deal with a development of single UAV systems or with systems of swarming entities without taking the intelligence of individual entities into account.

- **The concept of MR simulations has not yet been presented in the context of implementation and deployment**. Even though there are works that address development of robotic systems using MR simulations, to the best of our knowledge, there is none that would present implementation specifics of MR simulation concept, or show its benefits and challenges based on real experiments.

- **There is no work dealing with a comprehensive set of issues that are characteristic for the real-world applications and their effect on the development of complex robotic system**. There are works dealing with isolated problems of a real-world deployment of robotic applications—for example, unreliable communication, imprecise localization, energy consumption modeling in robotic systems, effects of wind on the flight, etc. However, they do not study the effects of such issues on the behavior of the multi-robotic system as a whole (e.g., how does the localization inaccuracy, or unreliable communication influence the behavior of the system). When the exact effect of these real-world deployment issues on a system under development is not known, the development process becomes harder since it may be difficult to discover the reasons of an incorrect system behavior. MR simulations can help in this case since the developers can add the

real-world feeds one at a time by incrementally lowering the virtualization level, as will be described later in this thesis.

1.4 Preview of the Thesis Contributions

Particular contributions of this thesis, with already published papers if applicable, are as follows.

- Concept of MR simulations extended to the domain of autonomous UAVs
  - MR simulation Architecture for development of Multi-Robot systems (MiRAMuR) was proposed [169].
  - Process of incremental development of multi-UAV systems was analyzed from risk and cost perspectives, Chapter 4 and [170].
- Architecture validation by implementation within a framework for UAS oriented research and development
  - Framework for Command and Control (C2) of a heterogeneous team of autonomous UAVs implementing the MiRAMuR concepts was proposed and implemented, Chapter 5 [166], [165].
- High-fidelity communication modeling
  - Communication subsystem for MR simulations was proposed and verified, Chapter 6, [171].
- Environment model in MR simulations
  - Method of active building of MR environment was proposed, Chapter 7.
  - Problem of refinement of the wind map by persistent measurement was formulated and solved as Repeated Orienteering Problem (ROP), Chapter 9.
- Utilization of the environment model for high-fidelity entity modeling
  - Method for time-optimal trajectory planning in horizontal wind was proposed, Chapter 8, [172].
- Validation in case studies
  - Proposed methodology was validated in various research and development projects, Chapter 10, [164], [167], [169], [170].

1.5 Thesis Overview

This thesis consists of four main parts. The first part introduces the problem addressed by the thesis—after this introductory chapter, the problems frequently encountered when moving from software simulations to full hardware deployment of algorithms in the physical world are presented in Chapter 2 together with the MR concept and its utilization in multi-UAV system development. The overview of problems and system development approaches leads to the specification of the thesis goals presented in Chapter 3.

The second part of the thesis presents the proposed MR system. Chapter 4 addresses the incremental development of autonomous multi-UAV systems by proposing *MR simulation Architecture for development of Multi-Robot systems* (MiRAMuR), and finally, presents approaches of increasing the simulation fidelity with the help of external simulators. Chapter 5 introduces a test-bed framework implementing the MiRAMuR concepts for UAS oriented research and development. Details on the communication in the MR simulations are presented in Chapter 6—specifically, the design of the communication subsystem and the problem of wireless network simulation within the MR system. Finally, Chapter 7 targets the problems of modeling the MR simulation's environmnent.

Applications and utilization of the proposed MR system are discussed in the third part of the thesis. In Chapter 8, the effects of the wind on the flight trajectory are addressed, in particular, the problem of planning in horizontal wind. The presented algorithm can be then used for high-fidelity entity modeling. Chapter 9 addresses the problem of updating the MR environment; the problem of the wind map refinement by persistent measurement is formulated. Chapter 10 presents details on three case studies where the proposed methodology was used for the development of various aviation systems.

Finally, in the last part of the thesis, we conclude the thesis and discuss possible future work in Chapter 11.

1.6 Terms Disambiguation

Throughout this thesis, we freely interchange the terms UAV, entity, and agent when appropriate, since we assume autonomy of each aircraft in the complex multi-UAV system and their interconnection within a multi-agent system.

Chapter 2

From Simulation to Real World

Development and operation of autonomous multi-UAV systems come with high risks and high costs. Thus, in times of development or modifications, it is a good practice to first verify the systems in the simulation with the help of computational models and various simulation frameworks. There are many simulation tools that individually allow modeling of almost every part of a real autonomous UAS, being it simulators of the flight dynamics, terrain, wind fields, communications, etc. They can be used for a simple testing of a separate subsystem and its behavior, training of system operators by focusing the training activity only on a part of the complex system, or to study the interactions among multiple entities in more complex multi-UAV systems.

However, in the later stages of the system development, the system needs to be verified in simulations of high fidelity to prevent problems caused by real-world phenomena and their combinations that are difficult to be modeled, e.g., problems such as localization errors, unreliable communication, etc. Moreover, some physical phenomena may be difficult to be modeled because of complex and computationally-demanding simulation methods. The system is usually deployed on hardware assets and tested outside the software simulations. There are multiple frameworks that simplify the process of robotic system deployment and testing. However these hardware deployed tests tend to be costly in terms of time and money, and come with significant risks of failure. To address this problem, MR simulations can be used for incremental system development, postpone the full hardware deployment and thus lower the development costs and risks.

This chapter first presents the crucial aspects of the real-world deployment that are difficult to be modeled in software simulations and need to be taken into account for correct verification of the overall system functionality. Common approaches to modeling these aspects and dealing with them during the real-world deployment are presented. Later, the currently used approaches to the development of UAS are summarized, together with an overview of the main frameworks used for development of robotic systems. Finally, the concept of mixed-reality and MR simulations is introduced and existing state of the art approaches are presented.

2.1 Real World Modeling

During the process of moving from software simulations to the full hardware deployment in the physical world, it is necessary to deal with various kinds of challenges and issues characteristic for the real-world applications. These aspects are difficult to be modeled and if handled improperly, they may influence the performance of the the whole unmanned system during the deployment. In the following, a brief overview of these aspects and the problems they present is presented, together with the effects of these aspects on the performance of the final system. Moreover, the works that have addressed these issues up to now are presented. The main real-world aspects that need to be addressed are computational units modeling, localization uncertainty, noise in payload sensors, communication issues, energy consumption and environment modeling, and the scalability of the whole system.

2.1.1 Computational Unit

Autonomous UAS should be equipped with on-board computational unit to be able to process sensory data, react on unpredictable situations, or autonomously update its plan if necessary. Even though the performance of embedded computers is improving, some operations, such as image processing and object recognition are still too much resource demanding to be solved on computers that would be able to fit into small UAVs. Slow execution of some operations can have significant effect on the overall behavior of the multi-robotic systems. It is thus convenient, in later development stages, to either use the target computer, or emulate its resources to verify the correct functionality of the system.

There are studies dealing with emulation of the limited computational power, e.g., the authors of [35] present a resource-constrained sandboxing approach that can enforce qualitative and quantitative restrictions on the system resources used by application programs, while providing these programs with soft guarantees on the fairness of the resources. Authors distinguish between implicit and explicit requests to the Operating System (OS), and show that resource utilization can be constrained either by intercepting explicit requests to the OS, or by monitoring the frequency of implicit requests. Similar approach is described in [111]. Other approaches try to enforce qualitative and quantitative restrictions on resource usage by relying on kernel support [91], binary modification [202], or active interception of the application's interactions with the operating system [56].

In some cases, the computationally demanding operations could be run on the ground based computers, while exchanging critical data with the UAVs via communication links. However, there are cases when the on-board data processing is necessary.

- *Data size* – some data can be too large to transmit efficiently (e.g., data from camera sensors).

- *Transmission delay* can cause erroneous function of algorithms (e.g., in a case of camera based guidance and stabilization).
- *Multiple data sources* – in a case of multi-UAV systems it is necessary to deal with wireless media access issues.
- *Independence* – high levels of autonomy require independence on other systems.
- *Unreachable GCS* – during large scale operations, the GCS may be out of range of the UAS and thus the communication link unavailable.

When deploying software algorithms on the target hardware platform, it is important to know whether the requirements of these algorithms can be satisfied by the run-time environment offered by the platform, otherwise the algorithms may produce unpredictable results.

2.1.2 Localization Uncertainty

In the real world, it is necessary to address imprecise localization of entities. Every sensor has to deal with errors in its precision and accuracy and every sensor is subject to noise to some extent. As for localization sensors, UASs most commonly use combination of *Global Navigation Satellite System* (GNSS) and *Inertial Measurement Unit* (IMU) sensors sometimes enhanced with barometric altimeter, laser, infra-red, or sonar range-finders, or vision aided localization methods.

The major error sources in IMUs are biases, and scale errors related to non-orthogonalities of the axes that cause unstable errors in positions, velocities, and attitudes. The growth of these errors depends on the quality of inertial sensor used. The inertial sensor errors can be classified into two types, deterministic (systematic) and random [130]. The deterministic error sources can be removed by specific calibration procedures [143]. The random errors primarily include the sensor noise with a high frequency and a low frequency component [200] and can be modeled using random processes, such as random constant, random walk, Gauss-Markov or periodic random processes [130]. Zhang and Knoll [213] analyzed and modeled the systematic error and bias with respect to their statistic properties during data collection from heterogeneous sensors such as GPS and radar.

GNSS based localization is subject to three main sources of errors [2], [72].

- **Satellite errors** – *satellite clocks* are very accurate, but they do drift a small amount. A small inaccuracy in the satellite clock results in a significant error in the position calculated by the receiver. These errors range is ±2 m. Similarly *orbit errors* are caused by small variations of satellite positions from predicted orbits and produce errors of ±2.5 m in the position estimation. Satellite errors can be compensated for by downloading additional information from *Spaced Based Augmentation System* (SBAS) or *Precise Point Positioning*

(PPP) service providers, or by using *Differential GNSS* (DGNSS) or *Real-Time Kinematic* (RTK) receiver configuration.

- **Atmosphere induced errors** are caused by various signal delays in Ionosphere and Troposphere influenced by solar activity, humidity, temperature, etc. These delays result in a significant position error, ±5 m for ionospheric delay and ±0.5 m for tropospheric delays. DGNSS and RTK systems are able to compensate for these kind of delays.

- **Receiver errors** are caused by *receiver noise* and *multipath reception*. The receiver noise refers to the position error caused by the GNSS receiver hardware and software. High-end GNSS receivers tend to have less receiver noise than lower cost GNSS receivers. Multipath reception occurs when a GNSS signal is reflected off an object to the GNSS antenna. The reflected signal travels further and arrives at the receiver slightly delayed, causing the receiver to calculate an incorrect position (±1 m).

The extend of the localization error can be decreased by utilization of more precise IMUs, DGNSS or RTK GNSS units or by integration of multiple localization systems and using filtering methods such as particle filters [148], [190], or extended Kalman filter [95], [117]. However, even when these precautions are taken and the localization error is minimized, there may still occur inaccuracies in the plan execution caused by noise in actuators, or by external factors such as strong wind gusts.

As for modeling errors in the plan execution, two types of errors can be distinguished [164]: (i) *time-related errors* (Δ_T); and (ii) *distance-related errors* (Δ_D), see Figure 2.1. The distance-related error is a deviation of the UAV from the planned spatial trajectory caused by sudden wind gusts, imprecise control, or inability to follow the planned trajectory. The time-related error is the deviation of the UAV from the time plan caused by imprecise autopilot velocity control, or by accumulating small delays during repairs of small spatial deviations from the plan.

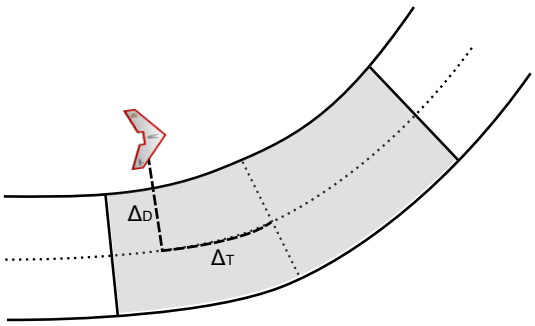


Figure 2.1: Two possible error types in the flight plan following: time-related error (Δ_T), and distance-related error (Δ_D). The gray area corresponds to the safety zone around the plan position.

To address these errors, we have proposed *safety zones* around the UAVs [164]. Generally, the worse the flight plan execution performance, the bigger the safety zone needs to be in order to keep the aircraft far apart from obstacles or other aircraft. These zones are asymmetric and use two safety ranges around expected plan position – (i) *safety time range* (s_T), and (ii) *safety distance range* (s_D), see Figure 2.2. The range s_T is given by the maximum allowed Δ_T, and range s_D is specified by the maximum allowed Δ_D. When the plan execution error is larger than these ranges, trajectory replanning or plan-repair methods [100] should be scheduled. This safety zone can be used during sense-and-avoid, obstacle avoidance and trajectory planning so that it is wrapped along the planned trajectory and should not cross any obstacle, no-flight zone nor other planned trajectory in time and space.

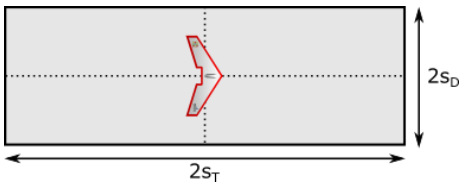


Figure 2.2: Two different safety ranges: *safety time range* (s_T), and *safety distance range* (s_D).

2.1.3 Sensory Payload Noise

Similarly to the sensors mostly used for localization discussed in the previous section, also the payload sensors are subject to different types of errors. These errors depend on the type of the payload. Various types of cameras (CCD, thermal, multi-/hyper-spectral), or radars images are subject to image noise, which is a random variation of brightness or color information in images. These errors are discussed in more detail in [59]. Cameras are also subject to optical errors producing image distortions (such as barrel, pincushion and complex distortion). The different kind of lens distortions and lens aberrations have been researched for decades. An extensive review of the lens distortion itself is given by Slama in [178]. Fortunately, these distortions can be corrected by several methods, either using the mathematical model of the distortion, or by estimation of the distortion model parameters using various calibration methods.

Also, *Light Detection and Ranging* (LiDAR) sensor data can be influenced by noise (e.g., from multi-path returns, reflections from object edges, etc.). Some portion of this noise can be modeled. Meyer et al. in [120] used first-order Gauss-Markov error model to provide noise model to Robot Operating System (ROS) framework [151]. They estimated each simulated measurement $y(t)$ at the time t as

$$y = \hat{y} + b + w_y, \tag{2.1}$$

$$\dot{b} = \frac{1}{\tau} b + w_b, \tag{2.2}$$

where $\hat{y}$ is the true value of the measured phenomena, b is the current bias, and w_y and w_b are independent, zero-mean white Gaussian noise variables. w_y is additive noise acting directly on the measurement and w_b represents the characteristics of random drift with the time constant τ.

The result of not taking these properties of the sensory payload into account could lead to situations where (i) in reality the objects of interest are outside of the sensor's field of view even though they were regularly sensed by the simulated entities; (ii) objects observed by real sensors could be severely distorted and thus not recognized; (iii) the noise in the data can hinder the objects of interest so they are not found. These problems pose a real issue during the deployment of algorithms into the real-world applications if they are not taken into account during the development.

2.1.4 Communication

There are three main reasons for having a communication subsystem in unmanned aerial systems: (i) remote UAS control and monitoring; (ii) knowledge and team management; and (iii) streaming of sensory payload data. One or more communication links can be used cover these requirements, depending on the target application demands.

The stability and reliability of the remote UAS control and monitoring link depend on the level of autonomy of controlled UAS; however, for safety reasons, higher standards are recommended. The second link type is required for teams of multiple autonomous aircraft. The applications for this link can range from cooperative collision avoidance and team coordination to message relay. Finally, some systems may need a link for streaming sensory data to the operator. This may be the case for missions where online feedback is required, such as patrolling or *search and rescue* (SAR). This last link type does not need to be as stable and reliable as the previous two, but needs to have sufficient throughput.

In simulations, the fidelity of communication models can influence the system scalability. Communication delays and limited communication range can affect the overall system performance. Limited channel capacity can cause that some messages are received with large delays or may not be even delivered at all. This may significantly change the system behavior. Precise communication models are thus necessary for high fidelity simulations.

Currently, there are several technologies that can provide the communication to UASs. Each of these technologies behaves differently and needs to be modeled in a different way. There are *satellite communications* (SATCOMs), *Free-Space Optical* links (FSO), *Radio Frequency* (RF) modems, hybrid RF/FSO communications, and mobile telecommunications technology (3G-5G). Each of them has some advantages and disadvantages for utilization in the area of UAS.

- **SATCOMs** provide reliable connection beyond line of sight, they are usable almost everywhere around the globe, except of extreme altitudes not covered by communication satellites. On the other hand, they require relatively large and heavy antennas, cause large communication latency (∼2 s) and they are expensive both to obtain and use. These links use radio frequency signals on frequencies ranging from small (1 GHz for L-band) to high (50 GHz for V-band). The higher frequency the higher data rates are available but the signal quality is more influenced by the atmosphere. The atmospheric signal fading was modeled, e.g,. by Castanet et al. [30].

- **FSO links** provide high throughput (∼10 Gbps) over large distances (∼10 km), are energy efficient, and difficult to intercept. On the other hand, FSO links provide only point-to-point communication capabilities, require a precise mechanism to track the target UAS and their correct functionality is subject to physical obstructions and atmospheric phenomena that can significantly degrade their performance [22]. These communication links were in the past used for ground-to-air connection where one of the nodes was stationary, but recently a modeling of FSO links between mobile platforms is being intensively studied [94], [96].

- **RF modems** are currently the most frequently used communication links for small UAS because of their relatively low price and light weight. They provide stable and reliable channel in local area, which, depending on the antennas used, can reach up to several kilometers providing data rates up to approx. 54 Mbps. RF signal is radiated in straight lines but unlike the light, bulk of its energy is carried in the space around the straight line connecting the transmitter and the receiver. For a successful data reception, the strength of the transmitted signal needs to be greater than the background noise at the receiver. The background noise can be caused either by devices operating in the same frequency band (such as radios, microwave ovens, or GSM phones), by other transmitting stations, or by a multi-path propagated signal that interferes with itself. RF signal is influenced by various physical processes such as scattering, diffraction, refraction and reflection, and is attenuated along its path. These phenomena influence the communication radius and reliability. They have been intensively studied and several path loss models have been proposed that describe this behavior on various levels of detail [152].

- **Mobile telecommunications technology** – currently, the GSM signal coverage is enabled thanks to a network of *Base Stations* (BSs) located at populated areas around the globe. The most recent generation of GSM technology (4G) can theoretically allow mobile units to reach download speeds of 100 Mbps and upload speeds of 50 Mbps and ten-times higher speeds for static units. It theoretically provides communication latency of less than 50 ms and provides thousands of connections for each cell. Field tests, however, report much worse characteristics for regular users [139]. The advantage of this communication

technology for UAS is the small size, light weight and low price of the devices. On the other hand, the major problem of this technology is its coverage since it is limited only to densely inhabited areas and the signal coverage of the BSs targets on covering mainly the ground-based nodes. The functionality of GSM technology can be modeled up to a certain level of precision using the same simulators as in the case of RF modems. If simplified, in the GSM network, there are multiple nodes that work as communication relays (BSs) and each communication between user nodes has to be translated between these relays. Further, Teng et al. [189] conducted field measurements showing that a typical cellular deployment could result in low-coverage areas for UAS and formed a combined propagation model that more accurately reflects reality for the tested scenarios.

There are many existing wireless network simulators [112] that allow to study problems connected with constrained communication channel. Some of these simulators can be exploited by third party simulation systems to increase their fidelity. In general, no network simulator is accurate. All simulators work with some abstraction of the real-world which in its nature causes a certain level of imprecision. When really good accuracy is needed, experiments should be conducted on the real devices, using testbeds. As studied by Heidemann et al. [79], a lack of detail may have serious impact on the obtained results and may cause the results to be wrong or misleading in two ways. Either they are simply incorrect, or they are technically correct but provide an answer to irrelevant parts of the design space, and thus making the results inapplicable.

Communication subsystem and its modifications for mixed-reality simulations is further discussed in Chapter 6.

2.1.5 Energy Consumption

Most of the currently used unmanned systems are significantly limited by their flight time. No matter what their power supply is, their energy consumption is large and energy supplies do not allow efficient energy storage for long flight times. There is an option to use ground based recharge stations [93], [175]; however, the times needed for a battery recharge are not negligible and increase the time where the UAV is not in the air executing its mission. Another option to increase the flight time of UAV is to use tethered systems that can use the tether to utilize the ground-based power sources. This vastly increases the flight times of UAVs but on the other hand limits their operation space. Recently, automatized battery recharge stations [186], [192] start to appear as a ready to use solution.

Also, flying in wind updrafts can extend flight endurance and range of aircraft that rely on energy in the form of wind currents. The flights of this type are known as soaring and there are two kinds - *static* and *dynamic soaring*. Static soaring involves extracting energy from rising air, such as thermals, while dynamic soaring utilises energy from spatial wind gradients.

Chakrabarty and Langelaan [33], [34] combined these mechanisms with the use of energy maps and heuristic search for a long-distance trajectory planning in a known wind-field. In simulation, algorithms for simultaneous exploration and exploitation of unknown wind-field were proposed [42], [109], [110]. In practise, successful results of autonomous thermal soaring were attained as well [5], [54]. Autonomous soaring has recently started to be used to enhance the efectiveness of information gathering tasks [46], [134]. No matter what the energy management strategy is, the energy consumption and its limitations on the UAV flight has to be taken into account in high-fidelity simulations, especially for estimation of flight times and following adjustments of missions.

However, it is not easy to acquire the energy consumption model. There are many different types of UAS constructions that may use various energy sources and modes of flight. The energy consumption can be influenced by propellers, weight, used maneuvers, external conditions, etc. It is not possible to find a universal model and thus only vehicle-specific models with limitations on maneuvers and external conditions have been proposed so far. For example, Ware and Roy [204] flew a quadrotor in a wind tunnel chamber with a Vicon motion capture system to measure the power consumption as a function of the air speed. Zeng and Zhang [212] derived the propulsion energy consumption model of fixed-wing UAVs in a level flight and utilized the model for trajectory optimization based on communication throughput and energy efficiency. Di Franco and Buttazzo [51] performed experiments with IRIS quadrotor aimed at understanding how the energy consumption is affected by the different operating conditions, such as speed, horizontal and vertical accelerations. They utilized these measurements to adjust and improve coverage planning algorithm for *Vertical Take-Off and Landing* (VTOL) UAVs.

Another mechanism that describes an energy consumption model is used in civil aviation. *Base of Aircraft Data* (BADA) [137] models used by European Organisation for the Safety of Air Navigation use so called *Total Energy Model* (TEM) that compares the work of forces acting on aircraft during changes in potential and kinetic energy. These models can be used to calculate thrust, drag and fuel flow as-well-as to specify nominal cruise, climb and descent speeds. Apart from that, they, for example, define classes of aircraft and their weights, aerodynamics. All of these models are in detail described in the BADA User Manual [136].

2.1.6 Environment

For high fidelity simulations it is necessary to model properties of the external environment that could influence the plan execution. The most flight critical properties of the environment are the terrain and wind model.

Depending on the required fidelity level, terrain can be represented as a simple set of obstacles wrapped in simple geometric primitives or a precise model of the terrain's 3D surface created by methods of computer graphics [158]. The more precise model is used in the simulation, the better trajectories in terms of distance traveled can be planned. In a case of too rough models, some

areas may become unreachable for UAVs, even there would be enough space in reality. To build a precise 3D environment model, it is necessary to conduct real-wold measurements. However, this is a complex task with many challenges [191]. The most significant are the following.

- Statistically dependent measurement noise caused by accumulation of control errors
- High complexity of the environment that requires large amount of data for 3D map creation
- Correspondence problem (data association problem) of determining whether different measurements correspond to the same physical object
- Dynamics of the environment that adds yet another way in which seemingly inconsistent sensor measurements can be explained
- Problem of robotic exploration that generates the robot trajectory for building a map

Wind modeling is an ongoing and difficult challenge [36], [77], [131]. *Computational Fluid Dynamics* (CFD) solvers can provide 3D wind field models of various environments; however, their precision and forcasting capability are strongly dependent on the quality of initial conditions [144]. These wind models can be significantly enhanced by data assimilation, i.e., by incorporating real measurements into the state of the numerical model [85]. The impact of wind effects on trajectory planning is addressed in [118] by iteratively solving a no wind case problem with a moving virtual target, or in [188] and [187] by generating candidates of smooth time-optimal paths between the initial and goal point. These approaches search only for a horizontal plan between the start and final configurations and do not take possible obstacles into account. Selecký et al. [172] combined the AA* algorithm [199] with the technique used by McGee et al. [118] and proposed a computationally effective algorithm for planning in locally constant wind, capable of obstacle avoidance and planning in 3D. Cobano et al. [43] used a genetic algorithm and Monte-Carlo method to deal with wind effects.

The so far mentioned methods studied effects of a constant uniform wind, but this is not the case of the real world. Wolf et al. [205] and Al-Sabban et al. [3] use Gaussian distribution to determine uncertainty in the time-varying wind fields and use *Markov Decision Process* (MDP) to plan a path based upon the uncertainty of Gaussian distribution. Authors of [156] applied evolution-based path planner that made use of wind information obtained from actual weather databases. Other approaches used A* algorithm with various heuritics functions. Garau et al. [62] studied planning for underwater vehicles and used heuristics based on Euclidean distance between a point in space and the final destination divided by the maximum possible nominal speed (maximum water current speed plus the vehicle speed). Ware and Roy [204] studied the problem of finding minimal-energy trajectories for a quadrotor flight in urban environment with wind using a CFD solver.

The problems of planning in the wind and increasing the simulation's fidelity by refinement of the wind field are further discussed in Chapter 8 and Chapter 9, respectively.

2.1.7 Scalability

The increase of the simulation's fidelity usually comes with the decrease of its scalability. Distributing the control logic helps to address the limited computational capacity but there are constraints on the scalability caused by limited radio frequency bandwidth, increased cost of hardware, and effects on the safety of operation. In case of all robotic systems, the cost of hardware can be a strong limitation for scalability. In the development phase, it may be difficult or impossible to obtain more than a few robots fully equipped with all the hardware necessary for the target application. Another problem of multi-UAV systems is their safety of operations. In a case of accident, they may crash and cause damage or injury. The risk is increasing with more aircraft simultaneously operating in the air. Even though the reliability of their hardware and firmware is increasing, the primary purpose of simulations is to test and validate a new behavior which may always produce unexpected results. Finally, there is also a problem with legislation. In many countries, it is not legal to simultaneously operate more than one unmanned aircraft by one certified *Pilot-in-Command* (PIC). This means that there have to be as many certified PICs as UAVs which may not be feasible in case many unmanned aircraft.

2.2 Approaches to Development of UAS

Simulations can significantly speed up the development process of robotic systems and save costs on hardware as they allow to early discover, isolate, and correct potential implementation issues. According to Mutter et al. [127] there are two approaches to robotic system development: (i) the *Software-in-the-Loop* (SIL), where all system components (sensors, actuators, airframes, communication, etc.) are simulated using mathematical models; and (ii) the *Hardware-in-the-Loop* (HIL), where only some system parts are simulated and others are replaced by particular physical hardware. Examples of simulators using the SIL approach can be found in [50] and [63]. The HIL approach is taken, e.g., by Goktogan et al. [68], [69], Day et al. [48], and Pizetta et al. [149] to name few.

2.2.1 Frameworks for Development of Robotic Systems

In the recent years, we have witnessed an appearance of many tools for the simulation of robotic models, sensors, and control in virtual environments. This effort is motivated by the proliferation of modern robotic applications in which robots are required to operate autonomously. Simulation of robotic systems is an essential tool in teaching, research, and development. It helps architecture designers in managing the complexity of embodied agents. Without simulation tools it would be hard to design and test feasibility of various complex robotic systems. These tools enable simulation of various layers ranging from artificial environments, robot models and their dynamics, to simulation of robot sensors or node to node communication. Further,

they usually provide models of existing hardware robots and sensors that can be used for better code reuse and hardware portability.

There are both open-source projects, where some of the most frequently used are Carnegie Mellon Robot Navigation Toolkit (Carmen) [125], Open Dynamics Engine (ODE) [179], or Gazebo [99]. There are also a number of proprietary, commercial tools for mobile robot simulation, such as MS Robotics Studio [150], Webots [114], or V-REP [155]. It may be hard to choose the right tool. Some of them are user friendly but provide only basic kinematic models; others include accurate physical simulation engines, but depend on many external packages which make them difficult to use and maintain and reduce the reusability of the code. Yet other tools may have high initial costs or periodic license maintenance fees to utilize the software. There are several studies [32], [108], [193] that compare these simulators from the point of view of their physical fidelity, quality of documentation, ease of use, and number of features they provide.

Apart from the simulation tools, there are robot frameworks—*robotics middleware*—that can be used to integrate these simulators together with control algorithms into a complex system that enable simulations of multiple entities with their sensors and actuators. Some of these frameworks even allow for deployment of algorithms on-board robotic platforms. Some of the most significant frameworks currently used are the following.

OROCOS - Open Robot Control Software [26] is one of the oldest open-source frameworks in robotics, being under development since 2001. It provides a hard real-time capable component framework—the so-called Real-Time Toolkit (RTT)—and it aims to be independent from any communication middleware and operating system. Components in Orocos exchange information primarily through anonymous publish-subscribe system.

ROS [151] is an open source robot software platform designed with the primary goal of enabling software reuse. Nowadays, ROS is one of the most frequently used mobile robot frameworks. It is intended to be a thin architecture, providing sufficiently few constraints as to be integrated with software written for other platforms, such as Orocos, or Gazebo simulator. There are many packages containing application-related code, that can be reused in ROS based systems (e.g., libraries for hardware control, computer vision or sensory data processing). Beside other features, it provides a structured communication layer above the host operating systems. A system built using ROS consists of a number of processes (*nodes*), potentially on a number of different hosts, connected at runtime in a peer-to-peer topology communicating by passing *messages* or by publishing *topics*.

ROCK - Robot Construction Kit[92] - is an alternative to ROS framework. Most of the algorithms and drivers in Rock are independent of the component layer for better code reuse. As ROS is mainly used as a thin communication library, Rock's component layer is thicker, to provide the tools necessary to scale to complex systems. While ROS uses a topic based communication model, Rock is connection based that makes it easier to control the flow of information in the system. Further, real-time applications can be implemented in Rock more

easily since it is directly based on the Orocos RTT. ROS also provides support for Orocos in real-time domains, but requires special interfacing.

2.2.2 Development of Multi-UAV Systems

Apart from robotics environments that provide support for low-level control development and simulation of autonomous systems, research in *Multi-Agent Systems* (MAS) aims on the high-level control features such as planning, coordination, cooperation, and task allocation in teams of multiple autonomous robots.

In the past, there have been various approaches to develop and deploy complex multi-UAV systems. For example Šišlák et al. [198] developed AgentFly system used for simulating UAVs and civilian air traffic. It allows complex coordination and cooperation of the aircraft agents but cannot control hardware assets. Bürkle et al. [29] presented a deployment of control mechanisms for teams of hardware VTOL micro-UAVs that were capable of on-board planning and cooperation on mutual tasks. However, there was no collision avoidance among the UAVs and the system did not allow co-existence of hardware UAVs together with simulated ones. Scerri et al. [162] designed a system for deployment of multi-agent technology for UAV coordination to industrially developed applications. Their development approach moved the system from a purely virtual setting directly to the fully hardware deployed one, which would not be suitable for more complex systems. Most recently, Sanchez-Lopez et al. [159] present a multipurpose system architecture for autonomous multi-UAV platforms with basic high-level planning features which, however, did not allow for complex distributed multi-UAV planning.

Göktoğan and Sukkarieh [68] presented the concept of using some kind of augmented reality for development of multi-UAV systems. They use the Real-time Multi-UAV Simulator (RMUS) developed at the Australian Centre for Field Robotics [67] for multi-UAV simulations and also in real flight trials as a mission control system. In their framework, the hardware and virtual UAVs can interact only on the level of sensory data exchange. Moreover, authors do not provide much design details on this topic. Later, the same authors presented a methodology for development of multi-UAV systems [69] using the waterfall model of development process [23]. This methodology however does not account for the possibility of incremental development using the augmented reality approach. This was not the case of the work [87] where Jakob et al. came up with a concept of incremental multi-level development of complex systems by a use of mixed-reality test-beds. Unfortunately, apart from introducing the concept, the authors gave no further details on how to carry out such incremental development. Similarly, this thesis addresses the problems of multi-UAV system development by using mixed-reality test-beds while avoiding the drawbacks of [87].

2.3 Concept of Mixed-Reality Simulation

2.3.1 Reality-Virtuality Continuum

One of the first notions of mixed-reality concept was presented in the work of Milgram et. al [122], where the authors defined it using Reality-Virtuality continuum (see Figure 2.3). According to the authors, real environments (consisting solely of real objects and entities) are presented at one end, and virtual environments (consisting solely of virtual objects and entities) on the other end of the continuum. The authors refer to terms *Augmented Reality* (AR), as a case where the real environment is augmented by means of virtual objects and *Augmented Virtuality* (AV), as the virtual world that is augmented with feeds from the real world. Since it may eventually become hard to distinguish whether the primary environment is predominantly real or virtual, authors classify anything between the extreme cases as the *Mixed-Reality* (MR) environment, i.e., an environment where the real and virtual objects coexist and interfere.

Work [122] and several ones that followed deal mainly with visualisation displays and overlays, e.g., for use in robot teleoperation [25], [123]. The AR was utilized to display the virtual and simulation data-feeds into the real-world camera feeds, e.g., an overlay for informing about the actual state of robots and their communications [208], or their heading [47]. On the other hand, the AV was used to enhance virtual environments with real sensory feeds, e.g., depicting objects newly detected by real sensors in a pre-constructed virtual environments [39] or for visualization of spatial information of robots in dynamically constructed environments based on distributed sensory readings [8]. Exiting MR systems and methods using visualisation displays and overlays were summarized by Costanza et al. in [44]. The concept of "mixing" physical and virtual objects and entities also occur in works dealing with *Shared Reality* [183], or *Merged Worlds* [126] that overlap with each other with fuzzy boundaries.

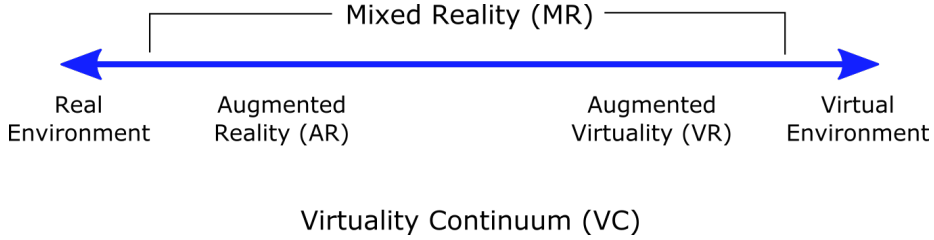


Figure 2.3: Simplified representation of Milgram's Reality-Virtuality continuum [122].

2.3.2 MiRA Cube

The one dimensional classification of environments and agents, as presented by Milgram et al. [122], was later extended by Holz et al. [82] who created the taxonomy for the Mixed-Reality Agents (MiRA), defined as agents embodied in a MR environment. The authors came with three-dimensional classification using MiRA Cube Taxonomy as shown in Figure 2.4. Two

axes are defined that span from the physical to the virtual extreme (*interactive capacity* and *corporeal presence*) and one axis that differentiate *weak agents* (those possessing base capabilities of autonomy, social ability, reactivity, and pro-activity), and *strong agents* (those incorporating also reasoning abilities such as knowledge, desires, or intentions) based upon the notions of agency introduced by Wooldridge and Jennings [206]. The interactive capacity is defined as the agent's ability to sense and act on the virtual or physical environment and its corporeal presence is understood as its degree of the virtual or physical representation. Such taxonomy should facilitate the ordering of agents in terms of their degree of embodiment in the shared MR environment. We extended this taxonomy to reflect the update capacity of agents. More details are provided in Section 7.6.

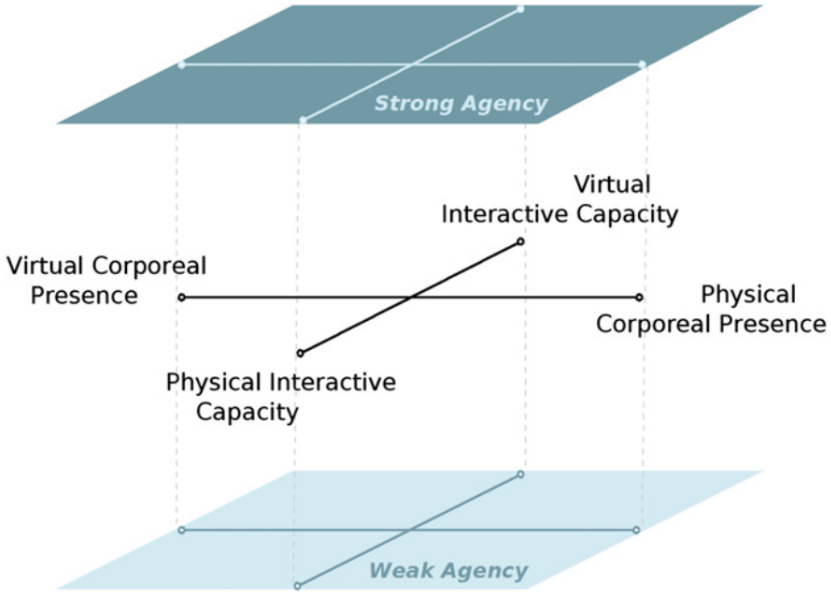


Figure 2.4: A diagram of the MiRA Cube Taxonomy spanned by the three axes - *agency*, *corporeal presence*, and *interactive capacity* as defined by Holz et al. [82].

2.3.3 Mirror Worlds

Ricci et al. [153] came with the concept of *Mirror Worlds* (MWs), where they integrate technologies of multi-agent systems, mobile mixed/augmented reality and the *Internet of Things* (IoT) and provide a framework for their investigation. MWs create an environment based on a bidirectional augmentation of the physical and the virtual reality. According to the authors, MW is a digital world shaped in the terms of a multi-agent system, situated into some virtual environment which is coupled to some physical environment, augmenting its functionalities and the capabilities of the people that live or work inside it.

The authors use notion of *artifacts*, as defined by Omnici et al. [140], for modeling basic non-autonomous environmental objects which can be shared, observed, and used by agents. Figure 2.5 shows the concept of the MW as an environment with a set of artifacts, part of

them directly *mirroring* artifacts in the real-world. Mirroring represents a form of coupling, where an action on artifacts in the physical world causes changes in artifacts in the mirror world, perceivable by software agents. Similarly, an action on artifacts in the MW can have an effect on artifacts in the physical world, perceivable by real robots or humans. The MW concept is, however, inconvenient for UAS applications since it requires maintaining a continuous and consistent coupling among entities and simulation engine which is not always possible or preferred in UAS scenarios.

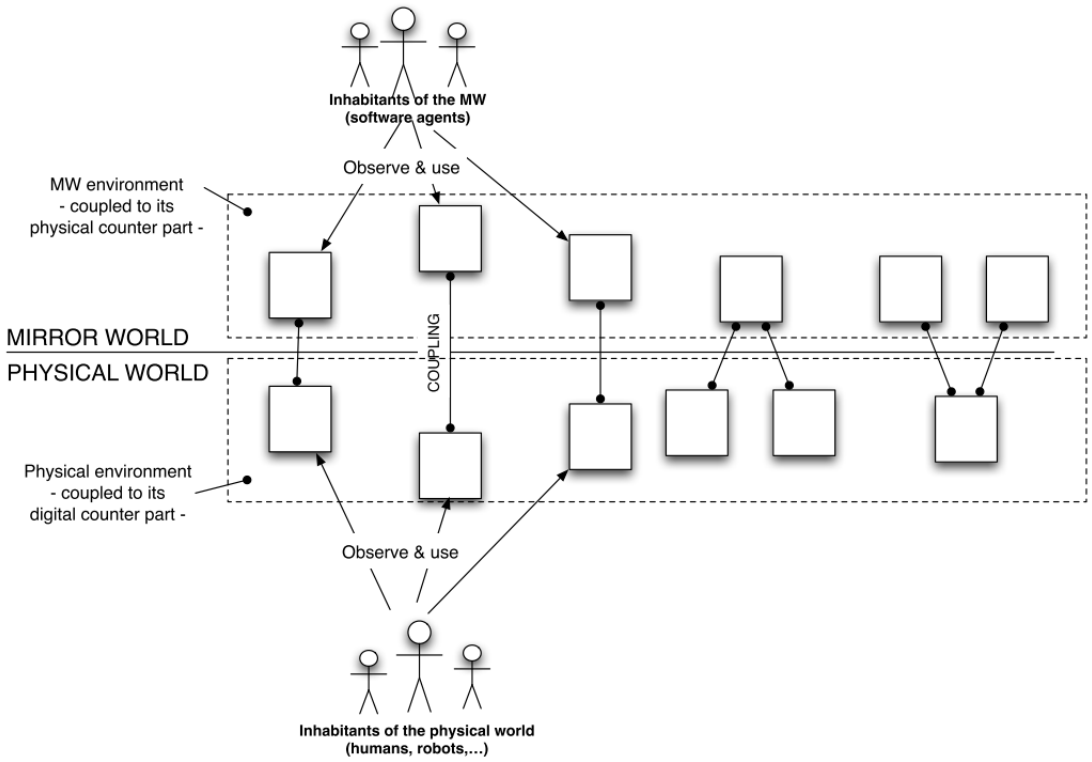


Figure 2.5: Concept of Mirror Worlds as defined by Ricci et al. [153].

2.3.4 MR Simulations

One of the first notions of MR simulations used for the development of mobile robots has been presented by Chen et al. [40]. The authors used an MR simulation tool based on the 3D robot simulator Gazebo. In a single robot scenario, they demonstrated advantages of combining a real entity and simulated objects for development of a practical robotic system. Later in [41], the same authors showed that MR simulations can help to provide efficient testing tools, identify improvements to the UAV controller, and provide valuable insights into the UAV system performance that would be otherwise difficult to obtain in real-world tests. Recently, Honig et al. [83] used MR simulations together with various robotic simulators and showed that these simulations can provide many advantages for research and development in robotics. The authors supported their findings by several use-cases with Crazyflie nano-quadcopters.

Jakob et al. in [87] came up with a concept of incremental multi-level development of complex systems by a use of MR test-beds. The authors defined levels of virtualization as the extent to which parts of the target application setup are replaced with synthetic computational models. The test-bed configuration is defined as a vector $\sigma = (s_0, s_1 ... s_n)$, where s_i is the number of entities modeled at the level of virtualization l_i (see Figure 2.6). Further, the authors proposed to begin development with a fully virtual system where all agents purely simulated and then iteratively move to the configuration with no virtualization level. The direction of the movement through the space of test-bed configurations depends on the cost of the particular iteration steps.

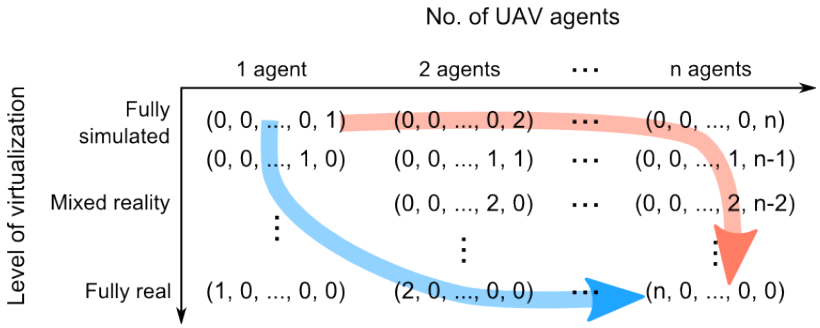


Figure 2.6: Space of test-bed configurations and two strategies of iterative development.

Determining which parts of the system to approximate with computational models is difficult due to the large number of involved entities, subsystems, and their dependencies. Unfortunately, the authors, do not give any details on how to carry out such a incremental development nor provide details on how to estimate the costs in particular development stages.

This thesis addresses the problems of high-fidelity simulations in the UAS domain discussed in this chapter. The MR simulations and incremental development approach is utilized to balance cost and fidelity of the multi-UAV testbeds throughout the development process. We introduce a methodology and a system architecture for incremental MR development process that allows developers to use MR testbeds of various sizes and virtualization levels in a way that maximizes the effectiveness of a development of multi-UAV systems. The efficiency of this approach is presented in multiple case studies from real deployment projects.

Chapter 3

Goals of the Thesis

This thesis addresses the problems of development and high-fidelity simulations in the multi-UAV domain mentioned in the previous chapter. It summarizes research and experience of more than 7 years of multi-UAV systems development. Here we present the main goals of the thesis.

- Propose the **concept of the MR simulations** applied to the domain of autonomous UAVs and provide the details of the **MR simulation architecture**.
- Present the **methodology of an incremental development** of complex multi-UAV systems from virtual computational models executed in the simulation to the full hardware deployment on board of the UAVs. Further, provide **the risk and cost analysis** of such development process.
- Propose a system **architecture for efficient incremental development** of complex UASs using MR simulations and validate it through an **implementation of a framework** for control of multiple autonomous UAVs.
- Address methods for **increasing the fidelity** of MR simulations in various aspects of the simulated system.
- Present the application of the proposed methodology and architecture and **demonstrate** it on case studies of development of **multiple aviation systems**.

Chapter 4

Mixed-Reality Simulation and Unmanned Aerial Systems

A development methodology for complex multi-UAV systems requires strong support for realistic evaluation and testing. In purely software systems, the functionality depends solely on the program logic. In contrast, functionality of the robotic systems is, in addition, influenced by the characteristics of the hardware and the dynamics of the environment. A reliable verification of such systems requires a testbed that approximates these factors with a sufficient level of fidelity. In general, such testbeds are those with full physical embodiment (i.e., systems with a complete set of hardware assets operating within the target physical environment). Unfortunately, field tests employing such testbeds are expensive in terms of money, time, and resources. They can be also subject to legal issues and carry substantial risks in case of malfunction. That is why these real-world experiments are impractical in the early stages of system development when quick testing and evaluation of software design is required and risk of errors is high. In these stages, it is more practical to use simplified testbeds with lower fidelity but reduced complexity, risks and costs, where parts of the system are replaced with computational models.

MR simulations have been shown to be beneficial for development and verification of complex multi-UAV systems [41], [83], [87], [169]. However, to the best of our knowledge, there is no work that would present implementation specifics of MR simulation concept, showing its benefits and challenges based on real experiments, and using it for the development of real autonomous multi-robotic systems.

In the first part of this chapter, a methodology for incremental development using MR simulation is presented together with an architecture for utilization of incremental development techniques for building autonomous multi-UAV systems and their components. Moreover, detailed risk and cost analysis of this approach is provided. In Section 4.2, suggestions for optimal development strategies for various use-cases of system verification and development are presented. These suggestions are based on our numerous experience from application projects. Finally, we present the possibilities of extending the MR simulation with specialized external

simulation tools for further increase of its fidelity in Section 4.3.

4.1 Incremental Development in MR Simulations

The herein presented Mixed Reality simulation Architecture for development of Multi-Robot systems, further refered to as MiRAMuR, has been introduced in [166]. It has a seven-layer structure that modularizes a multi-robot system into individual subsystems. These modules can be represented by a real hardware or by a software model with various levels of virtualization. By a combination of these real and virtual modules in MR simulations, we can gradually decrease the virtualization level of the whole system by adding hardware parts and thus efficiently develop various complex multi-robot systems.

There are two main parts of the development method: (i) an architecture with seven layers and various virtualization levels; and (ii) a set of development stages that determine the virtualization levels of the layers and a process of verification of the correct behavior on each level.

4.1.1 Design Layers

The MiRAMuR uses a seven-layer architecture that is shown in Figure 4.1. The architecture layers starting from the bottom are as follows.

- ***Environment layer*** represents the environment in which all the assets or agents exist. It subsumes real environment if the virtualization level of at least one agent is such that it can sense or move in the real world. It contains a simulation server that has a model of a virtual world with the defined terrain, weather or entities obtained from third party systems. The simulation server can use the information from real assets to enhance the virtual world model with real world data. More details on interactions between the Environment and Sensory layers are provided in Chapter 7.

- ***Asset Layer*** defines the *Flight Dynamics Model* (FDM) of the entities in the system and their physical behavior in the environment, i.e., how does the environmental forces affect the state of the entity. In a case of the zero virtualization level, this layer is represented by a physical airframe itself.

- ***Autopilot Layer*** represents the behavior of the autopilots. Entities can be equipped either by hardware autopilots, or by a set of algorithms that simulate the autopilot, such as stabilization, trajectory following, navigation, etc.

- ***Sensory Layer*** specifies both the flight support sensors, such as IMU, GNSS, or *Automatic dependent surveillance-broadcast* (ADS–B) receiver, and the sensoric payload, such as camera, LiDAR, etc. Either real hardware devices or their computational models can

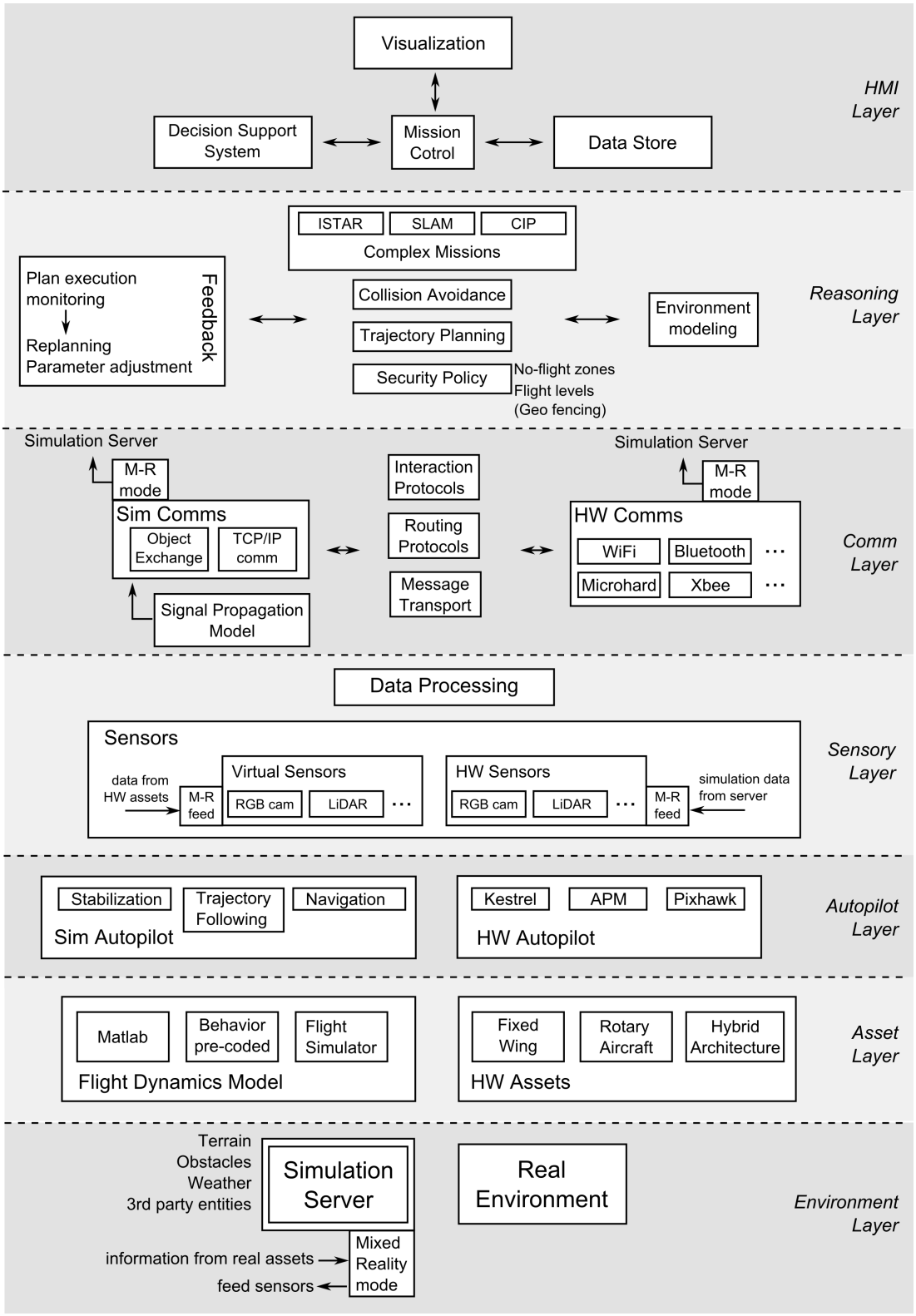


Figure 4.1: Architecture of the system for command & control of autonomous UAVs.

be used. In MR scenarios, data from simulation are passed to the real sensors providing them with augmented reality and thus increasing the scalability of the scenario. On the

other hand, data from the real sensors enhance the simulation with real-world information increasing the simulation fidelity. The information from the sensors can be used by the *Reasoning layer* for collision avoidance and team coordination or by the operator, e.g., for a reconstruction of environmental models.

- ***Communication Layer*** is used for message exchange and also defines a set of communication protocols, such as routing protocols for Mobile Ad-hoc NETworks (MANETs), various interaction protocols (e.g., FIPA protocols [60]), or protocols for message transport (handshakes, acknowledgments, etc.). Exchange of messages among entities with a different level of virtualization (e.g., between real and virtual entities) is handled by the simulation server, as described in Chapter 11.

- ***Reasoning Layer*** defines algorithms for high-level control, decision making, mission planning, team coordination, and plan execution monitoring. This is a pure software layer, and as such, it is the same for all agents independent of their virtualization level.

- ***HMI Layer*** is the uppermost layer used by the system operators as a command and control tool to assign tasks to the assets and to visualize mission data. This layer can be connected with external software (e.g., map frameworks, ground control station software, complex-event processing solutions etc.) for enhanced mission control, analysis, visualization, and other features.

The realization of the bottom five layers depends on the virtualization level of entities. It may vary from simulated models (depicted in the left side of the scheme in Figure 4.1) to the hardware equipment (depicted in the right side of the scheme in Figure 4.1). The ratio of hardware to virtual elements used in one agent determines the virtualization level as specified in [87]. Possible variations of individual levels can be seen in Figure 4.2.

4.1.2 Development Stages

As mentioned above, virtualization of the system under development can be decreased incrementally in multiple steps which lead to various development stages that differ in the utilized level of the virtualization of its system layers. Cost of these steps differ depending on the actual stage of the system, nature of the system being developed, used equipment, and probability of a failure. Basic categorization of these development stages from the simplest to the most advanced is depicted in Figure 4.3 and can be characterized as follows.

Purely virtual scenarios – In a purely virtual scenario (see Figure 4.3a), the environment of agents is represented solely by the simulation server with no mixed-reality feeds. The UAV assets are modeled by simple software behaviors and they are equipped only with simulated sensors. Communication among the agents is realized by virtual channels,

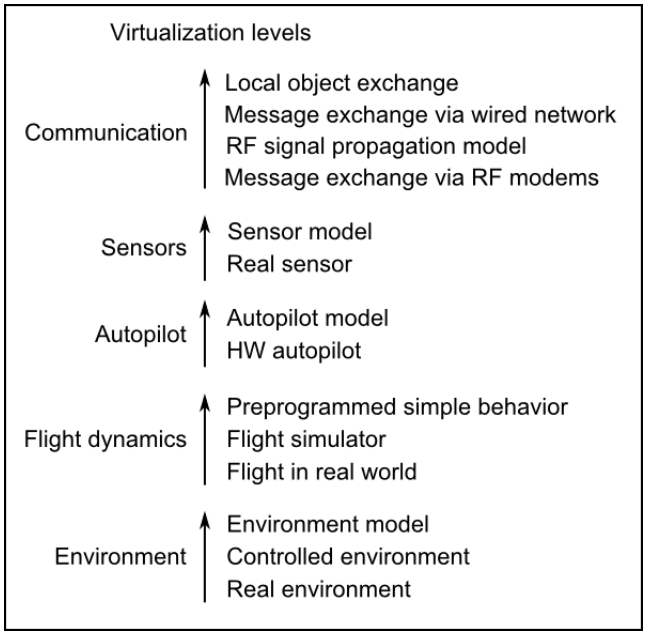


Figure 4.2: Possible virtualization levels of the system layers.

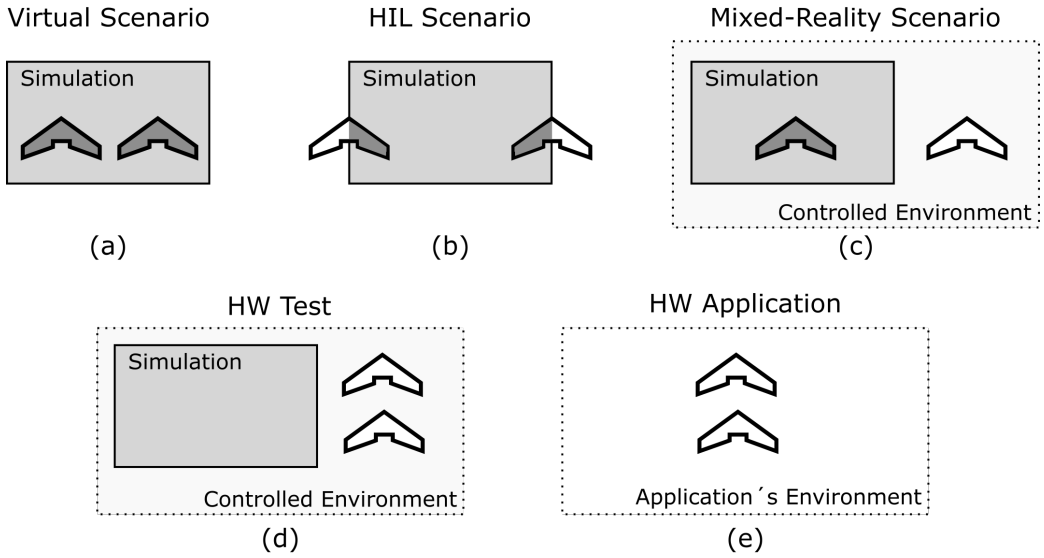


Figure 4.3: Consecutive scenario stages of a development process.

implemented as message exchange between agent processes. This configuration is suitable for early stages of development where the system fidelity is not yet important. It is used for high-level system design and study of algorithmic problems such planning, coordination, task allocation, or development of parts of the HMI layer.

This virtual scenario poses no danger of damaging expensive hardware during experiments. Tests can be carried out in short time without long preparation and with repeatable results. On the other hand, the fidelity of such simulation is dependent upon the precision of the employed computational models and its results need to be validated in field experiments

with real hardware before deployment in real-world conditions.

HIL and MR scenarios – HIL and MR scenarios are those with mixture of virtual and physical realizations of the entities subsystems. Depending on the virtualization level, the scenarios range from those with groups of agents on the same level of virtualization (HIL configuration), as depicted in Figure 4.3b, to scenarios where fully or partially simulated agents interact with others that are fully hardware deployed[1], as shown in Figure 4.3c. The most frequently virtualized layers are the *Asset layer* and *Environment layer*, that are replaced by a flight simulator, since utilization of hardware assets is the most costly and sensitive part of the development, and because outdoor flight tests are very time and resource consuming. The virtualization level of the remaining layers depends on the particular experiment's objectives. E.g., when the objective is to test a new communication layer, there is no need to use hardware sensors. On the other hand, when testing a new collision avoidance algorithm, the behavior of physical sensors can significantly influence the results of the experiments.

Fully deployed hardware field tests – In fully deployed hardware tests, all the agents are embodied by hardware assets with real sensors, with on-board computers and data modems. The tests are performed under controlled conditions and under operator and developers supervision (see Figure 4.3d). The reasoning and sensory data processing is executed by on-board computers that command the autopilots. Data modems are used to exchange information with other UAVs and the ground control station. This setting corresponds to the desired final stage of the autonomous UAV system where all the assets operate in a real world in distributed autonomous fashion with minimal requirements on human operator. However, carrying out the flight test in this configuration is the most demanding in costs, time, and risk taken. It is subject to all the issues mentioned in Chapter 2 and the system has to take them into account. This type of scenarios does not scale well because of the price of the overall setting, limitations of the communication medium bandwidth, legislation, and safety.

Fully deployed hardware application – These scenarios (schematically depicted in Figure 4.3e) are the final stage of the unmanned system development. The system is deployed in a real application such as industrial patrolling, communication relay, or surveillance and mapping missions. The system settings does not differ from the previous stage, however the conditions are not controlled anymore. System needs to be operational in all situations required by the application, such as various weather and light conditions. Energy management needs to be solved and safety and legal issues need to be taken care of. Exhaustive field testing must precede this final system stage.

[1]By fully hardware deployed entity it is meant an entity with none of its mission-critical parts virtualized that can be still equipped with virtual sensors for reception of simulation-based feeds.

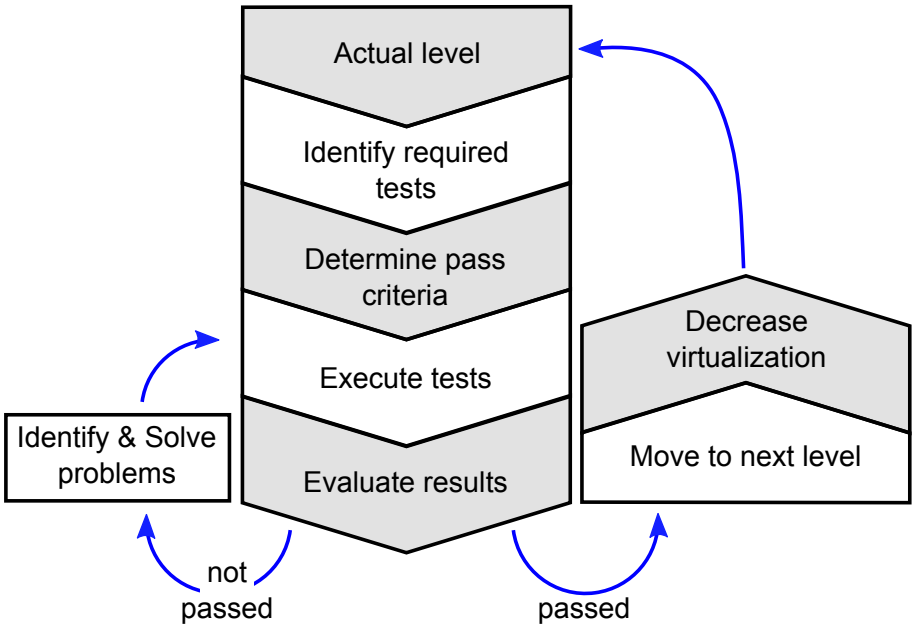


Figure 4.4: Steps for verification of individual virtualization levels.

4.1.3 Verification procedure

As the system moves through multiple virtualization levels during the development, it has to be verified on each such level before moving to the next level. The herein presented verification procedure ensures that eventual design or implementation errors are discovered and resolved as soon as possible. In such a case, the system is ready to proceed to further virtualization decrement. Respective verification steps for a given virtualization level are depicted in Figure 4.4.

At each level, it is necessary, at first, to determine the required behavior of the system and the requested fidelity features. Further, corresponding tests required for a transition to the next level need to be identified. Next, the criteria for successful passing of these tests, based on the predetermined requirements, should be set. The tests are then executed either in a software simulation or in field experiments. If the pass criteria are not met after the execution of the tests and evaluation of their results, it is necessary to identify and resolve the problems that caused the test failures and re-run the tests. Otherwise, if all the tests pass, it is possible to move to the next development level by decreasing the virtualization of the system.

4.1.4 Step Cost Analysis

To lower the system virtualization level during the incremental development process, a new testbed configuration has to be chosen at each development step. This selection should be based on the *iteration cost* for a given pair of the initial and target testbed configurations. As defined in [87], the iteration cost $cost(\sigma_1, \sigma_2)$ of a step that transforms the system from an application that works correctly on the testbed configuration σ_1 to an application that works correctly on the configuration σ_2 consists of three partial costs:

- the *testbed cost* of providing a testbed with configuration σ_2,
- the *development cost* of modifying the application logic of σ_1 to work correctly on the testbed σ_2,
- the *test cost* of verifying that the modified application works correctly on the testbed σ_2.

Authors of [87] do not specify how to estimate these costs. Based on years of experience of building UASs and many experiments with different testbed configurations, a more detailed specification of the costs of iteration step is provided here together with the analysis of the risks and costs during the development process. We presented the following analysis in [170].

The first two partial costs can be considered as self explanatory. The *testbed cost* $C_E(\sigma_1, \sigma_2)$ is a cost of the hardware equipment needed for moving from the testbed configuration σ_1 to σ_2. The *development cost* $C_D(\sigma_1, \sigma_2)$ is a price of the time and resources needed to modify the application logic to work in the testbed configuration σ_2. The estimation of the *test cost* $C_T(\sigma_2)$ reflect two following issues. First, it reflects the cost of the experiments $C_{ex}(\sigma_2)$ that consists of the preparation costs, cost for insurance and registration of the aircraft, cost of work-time of each person presented at the experiment, and cost of processing the results. Second, it reflects the costs caused by accidental failures $C_{fail}(\sigma_2)$ that consists of the cost of repairs and cost of repair time when no other experiments can be conducted with the broken equipment. Tools of the reliability theory can be used to estimate the probability of a failure or en error in design and implementation of a complex system. According to the reliability theory, the *failure rate* (i.e., the frequency with which an engineered system or component fails, expressed in failures per unit of time) in the initial stages of a system development can be modeled by Weibull distribution with a shape parameter $\beta \in (0, 1)$ [90], [207]. This distribution models a decreasing failure rate caused by design faults and initial problems and their fixing in the burn-in phase. The failure rate function according to Weibull distribution is given by

$$r(t) = (\beta/\alpha)(t/\alpha)^{\beta-1}, \tag{4.1}$$

where α is called the scale parameter and β is the shape parameter. The resulting iteration step cost can be expressed as

$$cost(\sigma_1, \sigma_2) = C_E(\sigma_1, \sigma_2) + C_D(\sigma_1, \sigma_2) + C_{ex}(\sigma_2) + r(t, \sigma_2)N(\sigma_1, \sigma_2)C_{fail}(\sigma_2), \tag{4.2}$$

where $r(t, \sigma_2)$ is the time-dependent failure rate of the testbed with the configuration σ_2 and $N(\sigma_1, \sigma_2)$ is the number of experiment required for verification of all new features of the configuration σ_2.

The development of the costs of failures $C_{fail}(\sigma_2)$ and the failure rate $r(t, \sigma_2)$ during the development process are depicted in Figure 4.5. When an error occurs and is fixed, the system

should be verified in all the previous development stages again and the next stage should be started only after thorough testing to ensure that the error rate is low enough. This way, in stage s_x, most of the problems that would otherwise occur also in the stage s_{x+1} are discovered and resolved. Thus, a lower probability of error can be expected in the stage s_{x+1} compared to the stage s_x. Notice the sharp step changes in the error probability at the beginning of each stage in Figure 4.5. These step changes are caused by an occurrence of new errors that could not appear in the previous stages because of their lower fidelity. Over the time, across all the stages, the overall error probability should be decreasing, since it follows Weibull distribution in a burn-in stage.

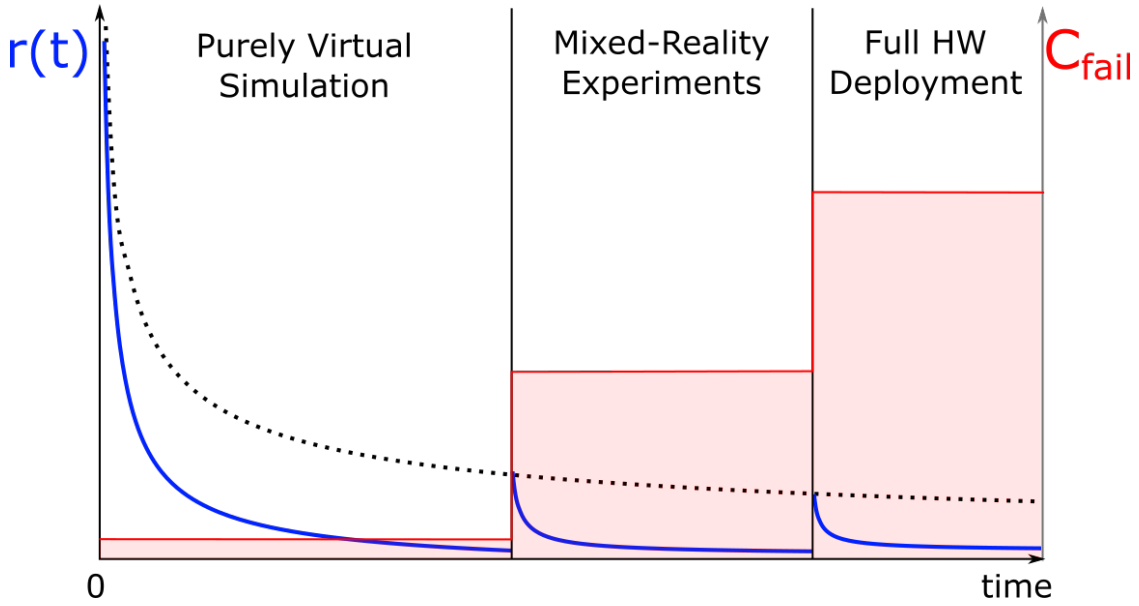


Figure 4.5: Development of the failure rate $r(t)$ and costs caused by incidental failures C_{fail} in three different development stages.

4.2 System Development Strategies

Depending on the purpose of the final system and nature of the simulation under development, some directions of the incremental development process by lowering virtualization levels may be better than others. They may result in faster development, lower risks, may require less resources or may be easier to implement. In the following text, recommendations for development directions, based on our numerous experiences, some of which are presented in Chapter 10, are presented.

- *Testing of single UAV planning algorithms*

  1. Start with simple models of asset and environment to verify the basic functionality of the algorithms.

2. Add more precise FDM to study the effects of flight dynamics on the plan execution precision.
3. Deploy the algorithms on physical hardware (on-board computer, autopilot) to study the effects of limited computational power.
4. Add simulated physical world feeds (wind, terrain, obstacles) to study effects of the Sensory layer on planning.
5. Conduct flight experiments in physical world with simulated sensory feeds such as wind, terrain, or obstacles.
6. Conduct flight experiments in physical world with real sensory feeds.

- *Deployment of new sensors*

1. Use sensor model to adjust planning and sensor data processing algorithms.
2. Deploy the real sensor in HIL simulation and conduct measurements in controlled environment to improve and validate the sensor model.
3. Deploy the real sensor on one real UAV and validate its correct function.
4. Deploy the real sensor on other assets.

- *Testing of communication subsystem*

1. Start with simple message exchange models to model concepts such as topology management, message routing, and message relay.
2. Enhance the simulation fidelity by using a wireless network simulator to model the message exchange.
3. Deploy physical communication devices in controlled environment with a small number of simulated entities to adjust the interfaces, study the effects of RF signal interference, and ensure correct media access control and message addressing.
4. Test the physical communication devices in the field experiments carried by people or ground robots for sake of safety. Study effects of distance, angle, and noise on the throughput and latency of the modems in use.
5. Test the physical communication devices in the field experiments deployed on UAVs to study the effects of real-world deployment such as altitude, speed, or attitude of the UAVs.

- *Testing of multi-UAV planning algorithms*

1. Start with simple models of assets and communication to develop and verify basic functionality of algorithms for team coordination and collision avoidance.

2. Incrementally decrease the virtualization of the system, similarly to the case of a single UAV, up to a level where real hardware is to be used.
3. Incrementally deploy the algorithms on the hardware assets. Start with hardware that does not need to be airborne to operate (modems, autopilots that can work in HIL mode, on-board computers, etc.) to avoid field experiments and keep the risk of damage minimal.
4. Deploy the algorithms on one real airborne asset with rest of the UAV agents being virtualized. Observe the interactions with the hardware asset.
5. Deploy the algorithms on a second airborne asset with rest of the UAV agents (if any) being virtualized. Now observe, in particular, the interaction between the two real assets.
6. Incrementally deploy the system on more airborne assets and observe the scalability of the simulation (dependent on the available communication bandwidth and safety of the system control).

4.3 Increasing Simulation Fidelity

The MiRAMuR architecture enables designers to arbitrarily change the simulation fidelity either by connecting a third party simulator or external computational model into the system, or using real feeds from physical sensors in MR. This way, a proper balance between accuracy of the simulation results, computational requirements, and complexity of the configuration and scenario setup can be achieved depending on the actual purpose of the simulation. In the following text, a utilization of third party simulators is discussed in the perspective of increasing the simulation fidelity.

4.3.1 Connecting External Simulators

In development phases when high fidelity levels are required, but deployment of real hardware is not possible yet, the MR simulations can be extended by external simulators for utilization of precise computational models they provide. Figure 4.6 shows possible layers, where the third party simulators or models can be utilized.

Individual external computational models and simulators can be imported to the MR simulation according to Figure 4.7. More details to the particular blocks are discussed further.

Weather simulation – Models of terrain and weather are kept and processed by the *Simulation server* as a part of its environment model. The most significant weather condition for the UAV simulation is the wind. The wind model in MR simulation can be used to provide more precise

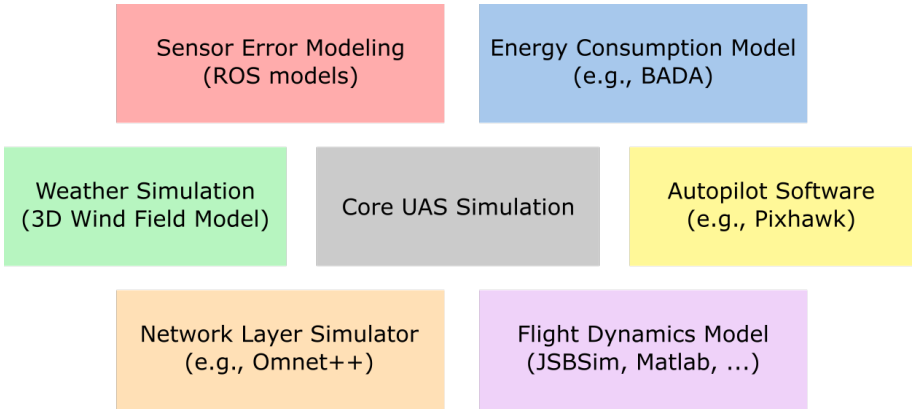


Figure 4.6: Possible third party simulators for increasing the simulation fidelity.

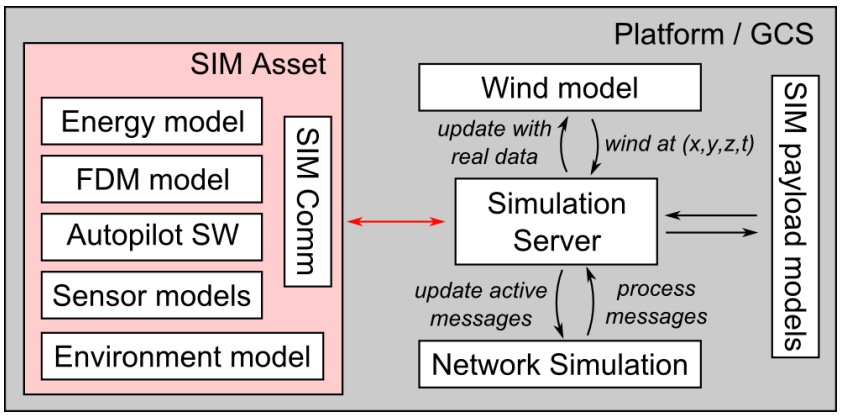


Figure 4.7: Interconnection of external models and simulators in the MR simulation framework.

simulations of flight. *Asset layers* of entities can use this model to update their FDM, and thus update the aircraft behavior in the wind field. *Sensory layers* of entities can sense the virtual wind vector in their vicinity if the entities are equipped with virtual wind sensors. In such case, the *Simulation server* sends the data from the wind field model to the virtual sensors.

For wind modeling, CFD solvers such as QUIC, or Autodesk CFD Solver, can be used [52],[71]. These solvers can model the 3D wind flow even in the vicinity of buildings. The wind field can be modeled with various levels of fidelity, from homogeneous case with single wind speed and direction, to a 4D array containing wind vector for every 3D position in specified time interval. This model is then sampled in space and time and provided as an input for the *Simulation server*. Precision of these models depends on initial conditions and can be refined by real-world measurements. This problem is further discussed in Chapter 9.

Flight dynamics simulators – Accurate simulation of a flight is a complicated task and requires precise models of aircraft as well as its environment. This modeling can be performed by existing simulators capable of providing FDM of aircraft. The FDM models can be used by the *Asset layers* of the simulated entities to precisely model their flight. The simulator runs on the

same machine as the virtual entity, i.e., either within the *Simulation server*'s machine, or, in a case of distributed simulation that would relieve the computation load of the server, on a separate computer. The inputs of the simulator are the models of the aircraft and its environment, as well as the actions taken by the autopilot. The output is the position, attitude, and momentum of the aircraft after application of such actions. There are variety of FDM simulators with different options of connectivity to external frameworks. Examples of such simulation tools are FlightGear [146] with JSBSim, ROS packages with Gazebo models, or Matlab with Aerospace Toolbox.

Autopilot simulations – Autopilot simulation provides the stabilization routines, basic localization and navigation capabilities and fail-safe procedures as reactions to the outputs from FDM, and instructions from the UAV operator. Autopilot Software is closely connected to the simulation of the entity flight provided by the FDM model. It can be connected to the architecture in the place of the *Autopilot layers* of the simulated entities. Some existing autopilot firmware can be directly employed in SIL simulation without the need of autopilot hardware. We have experimented with the APM and Pixhawk stacks, and with Kestrel autopilot simulation library, all of which were easy to deploy.

Models of sensors – Both the errors of the flight support sensors and those of UAS payload can be modeled to provide more accurate simulations. The errors of flight support sensors, such as IMU or GPS, cause localization errors of entities. Methods discussed in Section 2.1.2 can be used to model these errors. Noise and errors of sensory payload, such as cameras, radars, etc., can be modeled by the methods described in Section 2.1.3. The *Simulation server* has models of the payload sensors at its disposal. It decides whether a sensor can or cannot sense an object at each time step and in positive case, it sends precise sensory observations to the particular virtual sensor. These precise observations are then altered using the noise model in the *Sensory layer* and used for update of the entity's local model of the environment.

Energy consumption – The energy level and its consumption is modeled in the *Asset layer*. It can be part of the flight simulation of entities, however, as of our knowledge, there is no available simulator for modeling energy consumption of UAVs. Thus, the energy consumption needs to be modeled using results of flight tests and other methods, e.g., those described in Section 2.1.5.

Networking simulation – Communication among the entities and GCS is one of the critical parts of any autonomous multi-UAV system and its accurate modeling can significantly influence the behavior of the overall system. Communication and network simulation need to address problems such as modeling of the access to the shared medium, attenuation of the RF signal, communication delay, etc. The networking simulation is addressed on the level of the *Simulation server*, which receives messages from all the entities and process them on its own, as we described

in [168], or by the use of an external network simulator. This is in further detail discussed in Chapter 6.

4.3.2 Time synchronization

In a case of utilization of multiple high-fidelity simulators, the MR simulation platform may need to address the problem of time synchronization among individual simulator modules. E.g., when simulating the movement of an aircraft in real time and simultaneously modeling the effects of wind, the tasks of calculating the control signals and receiving the sensory information are compromised. In cases of system overload, when the simulation itself demands more time than the interval for the aircraft control, the UAV would not receive a new control signal for a period of a few cycles, and that could drive the system to an unstable operation.

In [182], the authors present a technique to synchronize distributed HIL system using an architecture based on the TMO (Time-triggered and Message-triggered Object) model. Another approach would be for the MR simulation platform to work as a quasi-real-time system, where the user should first verify, by testing, if the simulation computer has enough computation capability to support the simulation for several robots. This approach is taken, e.g., by Pizetta et al. [149] or by Selecký et al. [166].

Furthermore, in a case of high computational requirements caused by multiple aircraft in the simulation, it can be beneficial to distribute the visualization, aircraft control, and aircraft simulation onto several computers and share the data using network. For an example of centralized and distributed simulation, see Figure 5.1 and Figure 5.2 in Chapter 5.

Chapter 5

MR Framework for UAS Oriented Research and Development

According to the architecture and methodology of incremental development using MR simulations presented in Chapter 4, a MR framework was implemented that can serve as a validation test-bed for UAS oriented research and development. Moreover, the framework provides *Command and Control* (C2) capabilities for complex multi-UAV system.

At the beginning of this chapter, the requirements for such a Command and Control MR framework are stated. Next, centralized and distributed architectures of the proposed framework are presented in Section 5.2. Further, details of the designed approaches to particular multi-UAV system challenges, such as control of heterogeneous teams and planning with combination of different mission objectives are presented in Section 5.4 and Section 5.4.4.

5.1 Framework Requirements

We have defined several requirements for a framework to be successfully used for MR simulations of complex multi-UAV systems. These requirements can be divided into categories of MR-related capabilities and C2-related capabilities. The MR-related capabilities can be summarized as follows.

- **Modularity** – Framework layers should be easily replaceable for both virtual and real assets and their components. Interfaces between the layers must be identical for both simulated and hardware parts such that they are mutually interchangeable. Moreover, third party software, such as simulators, HMI software, flight dynamics models, etc. should be easy to integrate into the framework for reasons of increasing simulation fidelity and simplicity of the system usage.

- **Interaction of entities and environment** – Framework should distribute virtual feeds to the real entities and physical feeds to the rest of the system. Further, the *Simulation*

engine should be capable of detecting collisions of virtual objects with the terrain and other objects. Moreover, the *Simulation engine* has to provide routing of communication among real and virtual entities, according to Section 6.2

- **Integration of sensory feeds** – Simulation engine has to correctly integrate data from both real and virtual sensors in a consistent way (as described in Chapter 7).
- **Developments insights** – State information from both virtual and real entities has to be available for developers to exploit the advantages of MR for purposes of tuning and development.

The required C2-related capabilities of the MR framework can be summarized as follows.

- **Mission assignment** – Assignment of various missions to multiple UAVs should be provided.
- **Team management** – The UAVs should be able to form and maintain teams.
- **Asset allocation** – The UAVs should be able to efficiently allocate heterogeneous team members to assigned missions.
- **Planning for different tasks** – Different tasks mean different planning algorithms which have to be consistently integrated.

5.2 Framework Architecture

Design of the MR simulation framework is shown in Figure 5.1. The central element in the framework is the *Simulation server*. Its task is to update the overall model of the environment based on data from assets and sensors (as specified in Chapter 7) and distribute sensory data and communication messages between physical and virtual entities as described in Chapter 6. The physical (HW) and virtual (SIM) entities are further referred as HWEs and SWEs, respectively. Range of virtual sensors and virtual communication are based on their computational models. The *Simulation server* also interacts with the *HMI layer* for visualization of the simulation state and to allow mission assignment.

In cases when multiple assets need to be simulated on high-fidelity levels, problems with insufficient computational requirements may occur. A single computer may not be able to provide enough computational power. This can lead to time synchronization issues among HWEs and SWEs, or loss of stability in flight simulation of SWEs. This problem is discussed in Section 4.3.2. To relief the simulation platform in such cases, SWE simulation can be distributed on multiple machines as depicted in Figure 5.2. In such a scenario, the data exchange (sensory data, telemetry, communication etc.) among the *Simulation server* and SWEs is routed via TCP/IP network.

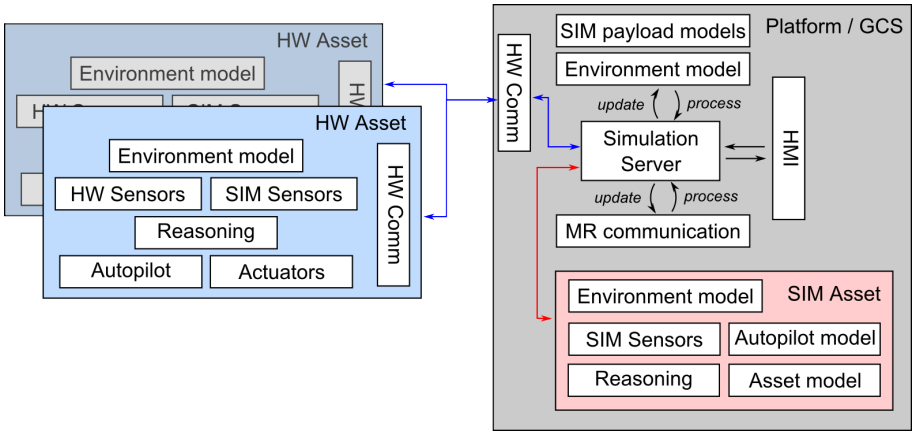


Figure 5.1: Design of centralized MR simulation framework. MR capabilities are provided by the *Simulation server* running on the same machine as the virtual assets.

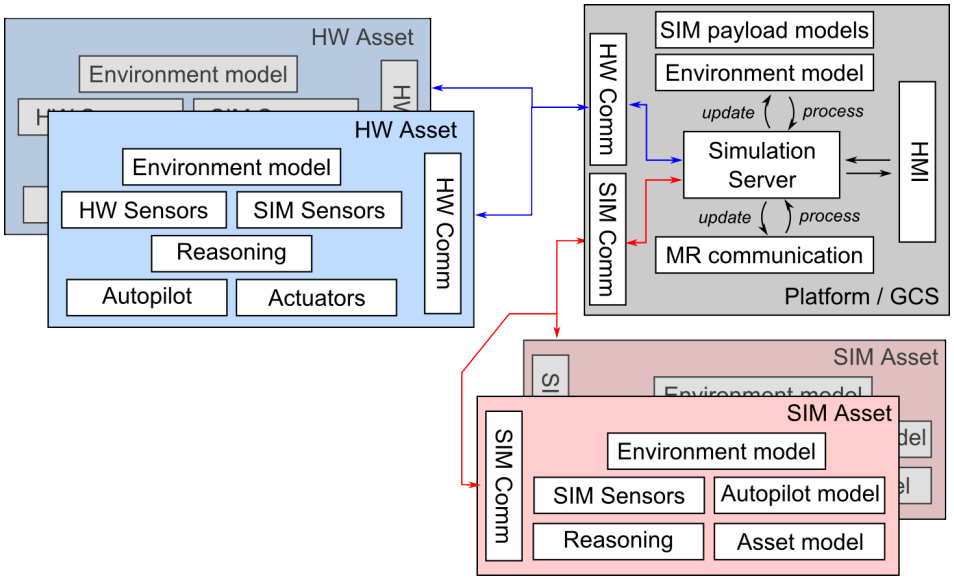


Figure 5.2: Design of the MR simulation framework with possibility for distributed simulation. SWEs are simulated on distinct machines and interact with the rest of the system using dedicated communication channel.

5.3 MR-related Capabilities

The MR capabilities of the framework are briefly covered in the following paragraphs.

Modular entity simulation – Design of SWEs needs to be similar to the HWEs as much as possible. All entities keep their local model of the environment based on observations from their sensors. Their state information (position, attitude, energy level, etc.) is passed to the *Simulation server*. Computational models of all system layers are implemented within the GCS to allow pure virtual simulations.

MR communication – To enable communication between HWEs and SWEs, the *Simulation server* has to function as a switch between two different networks as is further described in Chapter 6. Messages for SWEs are intercepted by the server, optionally delayed and their receivers are pruned according to the wireless medium model. This is in more detail described in Section 6.3.

MR simulation – The framework can be used for development with increasing level of fidelity. Figure 5.3 illustrates the development insights provided by the MR framework. An image from UAV's onboard camera is augmented with information about near virtual aircraft and a virtual village. Data from the simulation—the model of the environment and other UAVs—is fed to the sensors of the HW UAV allowing it to interact with the virtual UAVs.

Figure 5.3: Illustration of the development insights provided by the developed MR framework. Onboard camera image is augmented with information about near virtual aircraft and a virtual village.

5.4 C2-related Capabilities

The C2 of a team of autonomous UAVs is mainly addressed by the *Reasoning layer* of the assets. Agents build an environment model based on the information from their own *Sensory layer* or information received from other assets through the *Communication layer*. This model is then used an input for a set of planning algorithms that can produce complex behaviors. The *HMI layer* enables simple assignment of the operator-defined missions to individual UAVs or UAV teams. It process the arrived data (positions, flight plans, sensory data, etc.), visualizes them to the operator and stores them in a database for later analysis.

The process of control of heterogeneous teams is done in four steps: (i) submission of a mission consisting of a set of tasks; (ii) team formation and management; (iii) heterogeneous asset and task allocation; and (iv) planning and task execution. In the following, more details

are given to each of these steps.

5.4.1 Mission Assignment

Framework operators can submit a mission consisting of one or more tasks to one or more unmanned assets at any time. For this purpose, the framework provides a simple HMI layer. However this layer may be replaced by another, given it uses proper interfacing via TCP/IP for exchange of entities positions, plans, and control commands. Examples of two HMIs for operator control are depicted in Figure 5.4.

Several types of tasks can be assigned to the UAVs, e.g., waypoint following, area surveillance, target tracking, or an infrastructure patrolling. These tasks can be combined with cardinality 0-N into a complex mission. The operator can also specify *No-Flight Zones* (NFZs) of various shapes. Specifications of the NFZs are distributed to the UAVs and are considered during the planning process.

The process of mission assignment follows these steps.

1. All UAVs broadcast their ID and information about their currently available resources (remaining flight time, speed, available sensors, etc.).

2. Operator submits a mission that specifies one or more tasks and optionally a selection of the UAV assets.

3. Team *master* and its *followers* are determined. Assets utilize a master/follower coordination model, where the *master* is responsible for assigning all team members to particular tasks and the rest of the team follows (thus the *followers*) the given assignment. More details are provided in Section 5.4.2.

4. *Master* allocates available assets to individual tasks. More details are provided in Section 5.4.3.

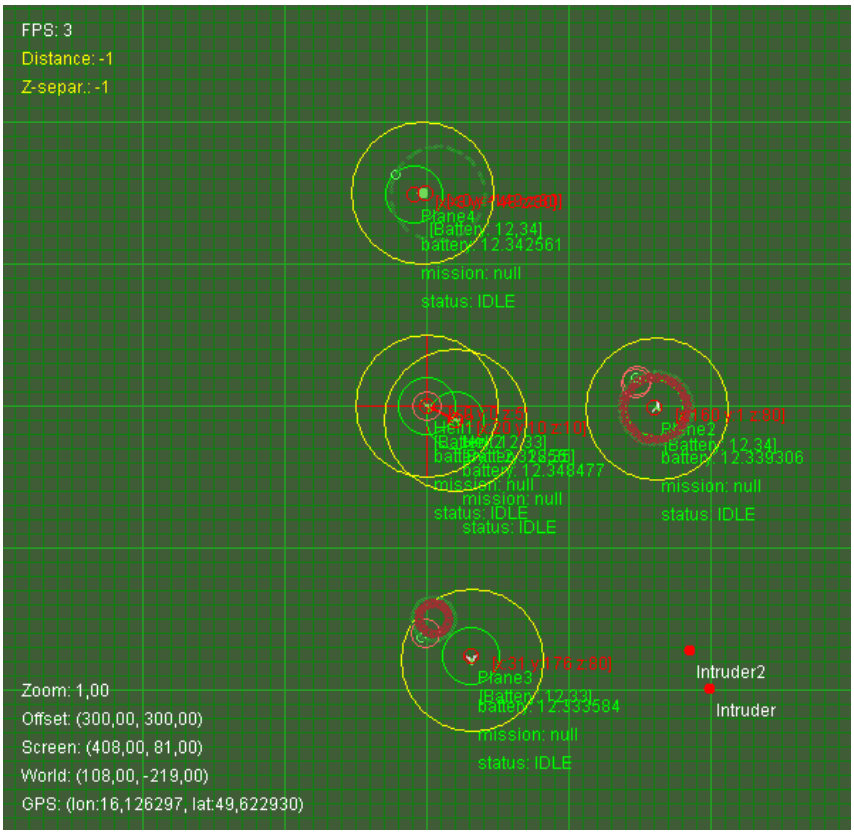
5. Optionally, the *master* asks operators for the approval of the mission tasks allocation.

6. *Master* sends commands containing the mission tasks to be executed by individual *followers*. More details on the planning for different goals are provided in Section 5.4.4.

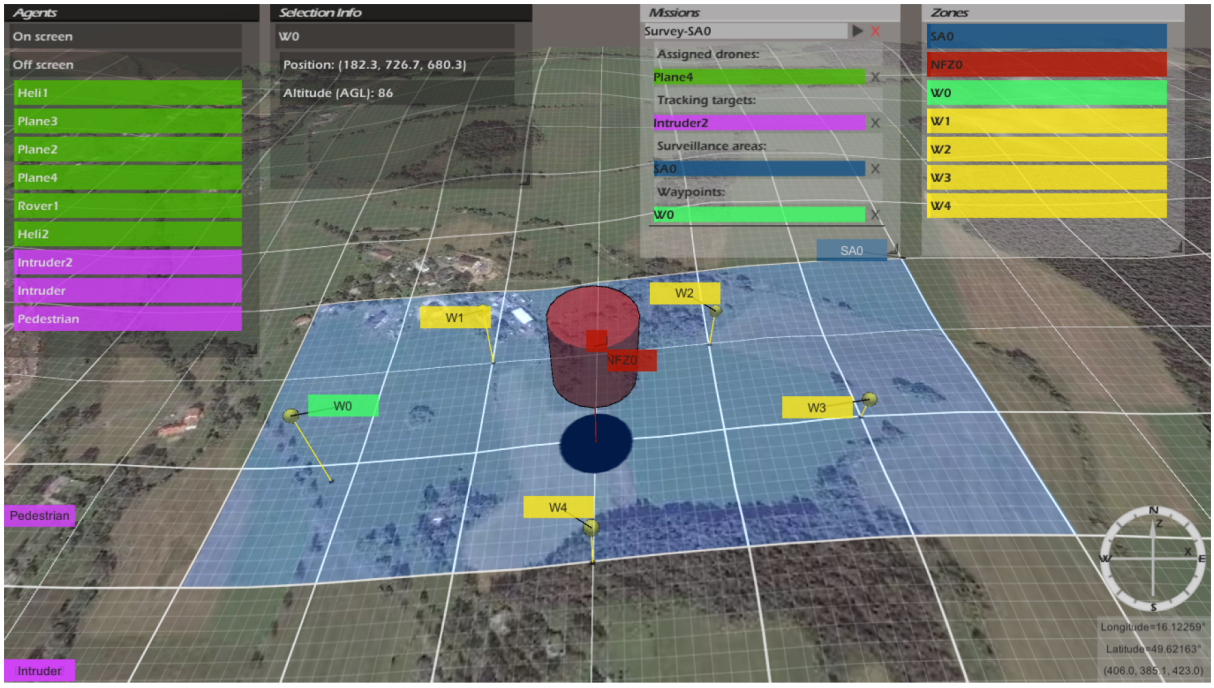
7. Execution of the mission is monitored by all assets and it is adjusted if conditions (number of available assets, remaining mission tasks, etc.) change.

5.4.2 Team Formation

Once a mission is submitted, UAVs form and maintain a team for efficient division of the tasks among them using the master/follower coordination model [138]. The team *master* is supposed to monitor the state of the team and to maintain team's allocation to the mission tasks efficient



(a)



(b)

Figure 5.4: Various HMI possibilities - (a) 2D operator window, and (b) 3D operator HMI, image adopted from [88].

in case of any state changes. The following paragraphs address three aspects of the team management process: (i) *master* selection, (ii) team management, and (iii) mission consistency preservation.

Master Selection – Every team member broadcasts a message to the rest of the participants, which contains information that can be used for the selection process (ID, battery level, communication capabilities, etc.). It is necessary for the *master selection* procedure to be the same for every team member so that each of them comes to the same result. The currently implemented selection procedure selects the asset with the lowest ID to be the team *master*.

Team Management – Once the team *master* is determined it starts to monitor the *followers* using *Master-Follower Protocol* as depicted in the sequence diagram in Figure 5.5a. Purpose of the protocol is to maintain connections between the *master* and its *followers* and to keep up-to-date information about the mission state. By default the protocol operates as shown in Figure 5.5a, part 1. *Master* sends a *query* that contains the current mission ID and the *followers* respond with an *inform* containing their status and currently available resources, such as their position and capabilities. In Figure 5.5a, part 2, the *follower* stops receiving *master*'s *queries*. After predefined timeout period the *follower* assumes that the *master* was lost and re-initiates the *master selection* procedure. This may result in several situations, depending on the reason of the communication loss.

- The *master* may be lost (e.g., because of malfunction), and therefore, all the *followers* restart the *master* selection procedure, select a new *master*, allocate resources and continue with the mission without the lost *master*.
- The *master* may be out of reach of the communication channel. All the *followers* which got out of reach of the *master* would form a new group, select a new *master* and try to continue with the mission.
- The *follower* lost a communication with all other UAVs in the group and thus it would become a *master* on its own and will continue with the mission.

To minimize the frequency of situations where the *master selection* is scheduled because the *master* is temporarily out of communication reach, the duration of the timeout period should reflect the size of the mission execution area and theoretical communication radius of the UAVs during the mission. Complementary situation, i.e., when one of the *followers* stops responding to *master*'s queries is depicted in Figure 5.5a, part 3. After the timeout period, the length of which should again reflect the communication specifics of the mission, the *follower* is assumed to be lost and the remaining assets are reallocated.

A team management process handles appearance of a new UAVs using *Master - New Member Protocol* depicted in Figure 5.5b. At the beginning of its operation, a UAV has no mission

assigned and is not a member of any team. It broadcasts a *query* waiting for *inform* responses of any *master* containing its current mission ID. Upon receiving *master*'s *inform* message, this new member responds with a join mission *proposal* that contains available resources such as sensing capabilities, battery level, flight speed, etc., *Master* checks whether it can allocate new resources and according to the result it sends back *accept* or *reject* as a response. New UAV then picks a mission to join from the accepted proposals according to the distance to the mission objective, number of assets in the mission, etc. It confirms the selected mission and rejects the others. The selected *master* then reallocates its team's resources including the new UAV member.

Mission Consistency Preservation – If the communication channel is noisy and unstable, the timeouts in *Master - Follower protocol* can result in establishment of several teams, each with its own master, executing the same mission. To prevent this inconsistent behavior, the *masters* follow *Master - Master protocol* as depicted in Figure 5.5c. If any *master* detects a *query* from another *master*, it checks the received mission ID. If both of the masters are executing the same mission, they re-run the *master selection* process. The asset that loses its master status sends a *query* for new mission allocation for itself and all its *followers* and upon receiving it, this asset redistributes the information to its *followers*.

If the multiple *masters* situation happens, it means that there are some communication issues and it can reoccur again. For this reason, both *masters* remain *masters* to their *followers*, but the mission is divided into two, each of them with new unique mission ID. The old mission ID is stored to be used for a comparison (together with the new one) with newly occurred *masters*.

5.4.3 Heterogeneous Resource Allocation

Once the *master* is selected, it processes the received information about available team resources and tries to allocate these resources to the mission tasks. A single UAV may be assigned only to a single task, some tasks require assignment of multiple UAVs to be executed effectively, and there is no need for delayed task assignment. This multi-robot task allocation takes a form of ST–MR–IA (Single-Task Robots, Multi-Robot Tasks, Instantaneous Assignment) problem, as formally defined by Gerkey and Matarić in [66]. In the multi-agent community, the ST–MR–IA problem is referred to as coalition formation [160]. The ST–MR–IA problem can be cast as an instance of (maximum utility) *Set Partitioning Problem* (SPP) [18], with E as the set of available UAVs, F as the set of all feasible coalition groupings G, and u as the utility estimate for each such grouping.

Definition (Set Partition) A family G is a partition of a set E if and only if the elements of G are mutually disjoint and their union is E:

$$g_i \cap g_j = \emptyset \quad \forall g_i, g_j \in G,\ g_i \neq g_j, \tag{5.1}$$

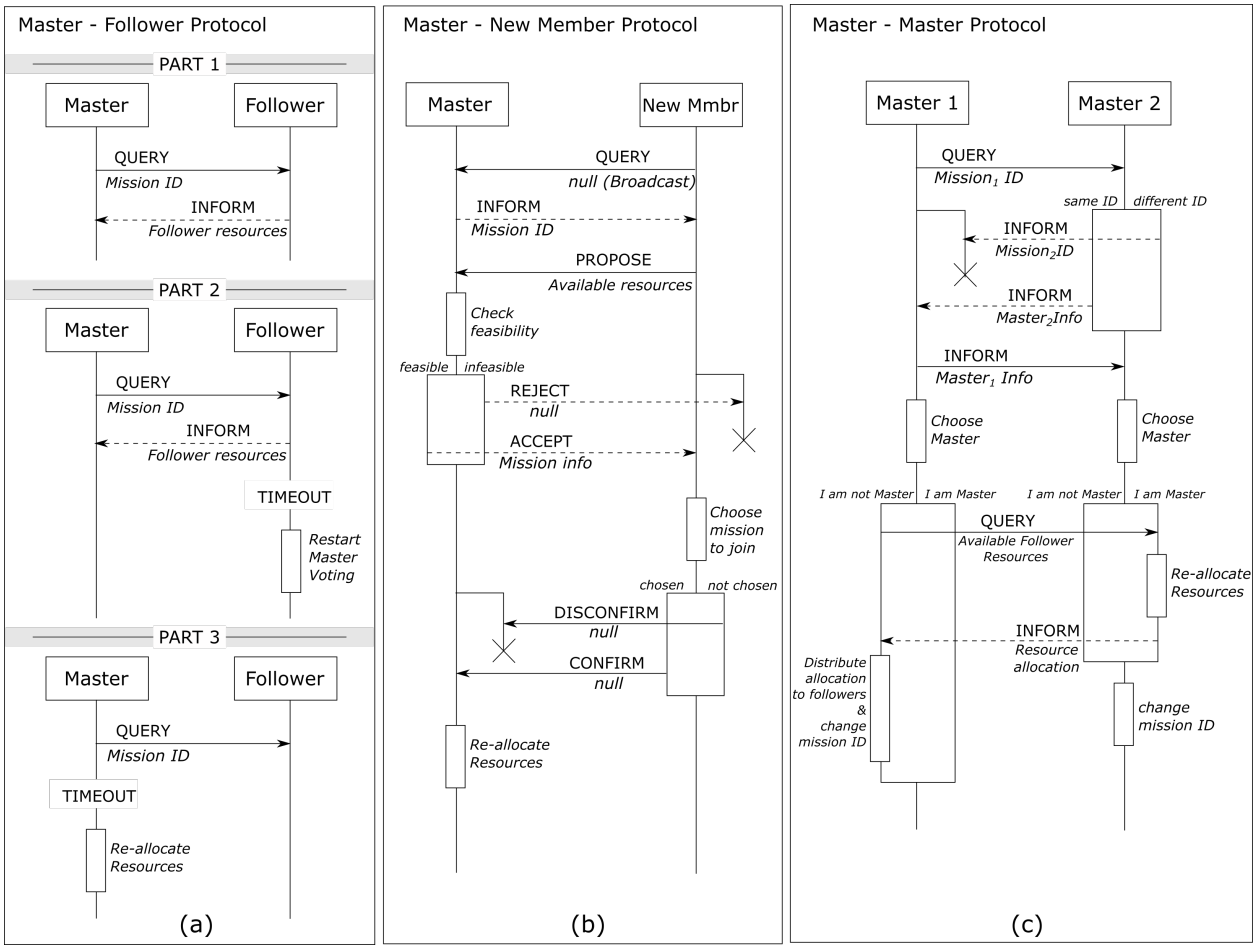


Figure 5.5: Sequence diagrams of (a) three situations during team maintenance, (b) a new UAV joining a mission, and (c) mission consistency preservation in case of multiple *masters*.

$$\bigcup_{g_i \in G} g_i = E. \tag{5.2}$$

Definition (Set Partitioning Problem) Given a finite set E, a family F of acceptable subsets of E, and a utility function $u : F \to \mathbb{R}_+$, find a maximum-utility family G^* of elements in F such that G^* is a partition of E.

The SPP is strongly NP-hard [64], however it has been intensively studied and many heuristic SPP algorithms have been developed [13], [81] that produce high-quality solutions in short time.

In the case of allocation of heterogeneous UAVs to multiple tasks, an optimal grouping of resources G^* is searched for where each mission task is covered by at least one sub-group of the UAV team and the utility of such grouping is maximal. Within the proposed MR framework, this G^* is found by solving the SPP using simulated annealing [97]. The utility value of a grouping G reflects how well are the available resources R assigned to the set of mission tasks T.

The available resources are defined as a set of available assets.

$$R = \{r_1, ..., r_N\}, \tag{5.3}$$

where N is the number of available assets. A single asset resource r_i is defined by a set of its features and capabilities.

$$r_i = \left\{p_i, v_i^{min}, v_i^{max}, S_i\right\}, \tag{5.4}$$

where p_i is the position of the asset r_i, v_i^{min} and v_i^{max} are its minimal and maximal speeds, respectively, and S_i is a set of its sensors.

The number of UAVs assigned to a single task is not limited. For example, an area can be equally well mapped by one UAV with large enough endurance, as by more UAVs. The difference is in the time necessary for the full area coverage. To reflect this problem, a notion of *saturation* is used. The *saturation* represents how well the task is covered by the assets. Its estimation process is task specific and generally holds that

- $sat < 1$ – there are not enough resources to satisfy the task
- $sat \geq 1$ – there are enough resources to satisfy the task

The utility of a grouping G is then estimated according the following equation.

$$u(G) = \sum_{t_i \in T} \left[sat(t_i, g_i) - \delta_d \sum_{r_j \in g_i} dist(r_j, t_i) \right], \tag{5.5}$$

where T is the set of mission tasks, g_i is a group of assets assigned to the task t_i in the considered grouping G. The *dist* function returns a distance of the asset r_j from the center of task t_i, and δ_d is a normalization constant which scales the distances to a value range of $(0, 1)$.

The saturation function is different for each task type. For illustration, we state the saturation functions for surveillance task and target tracking task in the following paragraphs.

Surveillance

Estimation of the saturation of surveillance tasks can be expressed as

$$sat(t_i = \text{`surveilance'}, g_i) = \begin{cases} x & x \geq 1 \\ 0 & x < 1 \end{cases}, \quad x = \sqrt{\frac{\sum_{r_j \in g_i} A_{max}(r_j)}{A_i}}, \tag{5.6}$$

where A_i is the area of the task t_i to be covered and $A_{max}(r_j)$ is a maximal area to be reliably covered by the asset r_j. $A_{max}(r_j) = 0$ if the asset r_j is not equipped with necessary sensors.

If the assigned resources are insufficient, the task's saturation is zero. It should be noted, that additional resources contribute to the task's saturation non-linearly to prevent over-saturation.

Tracking

The saturation of the tracking tasks can be expressed as

$$sat(t_i = \text{'tracking'}, g_i) = \max_{j \in g_i} comp(T_i, r_j), \tag{5.7}$$

where *comp* is a function representing competence of the asset r_j to track the target T_i. The competence is based on comparison of the minimal asset speed with the speed of the target, the minimum turn radius of r_j, and its sensor capabilities. $comp(T_i, r_j) = 0$ if the maximum speed of the asset r_j is lower than the speed of the target v_{T_i}. The asset need to be fast enough to be able to track the moving target and preferably capable of flying slow enough if the targets moves slowly. Also, note that we consider the maximum value of competence, not a sum, since one asset for tracking one target is enough.

5.4.4 Combining Different Planning Algorithms

UAVs need to be able to carry out various tasks at once, such as area surveillance, collision avoidance, or trajectory planning. Sometimes these tasks may be contradictory, thus a mechanism for combining different planning algorithms need to be employed. The planning architecture need to aggregate outputs of the planning modules, and provide solutions that respect various, often competing or conflicting, requirements and objectives of the modules.

One way to see the problem is as a general state-space search and consider various techniques for conducting such search: Distributed Constraint Optimization (DCOP) solving, planning techniques, game theoretic approaches, etc. These approaches generate overall optimal plans and allow utilization of state-of-the-art algorithms. On the other hand, they require a priori unknown and hard to deduce heuristics, and most of the time they would be intractable from both computational and communication point of view. Another approach is to create a fixed hierarchy of planning modules with various architectures (see Figure 5.6). This approach provide architecture modularity, simple implementation, instant convergence, and good information separation. On the other hand, the generated plans would be mostly sub-optimal. Based on analysis of all the considered options presented in [201], the sequential hierarchy of planners has been chosen and implemented as the most suitable option for the C2 of autonomous UAVs.

In the MR framework, individual planner modules are interconnected in the sequential hierarchy and connected to the sensors required for planning (e.g., GPS, radar, etc.). Each planner can modify or replace the current plan, or simply pass it without change to the next planner in the sequence. An example of a hierarchy of four simple planners is shown in Figure 5.7. All planners are implemented in such a way, that even when they have no valid plan ready (i.e., they

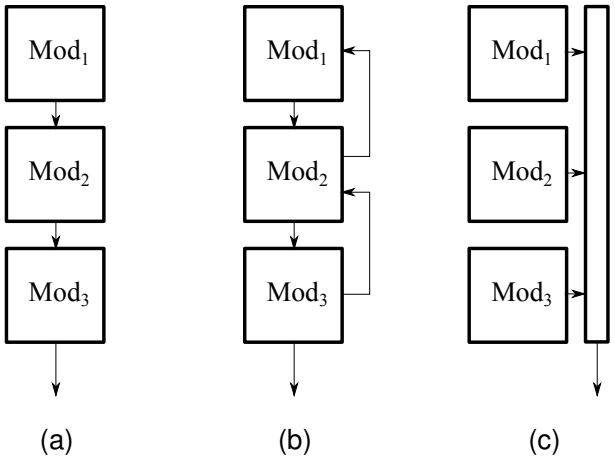


Figure 5.6: Considered hierarchical architectures of planning modules: (a) sequential, (b) backtracking, and (c) with a common bus, according to [201].

are still in the planning process), they simply hand over the plan received from the previous planner unchanged to the next one. This way, the UAV has always a plan to execute, even though it may be as simple as a loitering plan. Also note, that the first planner always produces a simple loitering trajectory at pre-configured flight levels different for each UAV. This is to provide time for more sophisticated planners to come up with their solution at the beginning of the planning procedure and in the meantime, provide a feasible collision-free solution for the UAV.

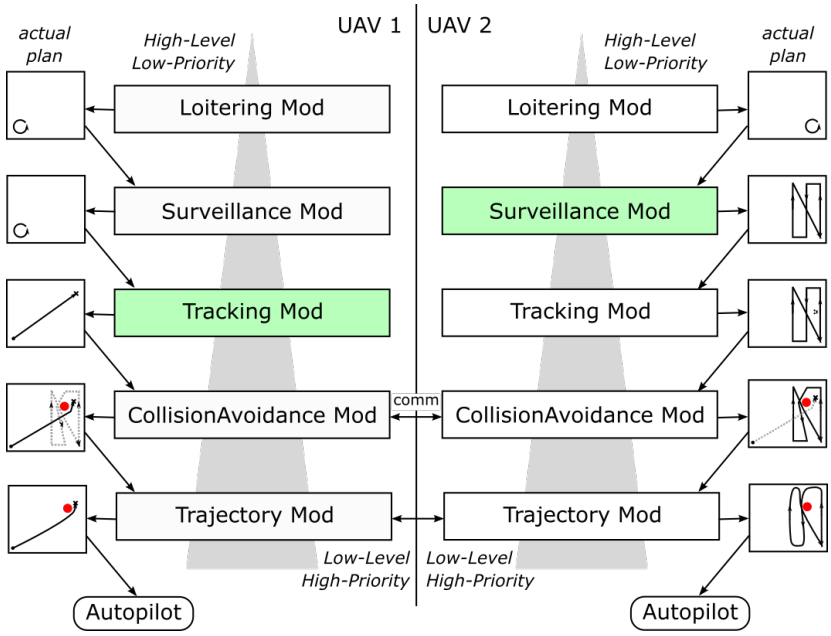


Figure 5.7: Sequential planning process of two UAVs. In the beginning, both UAVs loiter at their start positions. Then, UAV 1 is commanded to track a target and UAV 2 to map an area. Respective planning modules prepare their plans. When a mutual collision is detected through message exchange, collision avoidance modules fix the plans respectively. Finally, trajectory planners modify the plan to respect flight dynamics constraints and upload it to the autopilot.

5.5 Validation of the MR Framework

To validate the functionality of the MR framework, series of experiments with several fixed-wing and VTOL rotary UAVs were performed. Following the incremental development of multi-UAV systems, as specified in [169], these experiments started from pure simulation of all the assets and the environment. Next, HIL simulations with real modems and autopilots, and simulated sensors and flight dynamics were performed. Various autopilots, such as a Kestrel and MAVLink based APM and Pix-hawk autopilots were tested. Later, field experiments with MR simulations were performed with one hardware UAV and several virtualized ones. Finally, multiple fully deployed hardware assets were simultaneously deployed. More details about these experiments are presented in Section 10.1.

An example of a real mission with two heterogeneous UAVs—a fixed-wing and a VTOL rotary UAV—is depicted in Figure 5.8. When the operator assigns a surveillance mission, the UAVs perform master voting followed by resource allocation described in section 5.4.1. The final assignment respects the utility maximization criteria that reflects asset's speed, area coverage, and distance to the task. The zig-zag mapping pattern [61] is applied (see Figure 5.8b). Addition of a new tracking task causes re-allocation of the resources. The fixed-wing flies over the area, since it can cover large areas better and the VTOL UAV is chasing the target, since it can take advantage of its hovering capability (see Figure 5.8c). During the mission execution, the collision avoidance planners discover a possible collision risk and replan the trajectory accordingly (see Figure 5.8d).

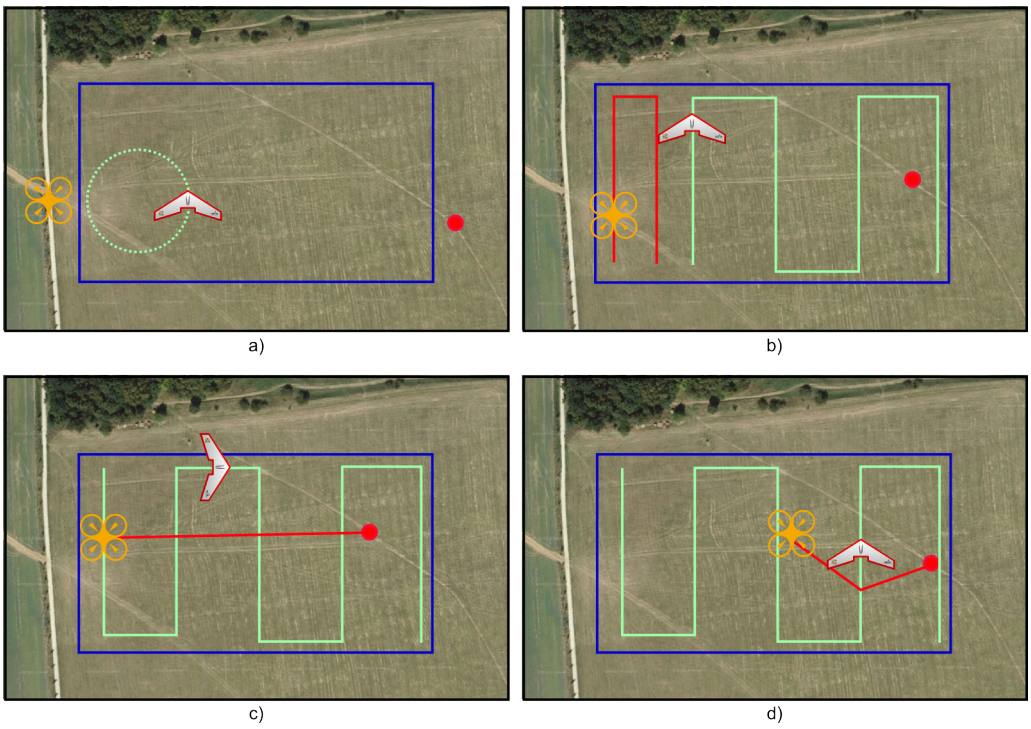


Figure 5.8: Example of a mission execution of the heterogeneous team comprising of one fixed-wing and one VTOL UAV: (a) while fixed-wing loitering and copter hovering on spot, a mission surveillance mission is assigned (blue rectangle); (b) tasks are automatically assigned to the UAVs to maximize the overall utility; (c) a new tracking task appears and the resources are re-assigned; (d) a possible collision risk is detected and avoided.

Chapter 6

Communication in MR Simulations

A proper design of communication subsystem (as any of the vehicle-to-vehicle or vehicle-to-base options) is one of the most important challenges of multi-vehicle system development. It is crucial for the control, cooperation, and collaboration of the vehicles, for purposes of routing and topology management [15], [135], message relay [113], communication constrained exploration [142], network-aware mission planning [20], or improvement of communication throughput and delay [57], [157].

MR simulation allows the interaction between both virtual (SW) and physical (HW) entities (further referred as the SWE and HWE, respectively), thus it is crucial to provide them with communication tools for complex group interactions. In particular, the lowest layers of the communication architecture, such as the physical interconnection of the entities, addressing and message routing need to be provided. Further, the MR simulation engine needs to address the aspects of signal propagation, interference and medium access control. Existing wireless network simulators enable modeling these aspects [7], [24], [195]; however, they mostly work with their representation of communication nodes and their simplified mobility models and it is not possible to utilize them from within other simulations.

In this chapter, UASs are used as a reference; however, the presented approach can be applied to other vehicular systems with direct communication among vehicles (vehicle-to-vehicle, V2V). After a discussion of the related work, mechanisms of message exchange in MR systems are described in Section 6.2. Communication architecture for the MR simulations is proposed, aspects of its fidelity are addressed, and a set of experiments is presented for its practical verification. In Section 6.3, a principle of a wireless network simulation that allows network co-simulation within the MR simulations is presented. This tool can provide the MR simulation with implicit mobility models—real trajectories of the vehicles—that help to increase the fidelity of MR simulations.

6.1 Similar Approaches to Communication Modeling

Wireless communication properties have been intensively studied, either on the level of signal propagation, throughput, and delays or on the level of network topology control and its effect on the asset connectivity [161]. Mahdi et al. [11] conducted detailed experimental analysis, which showed that wireless connectivity among small unmanned aerial vehicles (UAVs) is challenged by the mobility and heterogeneity of the nodes, lightweight antenna design, body blockage, constrained embedded resources, and limited battery power. Pinto et al. [147] provided a model for the quality of the UAV-to-UAV link regarding packet delivery ratio as a function of distance, packet size, and orientation, which was further supported by an extensive measurement campaign. Teng et al. [189] modeled the contributors to path loss and formed a combined propagation model that more accurately reflects the reality of tested scenarios. Various types of MAC protocols, whether they are contention-based, contention-free [75], or hybrid MAC protocols [89] have been suggested for vehicle ad-hoc networks (VANETs) and tested in simulations.

Network simulators such as ns-2 [24] and OMNET++ [195] support both wireless interactions and mobility for simulated nodes. Although these simple mobility protocols work well for simple mobile ad-hoc networks (MANETs), they fail to handle the mobility requirements of both flying ad-hoc networks (FANETs) [21] and VANETs. Use of a standard synthetic mobility model can result in incorrect conclusions, as the movement pattern can impact the networking performance of the system. FANETs require complex mobility models that involve autopilot behavior, effects of the wind, formation flights, and different mission scenarios [141]. Complex mobility models have been proposed based on an analysis of real movement traces [12]. For VANETS, there are traffic simulators such as SUMO [104] or VanetMobiSim [76] that are capable of simulating complex traffic scenarios and sophisticated driver behaviors. Specifically for the UAS reconnaissance, a pheromone-guided mobility model has been suggested in [107]. For different FANET application scenarios, various mobility models have been proposed in [28]. Despite these results, the existing approaches still fail to simulate complex operator interactions, effects of the environment, or multi-vehicle team coordination. Furthermore, the existing simulators can produce unrealistic scenarios and synthetic output [12].

Recent approaches to designing realistic simulation tools in the area of vehicular networks are presented in a survey by Silva et al. [174]. Real testbeds can solve some of the simulation's drawbacks and can be used to test applications in native environments [69], [149]. However, the costs of experiments with real testbeds are high, and they suffer from multiple issues such as scalability, limitations on specific scenarios, long preparation and running times, a lack of large-scale evaluation, and the difficulty of repetition, profiling, and modification of the experiment.

The problems of real testbeds can be addressed using an emulation concept where parts of the hardware components are emulated. The emulation data (e.g., the output of a simulator,

real traces, and an event-based generator's output) can be then processed by the network simulator. Representative approaches such as ns-2 and ns-3 [80] provide emulation extensions that enable the use of simulated nodes as if they were real network nodes. However, this emulation method can lead to unrealistic results due to the simulators' overhead caused by the required processing of all needed information [7]. This challenge is addressed by Ahmed et al. [1], who introduced a flexible VANET testbed architecture for VANET applications that tries to minimize the emulator overhead. However, since these approaches separate the mobility and network traffic simulation stages, they do not support a study of the interaction between these two stages, e.g., how the delays in communication influence the movement of the vehicles. Schünemann et al. [163] designed a testbed architecture that couples both traffic and network simulators and allows for the interaction between them in real time. In their testbed, vehicles broadcast information beacons (short messages with own position and other relevant data) using the network simulator. However, the architecture does not allow directed unicast messaging between the entities.

MR simulations, unlike the previously mentioned approaches, allow for simultaneous movement of entities and studying their networking capabilities on various levels of fidelity. For proper functionality, MR simulations require both broadcast and directed message exchange and they should allow for the movement of entities together with wireless network simulation. Although some of the mentioned work partially solve these problems, a communication architecture that would address all the required aspects, to the best of our knowledge, has not yet been described. However, since some MR simulation issues arise from a combination of two different networks (i.e., the networks of SWEs and of HWEs), some solutions can be found in the literature considering multi-network systems [185]. These issues are, e.g., problems with *network partitions* (where two or more UAVs are out of the communication range of the others and/or the simulation platform, usually the ground control station (GCS)) and problems with *sharing a single address* (when HWEs need to communicate with SWEs using a single address of the GCS). Furthermore, the entities in the architecture require information about the addresses of their neighbors. This problem is addressed by the IPv6 Neighbor Discovery protocol by sending advertisements, as described in [129], by broadcasting messages to create a proximity map of the vehicles' neighborhood [19], [128], or, if the available bandwidth is an issue, by using beacons on separate communication channels [6].

Concerning the existing work and the current lack of a proper communication architecture for the MR simulation, we propose a novel communication system architecture in this thesis. The proposed architecture allows for interaction and message exchange among HWEs and SWEs as well as an appropriate network simulation. The proposed solution enables high-fidelity testing in MR simulations that support a practical deployment of UASs.

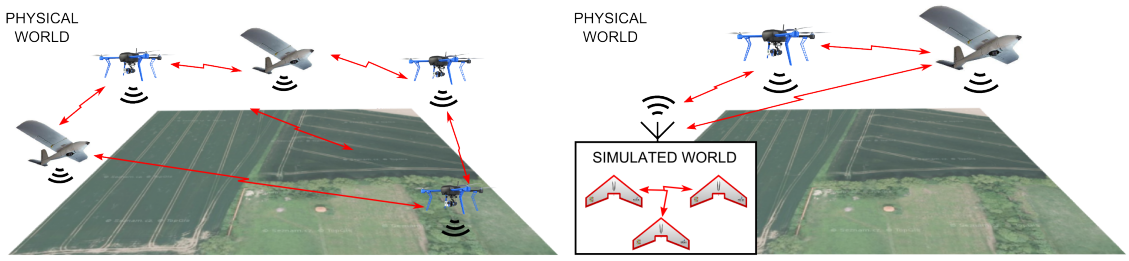


Figure 6.1: Principle of MR communication system—communication characteristics in MR simulation have to be similar to those of the fully hardware deployed target system.

6.2 Communication Architecture of MR System

One of the biggest challenges of integration of SWEs and HWEs in a MR simulation is to handle differences in the message transfer among the entities while preserving a high fidelity of the simulations (see Figure 6.1). The focus of this section is on designing communication architecture that would serve as a communication middleware for MR simulations and allow broadcast and directed message exchange between SWEs and HWEs while providing simultaneous wireless network simulation and movement of the entities in real time. The communication architecture must provide common addressing and message routing processes between SWEs and HWEs, and also suitable models of the medium utilization and communication reachability for the MR world. This architecture was first presented by Selecký et al. in [171].

6.2.1 Message Exchange in MR Simulation

In MR simulations, there are principally two networks—one among SWEs and one among HWEs. The simulation platform (located at the GCS) serves as a message relay between these two networks. Three types of connections are possible (SWE–SWE, HWE–HWE, HWE–SWE), and each of them employs different tools and methods for the message exchange and different processing of the communication visibility of neighbors, see Table 6.1.

Table 6.1: Types of communication connection in MR simulation systems.

Type	**Tools**	**Visibility**	**Method**
HWE–HWE	modems	physical	direct (modems)
SWE–SWE	sockets	SW model	direct (sockets)
HWE–SWE	combined	SW models (limited by physical vis)	message relay

The HWE–HWE connection uses real communication devices, such as radio-frequency (RF) modems, to exchange messages directly, and the visibility of neighbors is given by a physical RF signal propagation. In the respective SWE–SWE connection, messages are exchanged directly among virtual entities using TCP/UDP socket connection, and their mutual visibility needs to

be modeled. The HWE–SWE connection enables the MR interaction—both RF modems and TCP/UDP socket communication are required, together with a message relay node that serves as a bridge between two different networks. In this case, SWEs send and receive messages using the message relay node that is equipped with the RF modem of its own. The communication visibility in this type of connection is then partially determined using software models, and it is partially limited by the signal attenuation between the RF modems used for the actual message exchange.

6.2.2 Requirements for Communication Architecture

A communication subsystem in MR simulations needs to provide the following features to enable interaction between all the entities in the MR simulation and sustain the high fidelity of testing and verification.

- **Addressing** – Both SWEs and HWEs need to use unique addresses in a common address space to distinguish message receivers. In addition, messages need to have the option of being addressed selectively (not broadcast) so that (i) information is not given to entities that should not have it (e.g., when not covered by their sensors); (ii) the processing power of entities that are not receivers of the message is spared; and (iii), in some cases, wireless medium bandwidth is saved (e.g., when multiple wireless channels are used for communication or a directional information transfer methods are employed).

- **Transparent routing** – The system must provide the transparent routing of messages between SWEs and HWEs.

- **Identical message management** – Messages of HWEs and SWEs need to be managed similarly. Using different mechanisms for processing of HW and SW messages can bias tests regarding measurements of communication throughput, topology management, and team coordination.

- **Visibility model** – The system must be able to model communication visibility among all entities in the simulation (either based on their physical reachability or based on a signal propagation model) to provide high-fidelity simulations.

- **Network partitions** – In the case of two or more vehicles being out of communication range of the others and/or the simulation platform (i.e., GCS), the system should ensure that communication inside these newly created network partitions remain untouched.

- **Sharing single address** – When a communication bridge/relay between the networks of HWEs and SWEs is used, such entities need to be able to handle addressing using a single address of the bridge. It is necessary to determine the physical destination address of the messages that need to be relayed to other network types.

These features are important for the correct function of the MR simulation framework, high-fidelity validation, and testing, and it is mandatory to compose them within a single architecture of an MR communication system, which is proposed in the following section.

6.2.3 Proposed Architecture Design

Several topologies can be identified to fit the requirements of the communication system for the MR simulation that is defined in the previous section, and the most relevant topologies are depicted in Figure 6.2. One of the possible topology uses two different networks (the network of HWEs and network of SWEs) with one bridge to relay the messages between them (labeled as *MSG routing* in Figure 6.2 and 6.3). This bridge is represented by the simulation platform, most commonly the GCS. SWEs can be modeled either centrally within the GCS machine (as shown in Figure 6.2a), or individually in a separate machine that is connected to the GCS via TCP/UDP sockets to distribute the computational load on the simulation platform (as shown in Figure 6.2b). These topologies require only one communication modem to provide the simulation of all the SWEs; thus, they provide cheaper and less error-prone testbeds for high-fidelity system validation. On the other hand, they require an addressing mechanism for routing in both the networks mentioned in the previous section. Further, they have to address the accuracy-based aspects of the communication model discussed in Section 6.2.4.

The other possible communication system topology uses a separate modem for each entity in the simulation (both SWEs and HWEs); so, it creates one common network (as depicted in Figure 6.2c) with no need for any message routing and special addressing. Using RF modems for all entities prevents the problem of the bandwidth of single relay point discussed in Section 6.2.4. On the other hand, this topology requires an additional modem for every SWE in the system, and thus it decreases the scalability of the simulations and increases the cost. Moreover, this topology breaks the concept of the purely virtual entities since in this case, the SWEs have a hardware-deployed communication layer. For these reasons, only the topologies with two communication networks are considered hereafter.

Addressing

Each entity needs to have two parts of its address to send messages between two different networks. The first part specifies the HW network address, i.e., the address of the physical RF modem at which it can be reached, and the second (SW) part specifies the entity's address within the simulation and has to be unique for each entity. Information about these addresses needs to be propagated to all entities in both networks. A service similar to the IPv6 Neighbor Discovery protocol, as described in [129] can be used for that. Alternatively, if the available bandwidth is an issue and multiple channels are available, a method similar to the cluster-based beaconing process [6] can be utilized. The router advertisement service has been adopted in the proposed architecture. This service has been modified to help with maintaining the reachability

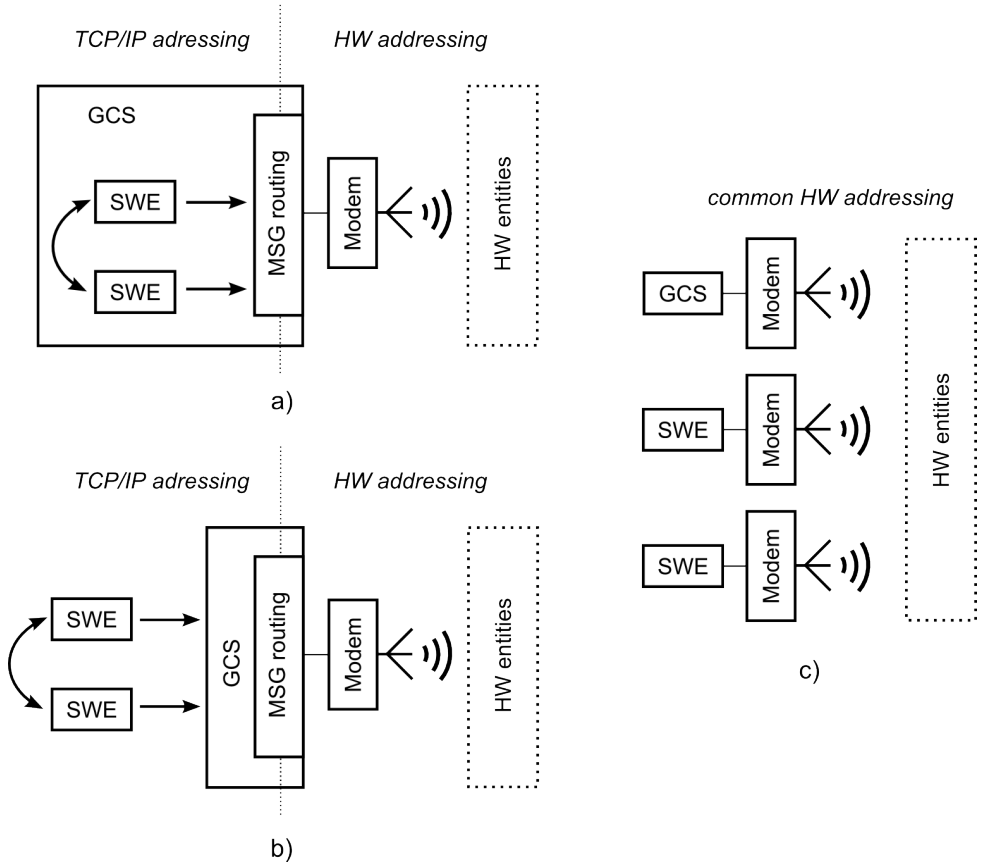


Figure 6.2: Communication system architectures—(a) virtual entities within a ground control station (GCS), (b) virtual entities apart from the GCS that use it for message relay, and (c) separated modems for all entities.

information of other active neighbors and the discovery of new entities in the environment. The entities periodically broadcast packets (*advertisements*) containing their ID and both their HW and SW address parts. After reception of such an advertisement, the entities update the list of their neighbors.

The message broadcast is the dissemination of a message to all reachable neighbors, both HWEs and SWEs. In the MR system, this means a usage of standard broadcast method at the sender's network (UDP broadcast in the SW network and RF transmission broadcast in the HW network) followed by a re-broadcast by the message relay node to the other networks upon message reception at the level of the GCS.

Figure 6.3 shows the principle of dual addressing on an example of two HW UAVs and two virtual ones modeled within the GCS. Here, all the communication participants have a unique SW address, and the SWEs share the HW address of the GCS.

Routing

Message routing used in the proposed architectures follows the flowchart depicted in Figure 6.4. If the target's HW address part corresponds to the HW address of a transmitting entity, a

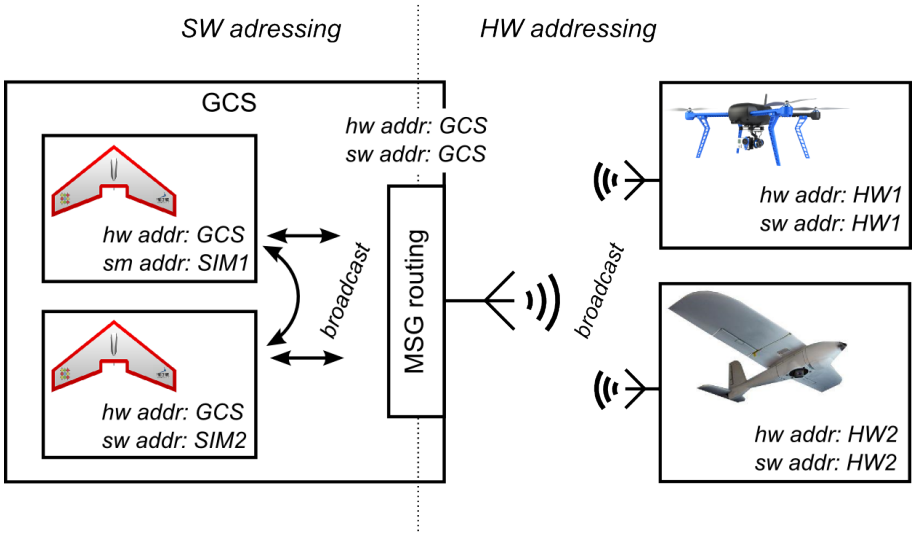


Figure 6.3: Dual addressing in MR communication system and distribution of *advertisment* messages.

mechanism of local message transport is utilized to deliver the message using the SW address part. We have used a combination of local IP addresses and TCP/UDP ports to represent the SW parts of the addresses and have used socket communication for local message transport. This applies to messages sent among SWEs. On the contrary, if the HW address parts of the target and the transmitting entity are different, a mechanism of remote message transport is employed to send the message to the appropriate HWE or GCS. When the HW part of the addresses is represented by the physical addresses of employed RF modems, the routing functionality provided by the modems can be utilized. If the utilization of specialized routing protocols is required, these protocols should be implemented outside of this architecture at higher control layers of the entities. Similar procedures are executed at the GCS during the message reception to provide the necessary message routing.

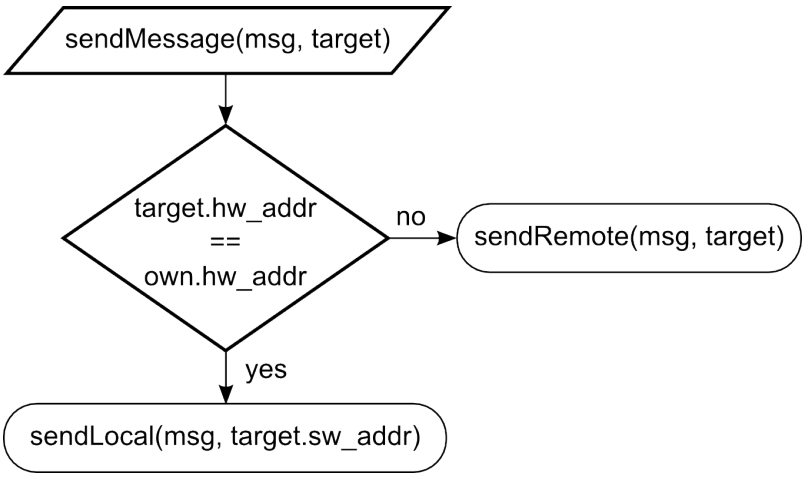


Figure 6.4: Routing principle in MR simulations.

When using such a system, SWEs need to know where to send packets for the HWEs, since they do not have direct access to the GCS's modem. The following two methods are proposed to provide this information. Both methods can be used for routing messages from SWEs to HWEs via the GCS. Each of the methods has its advantages and disadvantages that are detailed below.

- **Advertisement altering** – When re-broadcasting advertisements to the SWEs, the GCS changes both address parts of the advertising HWE to SW and HW addresses of this GCS while keeping a list that links the HWE IDs with their real addresses. Then, SWEs keep pairs of the HWEs' IDs linked to the address of the GCS; thus, it sends the message directly to the GCS whenever an SWE has to send a message to an HWE. The main advantage of this approach is that no fixed gateway address needs to be specified for the SW nodes. On the other hand, altering every advertisement message is inefficient in scenarios with many HWEs.

- **Fixed gateway** – In this method, SWEs have a predefined gateway for HW addresses that they cannot connect to directly. In this case, the gateway address is the HW address of the GCS. This approach needs the gateway HW address to be defined and known by all SWEs before the start of the simulation, which may be limiting in some scenarios. However, apart from that, no further actions are needed for the correct message routing. Due to its simplicity and efficiency, this particular method with the fixed gateway for all SWEs is the selected approach adopted in the proposed architecture.

The routing methods mentioned above allow for the correct operation of the system even in cases when the network is partitioned into several groups, which is one of the requirements of the communication system of the MR simulations specified in Section 6.2.2. When two or more HWEs are out of the communication range of the others and/or the simulation platform (GCS), they can still communicate with each other, independently of the platform.

Visibility model

The communication system in the MR simulation should provide the SWEs with a model of the wireless medium and simulate the neighbor communication visibility to provide a high fidelity simulation even on the message transport layer. Here, we propose to employ a wireless network simulation in the communication subsystem of the MR simulation, as shown in Figure 6.5. In the proposed architecture, the simulation engine can be configured to intercept messages from SWEs and delay them when the model of the wireless medium signals is currently occupied. The messages are sent only to a subset of the receivers if they are not reachable at the moment; otherwise, the message is discarded completely, see Section 6.3 for further details. The evaluation of the delay and reachability of the message receivers is based on the presence of environmental obstacles, the positions of entities, their real or virtual communication equipment, the used

MAC protocol, and a model of the actual utilization of the wireless medium and RF signal propagation.

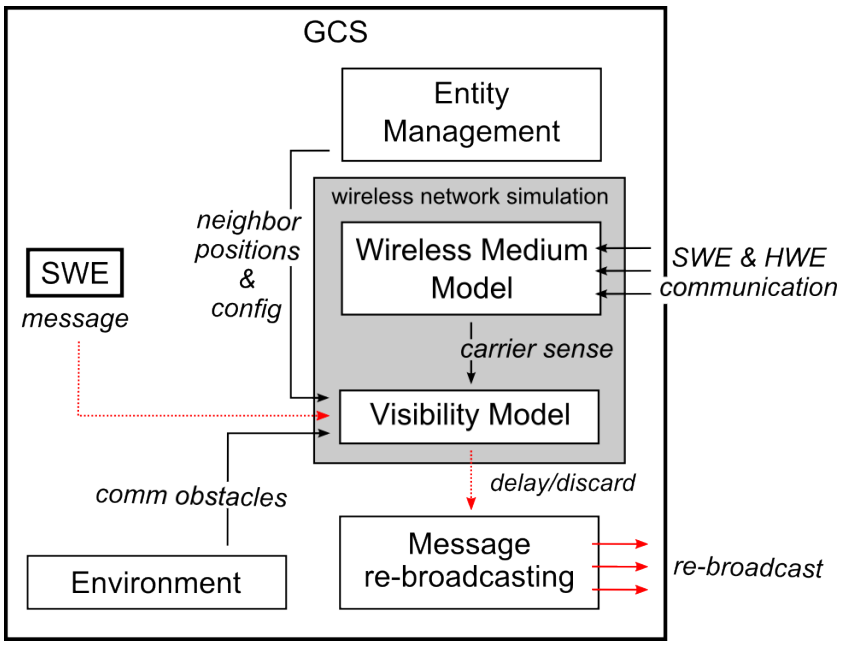


Figure 6.5: Visibility model for SWEs within the simulation platform (GCS) showing procesing of a message from one SWE.

Even though SWEs transmit their messages inside the SW network, thanks to the simulation platform, these messages are re-broadcast within the real wireless medium. This way, a controlled RF signal interference is "injected" into the wireless channel, so there is no need to model the communication medium for the HWEs. These modeling properties are experimentally verified in Section 6.4 and the principle of real-time network simulation inside the simulation server is in more detail described in Section 6.3.

6.2.4 Addressing the Precision Aspects of the Proposed Architecture

The proposed architecture enables validation process of complex multi-UAV systems through high-fidelity testing in MR simulations. It also provides functionality of precise modeling of wireless channel utilization by (i) employing networking simulation for SWEs and (ii) injecting an RF signal into the wireless channel, as described in Section 6.2.3. Even though the process of injecting the RF signal into the wireless channel is necessary for the high-fidelity modeling of the medium utilization, it also produces undesirable side effects caused by the fact that multiple SWEs use a single static modem for the communication with HWEs. It produces interference that is inconsistent with the reality and could lower the precision of the simulations. These simulation aspects need to be addressed, and we propose the following solutions.

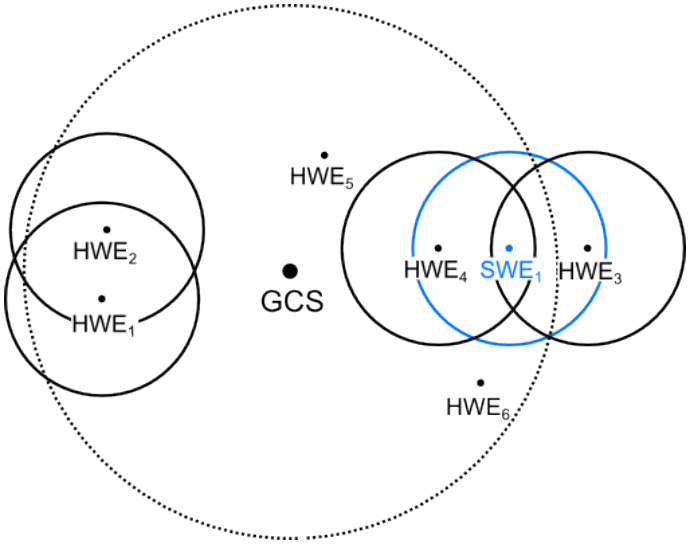


Figure 6.6: Illustration of configurations that could cause imprecise modeling of HWE–SWE reachability. Circles represent the communication ranges of individual entities and of the simulation platform (embodied by the GCS).

HWE-SWE communication reachability

Messages exchanged between the SWEs and HWEs need to be routed via the modem of the simulation platform. Thus, the RF signal characteristics of the message transmission between HWEs and SWEs are given by the signal characteristics between the HWE's modem and the modem of the simulation platform, and do not reflect the positions of the HWEs and SWEs in the simulation. This can result in situations where an HWE–SWE pair can be close to each other in the simulation, but the entities would be mutually unreachable for message transmission in reality since the HWE is too far away from the simulation platform. Similarly, an HWE–SWE pair could be far from each other in the simulation; however, if the HWE is close to the simulation platform, its messages can be delivered. The main issues in the communication reachability of HWEs and SWEs that can occur are illustrated in Figure 6.6 and are as follows.

1. Nodes SWE_1 and HWE_1 **should not be** mutually visible, but **they are**.
2. Nodes SWE_1 and HWE_3 **should be** mutually visible, but **they are not**.
3. Communication between nodes SWE_1 and HWE_4 **should not** interfere with the communication between nodes HWE_1 and HWE_2, **but it does**.

We propose the following precautions that should be applied to address the identified issues and to achieve the high-fidelity simulation of the communications.

1. A proper model of the RF signal propagation should be used for the message transmission between the HWE–SWE pairs to limit the message receivers to those that can be reached. This can be done during the message re-transmission issued by the simulation platform.

The particular mechanism of the wireless network simulation that can be used to address this aspect is described in Section 6.2.3.

2. Range testing of the simulation scenarios needs to be limited to the area that is reachable for the signal of the particular simulation platform's modem. The modem has to be equipped with a high gain antenna to provide as large a testing area as possible. This way, no HWE can get outside of the range of the simulation platform and thus out of the range of any SWE (unless it is caused by the signal propagation model mentioned in the previous case).

3. Undesired signal interference caused by the simulation platform affects all communications in the reachable area of the platform's modem. It cannot be avoided; only its effects are limited. Two particular precautions can be taken. First, the transmit power of the platform's modem can be adjusted for every transmission and set to the minimal required power for a successful message reception; therefore, it does not influence nodes further away from the platform. For example, if the power is set to transmit a message to HWE_4 in Figure 6.6, HWE_6 would not be affected; however, HWE_5 would still be. Second, the simulation platform's modem can be equipped with multiple directional antennas and a mechanism for using only selected ones for message transmissions based on the knowledge of positions of the target entities. However, the results of these two particular precautions have not been reported in the herein presented evaluation results. They are the subject of our future work.

Bandwidth of single relay point

Available transmission data rates of all entities are influenced by the use of a single message relay point, the simulation platform. The MR simulation is thus prone to three bandwidth modeling issues. The first issue is caused by the fact that SWEs do not use the RF modems directly. Thus, there are fewer modems than there are entities, so there is less RF noise than there would be in real scenarios. The second issue is caused by the communication among SWEs that is not affected by the limitations of the RF medium and modems. Hence, SWEs could theoretically send data at much higher data rates than entities in real scenarios. The third issue can arise from the fact that the platform's modem needs to transmit data for all SWEs, and this may require higher data rates than the modem is physically capable of achieving.

The following proposed precautions should be taken to address the simulation precision issues caused by the single relay point in MR simulations.

- Transmission delays caused by the medium access and back-off mechanisms need to be emulated. This is done by utilizing the medium access control (MAC) layer for SWEs as shown in Figure 6.5. Moreover, the arrival of HW messages at the simulation platform

needs to be monitored and integrated into the MAC layer model to limit the available bandwidth of the SWEs. Obviously, this behavior is dependent on the MAC protocol used, whether they are contention-based, contention-free [75], or hybrid MAC protocols [89] suggested for VANETs. Section 6.2.3 describes the mechanism of the wireless network simulation that can be used to address this simulation aspect.

- The broadcasting nature of the wireless channel traffic needs to be emulated, and the bandwidth of HWEs needs to be limited. In the proposed architecture, this is achieved by injecting controlled RF signal interference. Specifically, all message transmissions between SWEs need to be re-broadcasted over the simulation platform's modems as stated in Section 6.2.3.

- Transmission data rates of all SWEs must be limited not to collectively request to transmit more data per second than the modem used by the simulation platform is physically capable of transmitting. This is achieved by using a high data-rate modem for the simulation platform, limiting the number of simultaneous large data transmissions, or, if the previous cases are not applicable, by limiting the number of SWEs in the simulation. Another option to prevent large simultaneous data-rate requests is to ensure the distribution of SWE transmissions over time to comply with modem limitations. However, this last approach has not yet been implemented and is a subject of the future work.

A comparison of the reached data rates in the case of hardware deployed entities and in the case of entities in an MR simulation that employs the proposed architecture and precautions is the main subject of the experiments presented in the following section.

6.3 Network Simulation in MR System

Interaction of the entities in a multi-UAV system is sensitive to the reliability of the communication link, its throughput, message delays, and limited communication ranges. The effects of these phenomena on the behavior of individual entities and the overall system functionality can be studied either using real hardware or a network simulator. Utilization of network simulation is preferred in the early stages of the system development since it comes with lower costs and better repeatability of the experiments.

The principle of the deployment of network modeling features into the MR simulation is depicted in Figure 6.7. This figure shows the necessary steps to model the message transmission and reception in situation when entity X sends a message to entity Y.

1. Message is intercepted on the level of entity X's *Comm layer*.

2. Digest of the message (sender, receiver(s), size, transmission power) is send to the *Simulation server*.

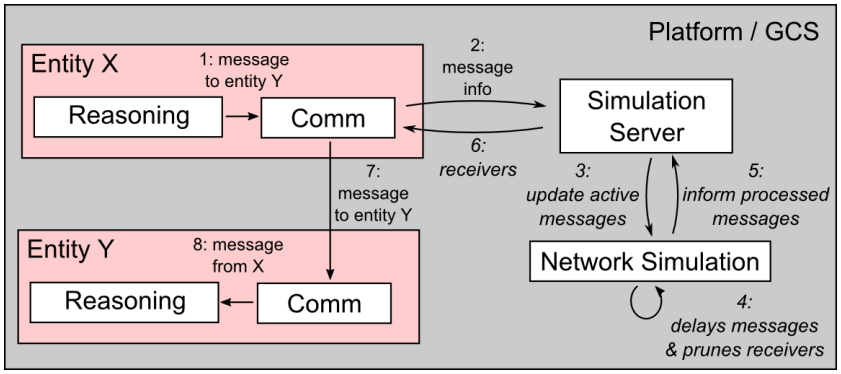


Figure 6.7: Principle of deployment of network simulation into MR simulations.

3. *Simulation server* updates its network simulation model with this message digest.

4. The network simulation model decides how much should the message be delayed and which receivers are reachable (see details in Section 6.3.1).

5. When the message is ready to be sent to their receivers, network simulation model informs the *Simulation server*.

6. *Comm layer* of entity X is informed about message ready to be sent.

7. Message is passed to the *Comm layer* of the entity Y.

8. Message is received and its content can be processed.

6.3.1 Network Simulation within Simulation Server

It would be difficult to fully utilize an existing network simulator in the MR simulations, since most of these simulators do not allow individual entities to move arbitrarily and only utilize particular mobility models which the entities must follow. Moreover, not all network simulators allow operation in the real time. Thus, it is more convenient to implement network simulation features directly within the *Simulation server*. A principle of such a message transport simulation implemented within the *Simulation server* is depicted in Figure 6.8. Characteristics of the simulation are described in the following paragraphs.

Discrete time – The simulation process is periodically triggered every time step T when all the messages intercepted in the previous time interval are processed. The simulation engine checks the reachability of the message receivers, delays the message transmission according to the size of the message and the occupancy of the frequency channel, handle interference of signals, and if necessary, simulate the message re-transmission process. In the scheme in Figure 6.8, the simulator processes messages from every node and puts them in queues according to the time of their transmission. These queues simulate the transmission back-off due to carrier sensing.

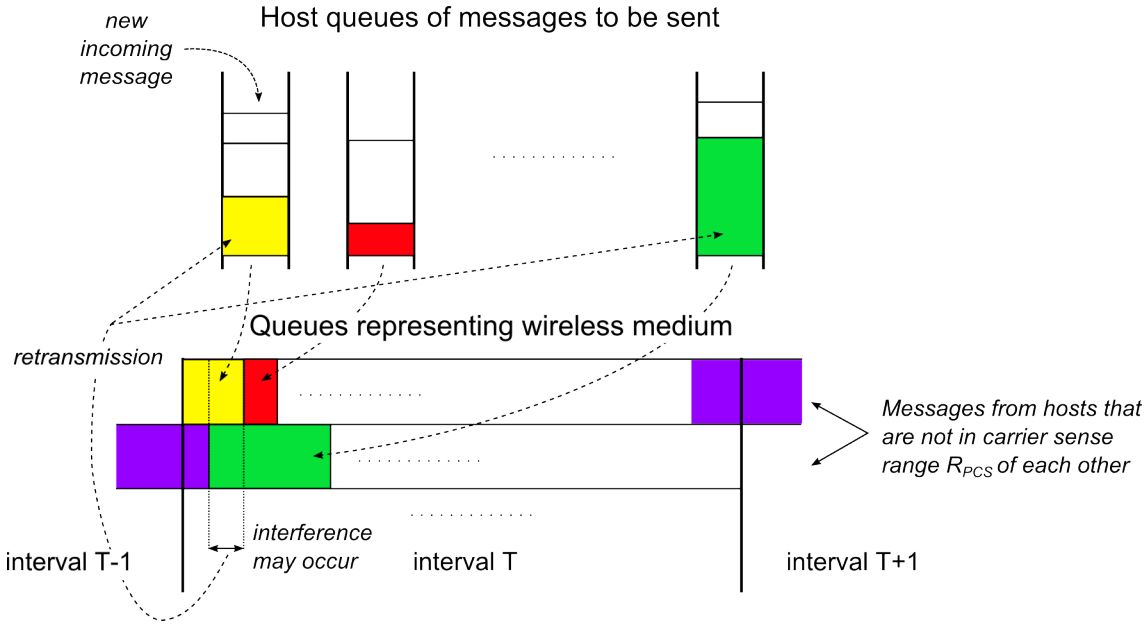


Figure 6.8: Basic scheme of message transport simulation within the *Simulation server*. Messages are processed in discrete intervals according to their transmission times, delayed according to other sensed transmissions, and checked for interference.

Thus there are separate queues for messages from hosts that are not in the physical carrier sense range (R_{PCS})[1] of each other. Messages that overlap in time may be subject to the hidden node problem [133] and need to be checked for interference at the receivers. If the messages interfere significantly, they need to be retransmitted according to MAC layer protocols defined in IEEE 802.11 standard [86]. Messages starting in the interval T_i but overlaps the interval, are processed in the interval T_{i+1}, with their duration being lowered by the time in the interval T_i (see the overlapping in Figure 6.8).

Message delay – Processing time of the messages can be estimated according to

$$t_M = \frac{L_M}{B_w C_o}, \tag{6.1}$$

where t_M is the transmission duration in seconds, L_M is the size of the message in bits, B_w is the channel's bandwidth in bits per second, and C_o is a coefficient that represents the overhead of the protocols of link and MAC layers. Discrete behavior of the simulation process adds further delay to the messages—a message cannot be processed sooner than at the end of the simulation interval. This adds a delay of $[0, T - t_M]$ milliseconds.

Node reachability – There are three cases when two nodes are not mutually reachable. Either (i) they are too far away from each other, so that the signal strength at a receiver is lower

[1] distance from an entity within which the other stations can detect the entity's transmission and postpone transmissions of their own

than receiver's sensitivity threshold; (ii) the noise strength at receiver is larger than the signal strength; or (iii) there is an obstacle between the nodes that is opaque for the RF signal.

Message processing – Every incoming message should be processed by the network simulator in the following steps:

1. Time required for the message transmission is estimated according to (6.1).
2. Receivers of the message are pruned according to their reachability, that is specified by the RF signal propagation models, obstacles, and signal strength at the receivers.
3. MAC layer protocols are applied to handle carrier sensing, back-off times, and optionally the CTS/RTS mechanism.
4. Transmissions are tested for interference.
5. Unicast messages, receivers of which were removed in the previous steps are re-transmitted.
6. Senders are informed about receivers where the message can be sent to.

6.4 Practical Verification

To verify the proposed architecture and its network simulation features, a set of experiments was designed. The configuration of the experiments' testbeds and their results are discussed in the following sections.

6.4.1 Verification of MR Communication

The following experiments show how the communication characteristics of a testbed configuration with solely HWEs present can be substituted with the characteristics of MR testbeds, where some of the HWEs are modeled using SWEs. The experiments measuring the data rates were chosen since they well illustrate the behavior of the system in various settings. This behavior is expected to be similar when other communication characteristics are measured.

Two testbed configurations are considered in the presented results for the proposed experimental setting: one with three HWEs, which is hereafter referred to as the HW scenario, and one with the simulation platform (represented by the GCS) to communicate with one HWE, and two remaining entities modeled in the MR simulation. The latter configuration is hereafter referred to as the MR scenario. These two configurations are schematically depicted in Figure 6.9. The MR simulation used the architecture with SWEs modeled within the simulation platform according to Figure 6.2a.

The main evaluation indicator of the communication characteristics is measured as the number of bytes per second successfully transferred between the three entities. All entities have

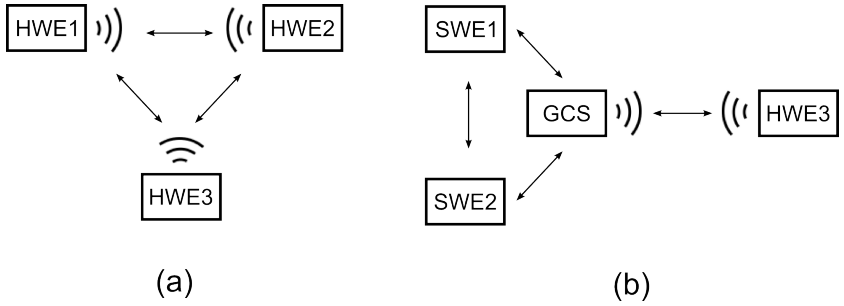


Figure 6.9: Two experimental scenarios—(a) HW scenario with three hardware entities and (b) MR scenario where one hardware and two simulated entities communicate via GCS relay.

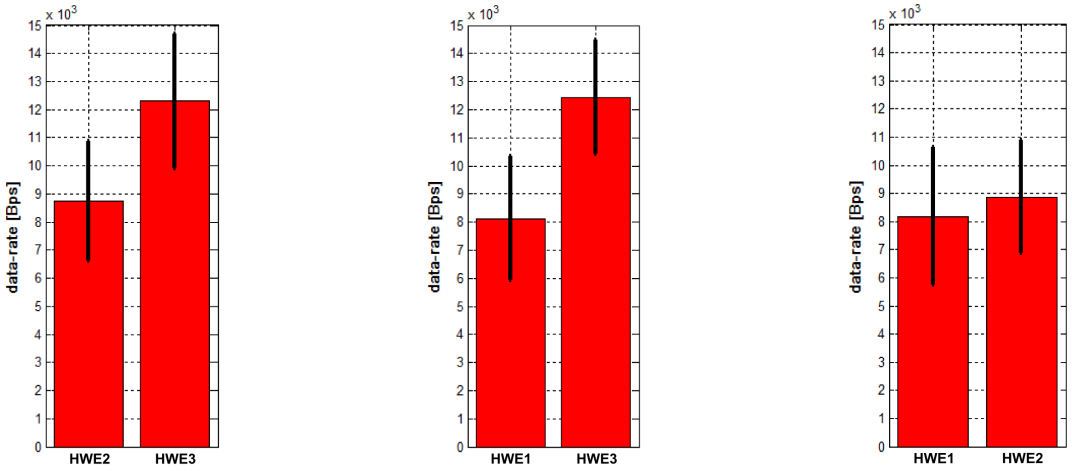
been static and positioned at the same distances from each other. During the experiments, all of the entities broadcasted packets of 1 kB at a rate of 15 kBps using the UDP protocol, and successfully received data was measured every second for each entity.

Three Microhard nVIP2400 modems with the TX power set to 14 dBm were used as the communication hardware. One of the modems (in particular the one utilized by HWE_3) was equipped with an antenna that was superior to the others (8 dBi vs. 2 dBi). Thus, in a case of simultaneous communication of all entities, it was capable of transmitting approx. 12 kBps through the wireless medium, while the other two modems (used by HWE_1, HWE_2, and by the GCS in the second testing configuration) were only capable of successfully transmitting approx. 9 kBps in the HW scenario.

The mean values and standard deviations of the measured data rates in the HW scenario are shown in Figure 6.10. The results indicate the above-proclaimed transmission data rates of the HWEs.

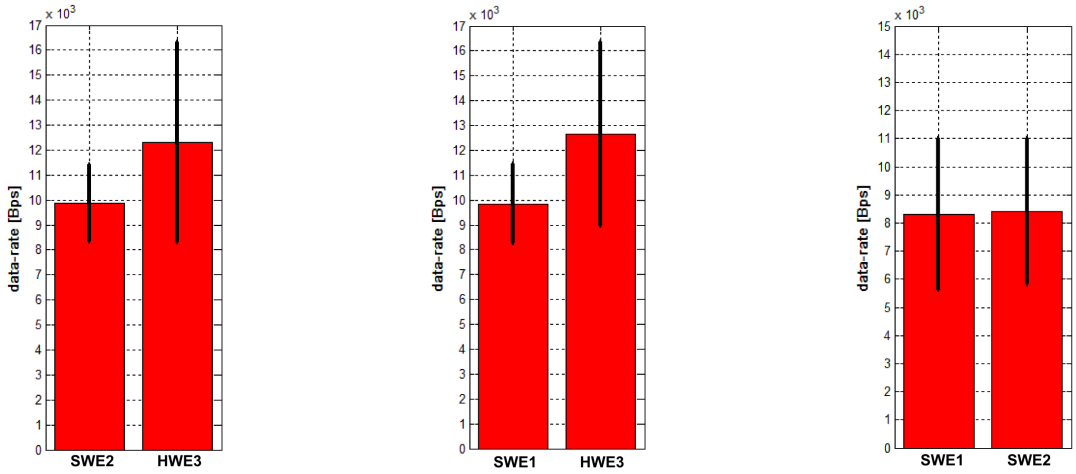
In the MR scenario, the GCS uses the modem of the HWE_1 from the HW scenario. The measured average data rates and standard deviations are shown in Figure 6.11. The characteristics are almost similar to those from the first experiment. The data rates measured between the two SWEs are higher with a lower variance, which may be caused by a low precision of the employed wireless network model and by the fact that the environmental background noise has not been taken into account during the simulation. On the other hand, the variances of the real traffic (data from HWE_3 as shown in Figure 6.11a,b, and data from SWE_1 and SWE_2 shown in Figure 6.11c) are higher. This is most likely caused by the medium access control in situation when the GCS's modem tries to send twice as much data compared to the modems in the HW scenario. The similarity in the characteristics of the SWE transmissions arises from the fact that the same modem is used for the message exchange; thus, the system operates in similar conditions during the transmission. In this configuration, the GCS's modem needs to transmit twice as much data compared to the first experiment since it is now transmitting data for both SWEs. However, thanks to the lower bandwidth occupancy caused by one missing modem, and because the model is physically capable of transmitting this amount of data, the packets

are transmitted with almost no change in bandwidth in comparison with the first experiment. The results of this experiments demonstrate that both configurations can be used without a significant difference nor with any effect on simulation accuracy.



(a) (b) (c)

Figure 6.10: Mean values and standard deviation of data-rates of transmissions from neighbors measured at entities (a) HWE_1; (b) HWE_2; and (c) HWE_3 in HW scenario.

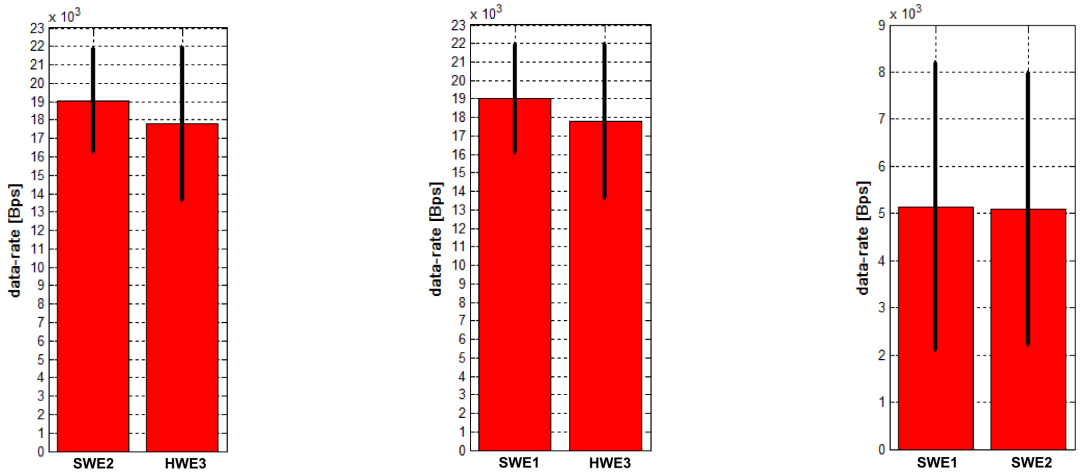


(a) (b) (c)

Figure 6.11: Mean values and standard deviation of data-rates of transmissions from neighbors measured at entities (a) SWE_1; (b) SWE_2; and (c) HWE_3 in MR scenario using the high fidelity MR architecture.

Further, we also performed MR experiments with the testbed configuration similar to the MR scenario, but without any wireless network modeling. In this case, the UDP packets were sent at a rate of 20 kBps to see how the system would behave in situations where the entities are trying to send more data than the real RF modems can transmit without any regulatory mechanism. The achieved results are depicted in Figure 6.12.

According to Figure 6.12c, the results indicate what happens when the modem tries to send



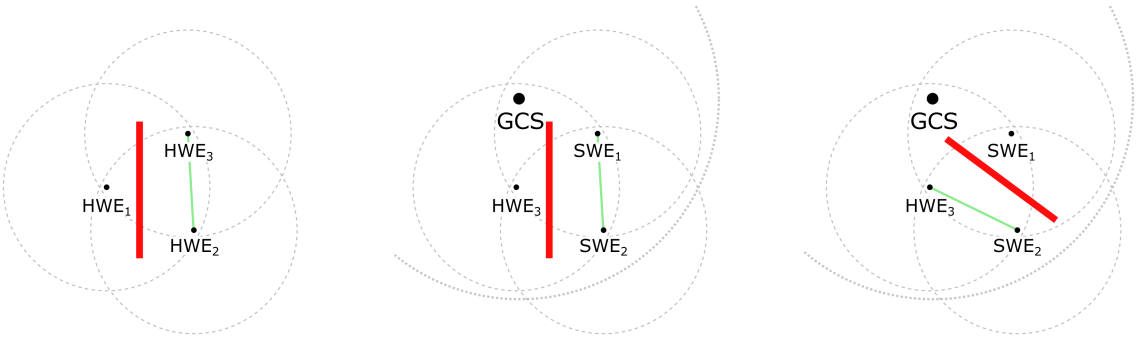
(a) (b) (c)

Figure 6.12: Mean values and standard deviation of data-rates of transmissions from neighbors measured at entities (a) SWE_1; (b) SWE_2; and (c) HWE_3 in MR scenario without the use of the high-fidelity MR architecture in case of sending data with the rate of 20 kBps.

more data than it is physically capable. There is a significant drop in transmitted data rates and a significant increase in their variance. The Microhard modems have their MAC mechanisms, but they are not satisfactory in this case, and many packets are lost. In this setting, the SWEs transmit much more data between each other than to the HWE. Thus this simple simulation of the communication cannot be used for high-fidelity validation of the communication system.

6.4.2 Verification of Performance on Physical Layer

The performance of the MR simulation in the PHY layer is demonstrated in Figure 6.13. The message receivers are pruned when three nodes send messages to each other and when an obstacle that blocks the transmission is placed between them. In all cases where SWEs are present, the wireless network simulation block would (as shown in Figures 6.5 and 6.7) intercept all messages from and for SWEs and prune their receivers according to their visibility from senders. Note that, for the correct operation of this functionality, the GCS needs to be aware of the obstacles and have connectivity with the HWEs, as specified in Section 6.2.4.



(a) (b) (c)

Figure 6.13: Scenarios showing the system behavior during message exchange under the presence of a communication obstacle (red). In all cases, each entity tries to send a message to the remaining two. The messages are exchanged only between entities that have mutual obstacle free connectivity (green lines). The rest of the messages are dropped either directly by the effect of the obstacle in the HW scenario or by the network simulation within the GCS in the MR scenario.

6.4.3 Verification of Network Simulation

To verify the network simulation within the *Simulation server*, a module *aglobex.wireless* was implemented as an extension of multi-agent simulator A-globe [177] and compared with SWANS++ [9] network simulator. SWANS++ is an open-source simulator with active user support and accuracy comparable to that of the GloMoSim simulator [211]. Unlike other network simulators, it is not too complex, thus changes of its behavior were easy to implement.

The modeled wireless transmitters in the testbed were not moving for better comparison of the results, and they were placed in the modeled environment with 100 m spacing, according to Figure 6.14. Free space path loss model [161] was used to model the signal propagation and no obstacles were placed in the environment. The transmitters used 2 GHz frequency with 2 Mbps bandwidth. The transmission power was set to 1 dBm, antenna gain to 1 dBi, and sensitivity threshold of the receivers was set to -81 dB. Thus, the transmission range encompassed only the nearest neighbors as shown in Figure 6.14. The hosts were sending 1 kB unicast messages to their neighbors with the rate of 5 messages per second for the period of 5 minutes. For better comparison of the simulators, two modes of SWANS++ were tested. One with the RTS/CTS mechanism in the message exchange, denoted *swans RTS* in the following graphs, and one without the RTS/CTS frames denoted *swans no_RTS*.

The examined simulation properties were average delay in message delivery time, ratio of the lost messages to all sent messages, and the network throughput. All of these properties were measured as functions of the number of simulated transmitters and are reported in the following paragraphs.

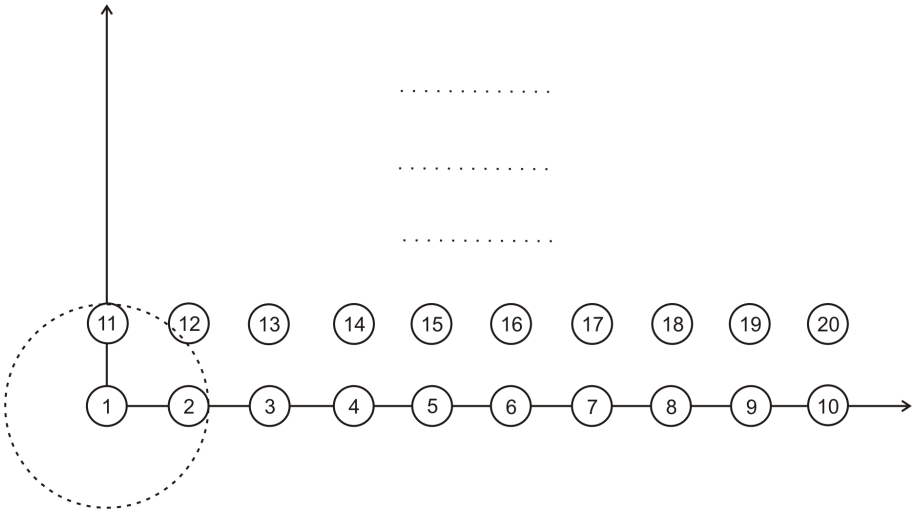


Figure 6.14: Nodes' placement used for the comparison of network simulation characteristics.

Message delay – The comparison of the average times necessary for message transport are shown in Figure 6.15. The results of *aglobex.wireless* module and SWANS++ simulator are similar for up to 75 transmitters. With increasing number of nodes, the delays in *aglobex.wireless* module grow much faster. This is caused by the fact that SWANS++ drops messages that are about to be send by the MAC layer when that layer is not in the idle mode (i.e., is sending a frame or is expecting a frame). Instead, the *aglobex.wireless* module is queuing all the messages which causes the average delivery delay to accumulate. This behavior is protocol dependent and can be adjusted by decreasing the capacity of the incoming message buffer of the *aglobex.wireless* module. The delay differences in scenarios with less nodes are caused by the discrete nature of module's message handling discussed earlier.

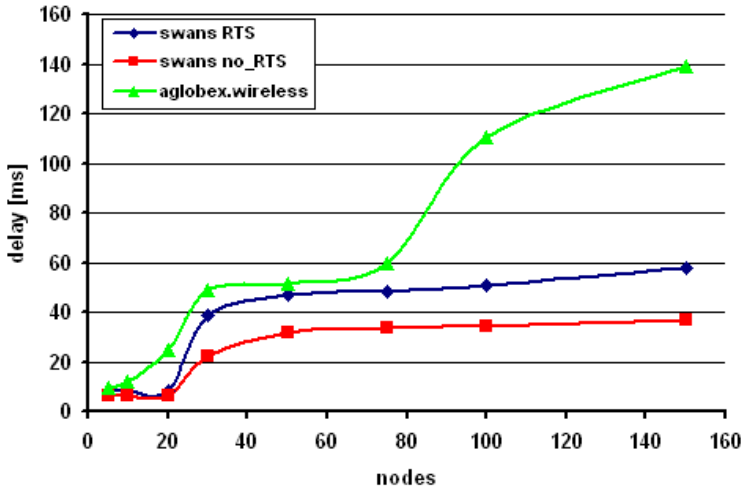


Figure 6.15: Average message delays in milliseconds in dependence on the number of transmitting nodes in the scenario.

Lost messages – The comparison of dropped messages ratios are depicted is Figure 6.16. A message can be lost due to interference or receiver's movement out of the sender's transmission range. In these cases, the message is re-transmitted for several times and if it still does not succeed to be delivered, it is dropped.

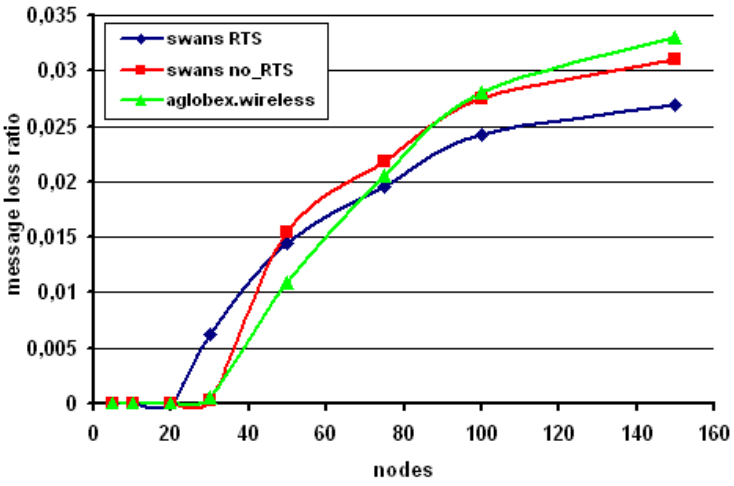


Figure 6.16: Ratio of lost messages to all sent messages depending on the number of wireless hosts present in the scenario.

Channel throughput – The overall network throughput was measured as a function of the number of successful transmissions. With rising number of transmitters, the wireless channel can no longer send all the messages and the network throughput decreases. The resulting channel throughputs are shown in Figure 6.17.

6.5 Remarks

In this chapter, we have present the communication architecture of MR simulations, that allows co-existence and interaction of virtual and physical entities within a single multi-vehicle system. This architecture was first presented by Selecký et al. [171]. The system implementing this architecture can be utilized as a high-fidelity network simulator for vehicular systems with implicit mobility models that are given by real trajectories of the vehicles.

The presented experiments validated the proposed communication architecture and showed that testbeds with hardware entities are substitutable with the MR testbeds, where some of the entities are modeled, and that both configurations can be used without significant difference and effect on the simulation accuracy. Further, the proposed wireless network simulation was compared with the network simulator SWANS++ giving similar results in terms of the main communication characteristics such as target visibility, message delay, data throughput or lost message ratio.

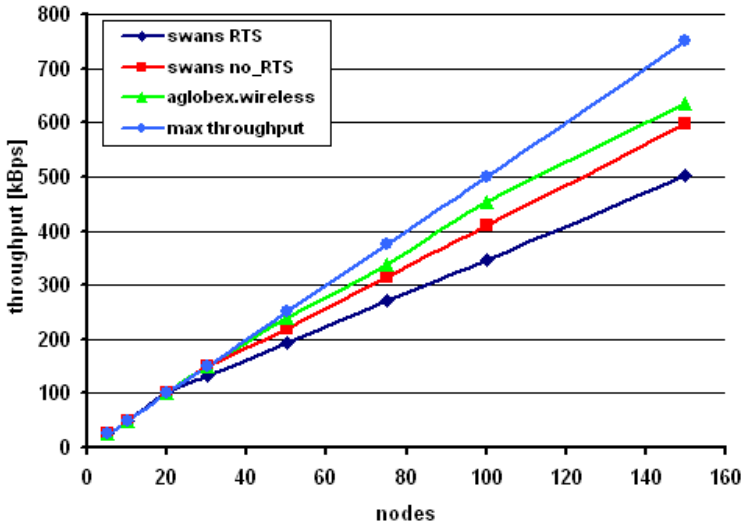


Figure 6.17: Channel throughput in kbps depending on the number of wireless hosts present in the scenario.

For the future work, we would like to extend the approaches discussed in Section 6.2.4. In particular, we aim to consider the adaptation of the dynamic transmit power of the simulation platform's modem and exploit usage of multiple directional antennas and distribution of large transmissions over the time to prevent transmission requests from virtual entities that would collectively exceed the available data-rates of the platform's modem.

Chapter 7

Environmental Modeling in MR Simulations

MR simulations allow interaction of physical and virtual entities and objects, and thus need to aggregate data from both the virtual world and the physical world. Entities can be equipped with both virtual and physical sensors and actuators, and they can sense and interact in both worlds depending on their virtualization level and position in the MiRA cube [82]. Thus, it is important to manage the propagation of actions and sensory data at various levels of virtualization. The *Simulation server* builds a model of the environment based both on the pre-configured environment data, and on the aggregated observations of the entities in the simulation. This chapter describes the management of sensory data and actuator actions on the level of the *Simulation server*. Possible issues in data management in MR systems are presented, together with approaches that address them. Finally, a necessary update of the MiRA Cube [82] taxonomy is presented in Section 7.6.

7.1 Requirements on Data Management

For a correct function of the MR simulation and for utilization of its benefits for the system development, the MR simulation framework should meet the following requirements on the management of sensory data and actuator actions.

- Framework should allow both the physical (HW) and virtual (SIM) entities—referred to as HWE and SWE, respectively—to explore and move in an unknown interactive environment while maintaining a "cognitive map" of what they have perceived by their HW and SIM sensors.
- Framework should be able to deal with changing environments, where objects can appear, move, or disappear without prior notice.
- Framework should determine which objects are currently observable by particular entities.

- Framework should ensure propagation of actions and sensory data between virtual and physical worlds.

7.2 Actions and Sensory Data Propagation

Typical MR simulation framework consists of the *Simulation server*, purely virtual entities, entities with various levels of virtualization, and possibly stationary physical sensors, such as anemometers, surveillance cameras, intrusion detection sensors, etc. Each entity in the simulation maintains its internal model of the environment. It can plan collision-free trajectories inside such environment and can interact with it, e.g., in target tracking missions. The environment model of the *Simulation server* of a multi-UAV system consists of several parts.

- Model or map that represents static obstacles and terrain, e.g., occupancy grid model, height map, set of geometric objects, etc. This map is usually known to all the entities and should allow for fast model update and fast estimation of collision-free trajectories.
- Set of *No-Flight Zones* (NFZs) which can be inserted and updated by the system operator. NFZs are of various geometrical shapes and are either static or temporal. They should be distributed to all the entities.
- Set of dynamic obstacles and objects that can be interacted with, such as moving targets, base stations, etc. These objects are observed only by those entities, in the sensor range of which they occur. This set of objects is also the source of the MR simulation issues discussed later on.
- Other entities, either virtual or physical. These entities are either interacted with via communication channel (as discussed in Section 6.2) or using sensors similarly to the previous part.

Figure 7.1 shows the flows of information inside the MR simulation framework. The *Simulation server* is updated by stationary physical sensors and physical sensors of the MR UAVs, and by actions of virtual actuators. *Simulation server* distributes information from the both the virtual and physical world to all virtual sensors. Physical world can be influenced only by physical actuators. Before passing any information to a virtual payload sensor, the *Simulation server* needs to check, whether the observed phenomena is in the respective sensing range, and whether the sensor is capable of reception of such phenomena. The server thus needs to be aware of the virtual sensor models.

7.3 Observations history

Autonomous entities need to posses a memory of observations, i.e., keep a history of observations of both virtual and physical objects to be able to act upon objects that are currently outside

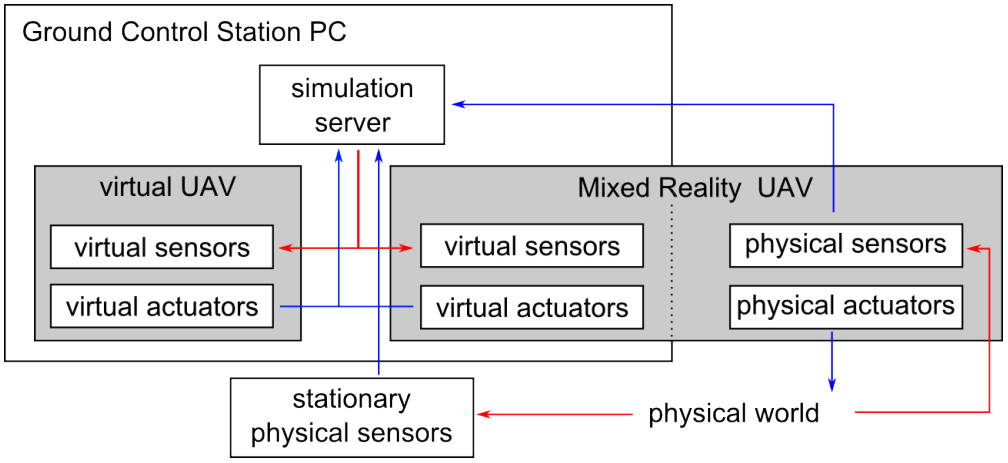


Figure 7.1: Effect of sensory readings and actuator actions in MR simulation. Red arrows represent information being sensed and blue arrows represent information that updates the state of either virtual or physical world.

their sensing range. For example, fixed wing aircraft loitering above a ground target can lost a track of the target for a short period of time. Nonetheless, it needs to act as it knows the target's position for a successful mission execution. For this purpose, each entity should keep a history in a set Γ of observations of all visible objects (targets, obstacles, etc.) and observable environment properties (e.g., a wind map). Let these observations be denoted as Ω and represented by a tuple $\langle objID_i, T_i, P_i, t \rangle$, similarly to the approach taken by Kuffner and Lamota in [106]. Components of these tuples are following

$objID_i$	the ID of the object i
T_i	the position and attitude of the object i
P_i	any properties of the object i
t	time of the observation

Objects that are not currently visible but have been observed in the past, keep their most recent previously-observed states. Currently visible objects are added to the set of known objects if observed for the first time, otherwise they are just used for update of their state. Special attention needs to be paid to the *Simulation server*. It has continuous knowledge of all virtual objects, but in a case of the physical world, the server has to rely on discrete observations from physical sensors. Thus the server needs to operate with the concept of a memory similarly to the entities.

7.4 Inconsistencies in Data Representation

When virtual and physical sensors detect the same objects or class of phenomena, they may produce inconsistent results that the MR simulation designer have to handle. Two types of such inconsistency can be seen in Figure 7.2 and they are following.

- *Unknown object interaction* is a problem with physical objects that were not yet observed by HW sensors and that would be sensed with SIM sensors otherwise.

- *Non-actual observation* is a problem with dynamic physical objects. Particularly, when a physical object was not observed for a long time by any HW sensor and it moved away while being observed by SIM sensors.

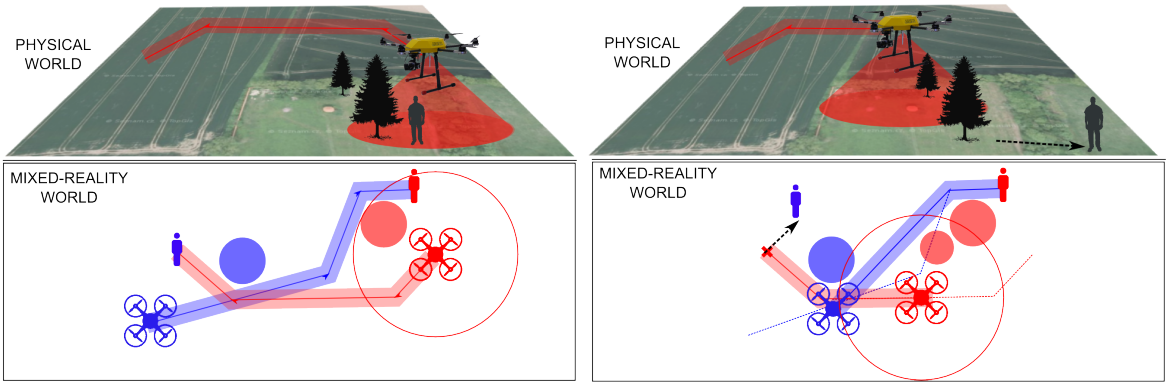


Figure 7.2: Inconsistencies in MR simulations. Physical objects are depicted in red, virtual objects are blue. Cause of the *Unknown object interaction* is the virtual UAV not knowing about the second tree and example of the *Non-actual observation* is the non actual position of the physical person.

An example of the *Unknown object interaction* would be a stationary object (e.g., a new grown tree, newly built walls or buildings, etc.), that was not yet observed by any HW sensor and so it is not yet included in the world map of the *Simulation server*. This problem is hard to solve once such inconsistency in the map and the physical world exists. It must be prevented by providing the entities with a up-to-date map of stationary obstacles and by frequent coverage of all dynamic obstacles by HW sensors. An example of the *Non-actual observation* would be a person moving in the physical world detected at position p_1 by a UAV equipped with physical sensor. This observation is passed to the *Simulation server* and distributed to virtual UAVs that are close enough to observe p_1. However, when the physical UAV no longer observes the moving person, the *Simulation server* distributes the old position p_1. Virtual sensors thus incorrectly sense the person as stationary. Two precautions need to be taken to prevent this problem: (i) virtual sensors must be prevented from updating the information about physical objects, even though they can still sense them; and (ii) dynamic properties of the physical objects and phenomena must be taken into account.

7.5 Dealing with Dynamic Environments

Several approaches can be taken to address the dynamic properties of observations. Traditional methods treat measurements of dynamic environment states as outliers [14]; however, this approaches lead to a loss of information that could provide support for long-term autonomy of robots. Kucner et al. [105] learn conditional probabilities of neighboring cells in an occupancy grid to model typical motion patterns in dynamic environments. Neubert et al. [132] presented a method that can learn appearance changes based on a cross-seasonal dataset and use the learned model to predict the environment appearance showing that state prediction can be useful for long-term place recognition in changing environments. Finally, Krajnik et al. [103] represent the probability of the elementary environment states by combination of harmonic functions whose amplitudes and periodicities relate to the influences and frequencies of the hidden processes that cause the environment variations.

An approach directly applicable to the system formalization presented so far was presented by Kuffner and Lamota in [106]. It suggests to use a *forgetting* mechanism by utilization of the *temporal model* for memory updates where the observations are remembered only for a certain period of time. In this case, old observations are periodically deleted from the memory of the *Simulation server*. Alternatively, in the *logical model* as defined in [106], updating the memory involves logical rules regarding observations of certain objects. In the case of objects sensed by the HW sensors, the *Simulation server* keeps a note of their information age and a class specifying their rate of change (*unstable/stable*—e.g., a person's position or wind vector, versus a wall or a tree). When the information age reaches the limit given by the object class, the *Simulation server* removes the sensed object from its memory. This prevents the virtual sensors from creating wrong beliefs about the physical objects. However, this process requires online classification of the objects to unstable/stable classes.

7.6 MiRA Cube Taxonomy Update

We have shown that some entities are allowed to update the state of MR simulations while others are not. Thus, the taxonomy of the MiRA cube was updated with one more characteristics — the possibility of an agent to update the state of the simulation world. This characteristics (called *update capability*) define whether the entity can update objects of the physical world, virtual world, both or neither in the simulation. Table 7.1 shows this updated taxonomy for possible UAV configurations in the MR simulation.

In Table 7.1, *pure HWE* represents an entity that is fully embodied in the physical (P) world with no interaction with the virtual (V) world. An example of such entity is a remotely piloted aircraft collecting data in physical world — its corporeal presence (CP), and interactive capacity (IC) is in the physical world only and its update capability (UC) affects only the physical world. An opposite of such an entity is *pure SWE* that has no interaction with the physical world. It

Table 7.1: MiRA Cube taxonomy for various configurations in common MR scenarios.

Type	***CP***	***IC***	***UC***
pure HWE	only P	only P	only P
pure SWE	only V	only V	only V
HWE with MR inputs	stronger P, weaker V	equal V and P	both V and P
SWE with MR inputs	stronger V, weaker P	stronger V, weaker P	only V

Legend: CP (corporeal presence), IC (interactive capacity), UC (update capability), P (physical), V (virtual), HWE (hardware deployed entity), SWE (virtual entity). All the configurations possess a strong agency feature in the MiRA cube.

is presented only in the virtual world, can interact only with virtual objects and its sensing can update only states of the virtual world.

UAVs in MR simulations have CP in both the physical and virtual world. The strength of CP and IC in these worlds depends on the overall virtualization level of the entities. HWE with MR inputs represents a hardware UAV that has strong physical CP, it is embodied in the physical world and can sense there. Example of such an entity would be a hardware UAV with a camera avoiding virtual obstacles. This UAV would be present in both P and V worlds, would be able to sense in the P world using the camera (e.g., for detecting physical obstacles) and in the V world using virtual sensors mediated by the *Simulation server* (e.g., for detecting virtual obstacles) and react on both sensory streams. SWE with MR inputs represents a virtual UAV that has strong virtual CP. It can sense only in the virtual world (e.g., for virtual obstacles), or at most, it can process sensory information from HWEs mediated by the *Simulation server* (e.g., wind speed, or positions of other UAVs).

As the table shows, a virtual UAV in MR can sense in both worlds but updates the common knowledge only in case of the virtual world objects. On the other hand, hardware UAVs in MR can update the knowledge about physical objects by direct measurements and, if connected to the *Simulation server*, they can also distribure the knowledge about virtual objects received from the server.

Chapter 8

Interaction with the Environment

This chapter addresses the interaction between the entities and MR environment. The particular problem is to plan a realistic trajectory that can be followed by fixed wing UAVs under windy conditions. To address this problem, it is necessary to take the wind influence into the account during the trajectory planning.

Small fixed-wing UAVs are significantly influenced by the effects of wind. Wind drifts the UAVs away from their original trajectory and influences their minimal turn radius and ground speed. The impact on the UAV's flight is most significant in the horizontal plane because of (i) better maneuverability of a fixed wing UAV in the vertical plane and (ii) slower vertical winds in the lower layers of the atmosphere. The vertical winds can be on the other hand utilized for energy efficient flights. Sudden gusts of wind can push the aircraft several meters away from its original trajectory thus decreasing the plan execution precision. Since these gusts are difficult to predict or map, their effect is addressed by using the safety separation zone around the aircraft during the planning.

In this chapter, we describe an approach for trajectory planning for non-holonomic UAVs under horizontal wind with constant strength and direction. The herein presented approach was published in [172]. First, the planning problem is stated using modified Dubins vehicle maneuvers [53] in Section 8.1. Then, the principle of Accelerated A* (AA*) algorithm [199] is explained in Section 8.2 and proposed changes for planning in windy conditions are presented in Section 8.3. Experimental results are reported in Section 8.4.

8.1 Problem Statement

The dynamics of a fixed wing UAV is often modeled as the Dubins vehicle, which is moving forward at a constant velocity v and with the limited turn radius ρ. The state of the vehicle q can be represented as the configuration $(x, y, \theta) \in SE(2)$, where $(x, y) \in \mathbb{R}^2$ is its position p in the plane and $\theta \in \mathbb{S}^1$ is the heading of the vehicle. The dynamics of the vehicle can be then

described as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{u}{\rho} \end{bmatrix}, \quad |u| \leq 1, \tag{8.1}$$

where u is the bounded control input.

Dubins proved that the optimal path connecting two configurations $g_1 \in SE(2)$ and $g_2 \in SE(2)$ consists of only straight line elements (further denoted as S) and turn segments (further denoted as C) with the minimal turning radius ρ [53]. He also proved that such an optimal path can have at most 3 segments (where segments can have zero length) that can be categorized in two main types:

- CCC type: LRL, RLR;
- CSC type: LSL, LSR, RSL, RSR,

where R and L stands for right turn and left turn, respectively.

Using the turn elements in presence of wind, the aircraft still moves in circles in the air mass; however, its movement is trochoidal according to the ground. Trochoid is a locus of a point (x, y) orbiting at a constant rate around an axis located at (x', y') with the parametrization

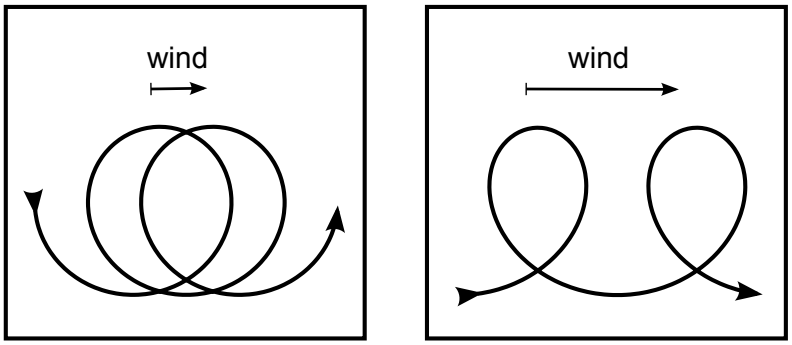
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' + \rho\cos(\omega t) \\ y' + \rho\cos(\omega t) \end{bmatrix}, \tag{8.2}$$

where ρ is the radius of the orbit corresponding to the minimal turn radius of the Dubins vehicle, and ω is the angular speed. The axis is being translated in the xy-plane from position $p = (x_0, y_0)$ at the constant speed $\vec{w} = (w_x, w_y)$ in the straight line

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x_0 + w_x t \\ y_0 + w_y t \end{bmatrix}. \tag{8.3}$$

When applied to the flight in wind, the axis translation speed $\vec{w}$ corresponds to the wind speed and the angular speed ω corresponds to an angle α traveled on a circle with the radius ρ in the time t at the airspeed $\vec{v_a}$.

McGee et al. [118] shown that flight curve depends only on the aircraft's turn radius ρ and on the wind speed to aircraft's air speed ratio $\beta = \frac{|\vec{w}|}{|\vec{v_a}|}$. Figure 8.1 demonstrates the effect of wind on an aircraft flying with a constant turn radius. In [17], the authors show that if the ratio of the wind speed and the current airspeed $\beta < 1$, then any two points with arbitrary orientations are reachable for a vehicle with Dubins type kinematics operating in the presence of wind. Thus, the approach presented in this chapter is valid for wind speeds up to the maximal airspeed of the UAV.



(a) $\beta = \frac{|\vec{w}|}{|\vec{v_a}|} = 0.1$ (b) $\beta = \frac{|\vec{w}|}{|\vec{v_a}|} = 0.3$

Figure 8.1: Effect of wind on the flight trajectory.

8.2 Accelerated A* Algorithm

Accelerated A* (AA*) is a modified version of the A* algorithm presented in [199] that overcomes the trade-off between efficiency and search precision by using an adaptive sampling mechanism. The algorithm generates samples using elementary flight maneuvres where the length of the maneuvres is given by the aircraft's distance to the nearest obstacle—the further the obstacle the sparser the sampling and vice versa (see Figure 8.2). The search precision is then defined by the minimal sampling step. The AA* algorithm finds optimal trajectories with respect to the sampling and using the adaptive sampling mechanism, it is typically faster than the original A* algorithm.

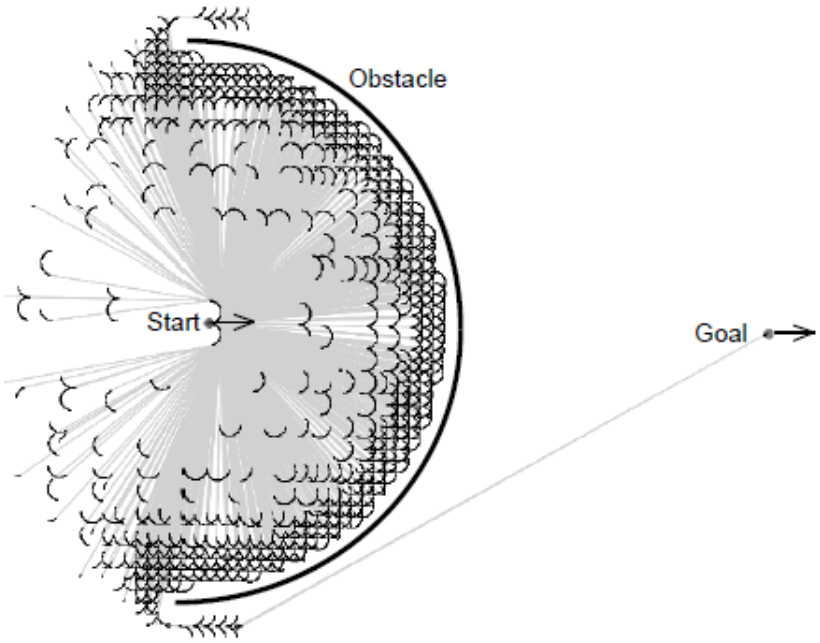


Figure 8.2: Example of the adaptive sampling. Adapted from [199].

Maneuvers that define the sampling correspond to the aircraft's elementary actions: *straight*

element that is given by the start configuration and its length; and *horizontal turn* and *vertical turn* elements that are defined by the turning radius, arc angle and turn direction—left or right, and up or down, respectively. The algorithm works similarly to the regular A*, see Algorithm 1. The input to the algorithm are two configurations—the start configuration pc_S and the goal configuration pc_G which contain positions and orientation constraints. The output is the flight plan Φ. Initially, the validity of the input configurations is checked, i.e., the positions are not in the restricted space (line 2). The function `Connect`(pc_{from}, pc_{to}) provides the shortest flight plan from the configuration pc_{from} to the configuration pc_{to} respecting the airplane's nonholonomic model but not considering the restricted airspace. The state of the Accelerated A^* algorithm is defined as a tuple

$$s = \langle pc, \epsilon, fp, u, g, h, s^{parent} \rangle. \tag{8.4}$$

The planning configuration pc contains the position and orientation of the state s, the sampling step ϵ identifies the sampling step for the state's position, and the flight plan fp contains the flight trajectory for transition from the planning configuration of the predecessor state s^{parent} to the planning configuration pc of the state s. In contrast to the plan provided by the function `Connect`, the flight plan fp respects both the nonholonomic constraints and the restricted airspace. The attribute $u \in \{\texttt{true}, \texttt{false}\}$ specifies whether the shortest connection from pc to the goal configuration pc_G is valid. The value g is the cost of the path from pc_S to pc and the heuristics h contains estimation of the cost of the path from pc to pc_G. The heuristics estimation is discussed in Section 8.3.3. The component s^{parent} is the reference to the predecessor state.

The algorithm takes the best state s_C from the OPEN list and if it can be directly connected with the goal state (line 10), it smoothes the found path by application of Dubins maneuvers [53]. Otherwise, it expands the current state by application of the flight elements, estimates the new position and new sampling (lines 15–17), checks the presence of this new state in the CLOSED list and checks the validity of the element, counts heuristics using Dubins curves, and inserts the new state into the OPEN list if possible (line 23). The sampling of continuous space can generate an infinite number of states where none are exactly the same. Thus, two states are considered equal if their positions and direction vectors differ less than a value given by the sampling parameterization.

8.3 Proposed Algorithm Changes

The original AA* algorithm does not take wind into consideration, and thus, in windy conditions, the planned trajectories are hard or impossible to be followed. The impact of wind on UAV flight is most significant in the horizontal plane because of (i) better maneuverability of a fixed wing UAV in the vertical plane and (ii) slower vertical winds in the lower layers of the atmosphere. That is why we modeled wind as a two-dimensional vector and modified only the horizontal

Algorithm 1 AA* algorithm pseudocode

Require: Input and goal configurations pc_S, pc_G
Ensure: Flight plan fp^{result}

function FINDPATH(pc_S, pc_G)
  if `not isValid`(pc_S) **or** `not isValid`(pc_G) **then return** $\emptyset$
  $fp^{end} \leftarrow$ `Connect`(pc_S, pc_G)
  $s_S \leftarrow \langle pc_S, \texttt{DetectSamplingStep}(pc_S), \langle\rangle \texttt{ IsValid}(fp^{end}), 0, \textbf{cost}(fp^{end}), - \rangle$
  $OPEN \leftarrow \{s_S\}$
  $CLOSED \leftarrow \emptyset$
  while $OPEN \neq \emptyset$ **do**
    $s_C \leftarrow$ `RemoveTheBest`$(OPEN)$
    `Insert`$(s_C, CLOSED)$
    if u^{s_C} **then**
      $s_G \leftarrow \langle pc_G, -, \texttt{Connect}(pc_{s_C}, pc_G), \texttt{true}, g^{s_C} + h^{s_C}, 0, s_C \rangle$
      $s_G \leftarrow$ `SmoothPath`(s_G, pc_S)
      $fp^{result} \leftarrow$ `ReconstructPath`(s_G)
      return fp^{result}
    for all $fp_i \in$ `Expand`(s_C) **do**
      $pc_N \leftarrow$ `EndConfiguration`(fp_i)
      $\epsilon_N \leftarrow$ `DetectSamplingStep`(pc_N)
      if `Contains`$(pc_N, CLOSED, \epsilon_N)$ **then continue**
      if `not isValid`(fp_i) **then continue**
      $fp^{end} \leftarrow$ `Connect`(pc_N, pc_G)
      $s_N \leftarrow \langle pc_N, \epsilon_N, fp_i, \texttt{IsValid}(fp^{end}), g^{s_C} + \textbf{cost}(fp_i), \textbf{cost}(fp^{end}), s_C \rangle$
      $s_N \leftarrow$ `SmoothPath`(s_N, pc_S)
      `InsertOrReplaceIfBetter`$(s_N, OPEN, \epsilon_N)$
  return $\emptyset$

properties of the AA* algorithm. Three changes of AA* algorithm are proposed: (i) change in definition of the maneuvers; (ii) change in connection of two configurations; and (iii) change in the heuristic function.

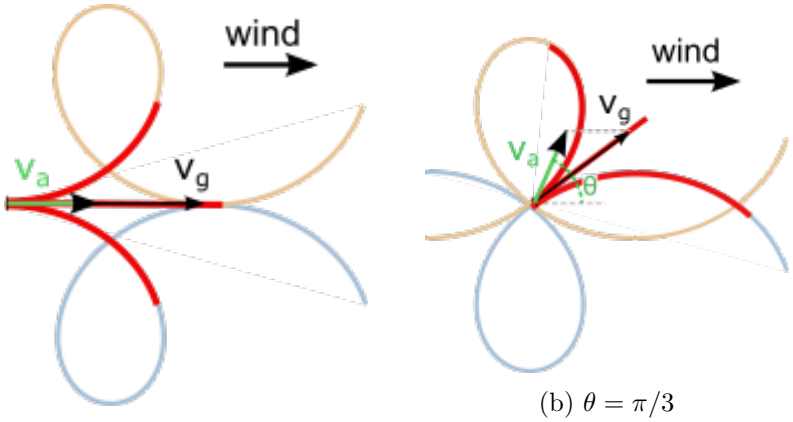
8.3.1 Modification of Maneuvers

New straight flight and turn maneuvers are shown in Figure 8.3 for different values of the aircraft's heading.

Turn maneuver – Trochoidal curves defined by (8.2) and (8.3) are used for new definition of the turn maneuver

$$x = x_0 + \rho \cos(\alpha) + \rho \frac{w_x}{|\vec{v_a}|} \alpha, \tag{8.5}$$

$$y = y_0 + \rho \sin(\alpha) + \rho \frac{w_y}{|\vec{v_a}|} \alpha, \tag{8.6}$$



(a) $\theta = 0$ (b) $\theta = \pi/3$

Figure 8.3: Modification of flight maneuvers for different start configurations.

where $\alpha \in [\alpha_0, \alpha_0 + \alpha_t]$, $\alpha_t \in [-2\pi, 2\pi]$. The parameter α_0 determines the initial position in the trochoid, the α_t corresponds to the maneuver's shift along the trochoid, see Figure 8.4. The aircraft's heading θ is used to estimate the value of α_0 and to correctly place the trochoids (see Figure 8.3).

New straight flight maneuver – The effect of applying the straight flight maneuver in wind is expressed by (8.7).

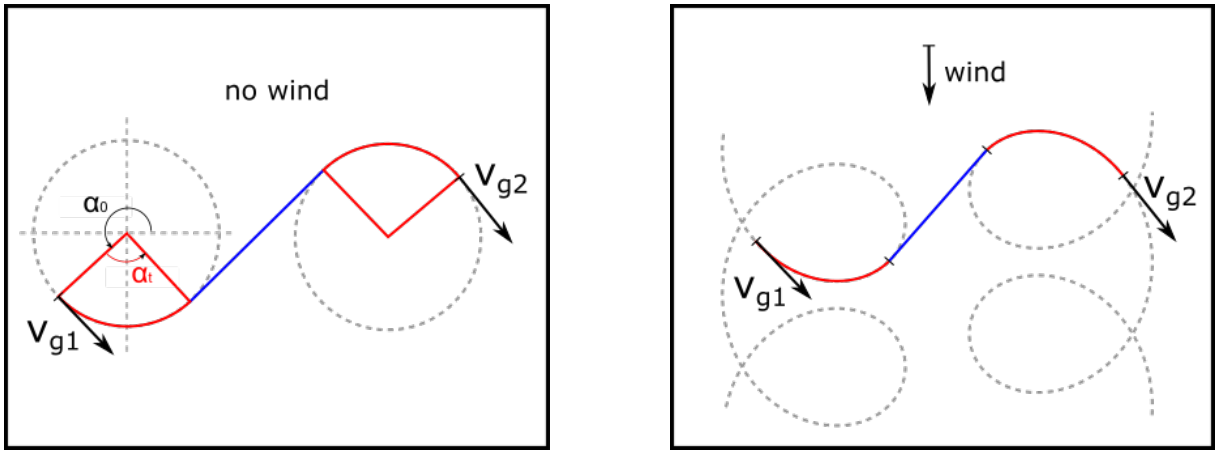
$$\vec{v_a} = \vec{v_g} - \vec{w} \tag{8.7}$$

8.3.2 Connecting Two Configurations

When planning a path between two state configurations without obstacles, we use Dubins maneuvers with trochoidal curves instead of the turn elements (see Figure 8.4b). This connection of three elements, called *Smooth maneuver*, is used for the estimation of heuristics and for smoothing the planned trajectory. The initial trochoid is placed so that it passes through the initial configuration and it is tangent to its direction vector. The second trochoid is placed similarly with respect to the second configuration. The shortest tangent to both trochoids represents the straight maneuver. The mechanism of finding the tangents differs according to the turns sides in the three consecutive maneuvers. We distinguish two cases: (i) turn – straight segment – same side turn, or (ii) turn – straight segment – opposite side turn (see Figure 8.5).

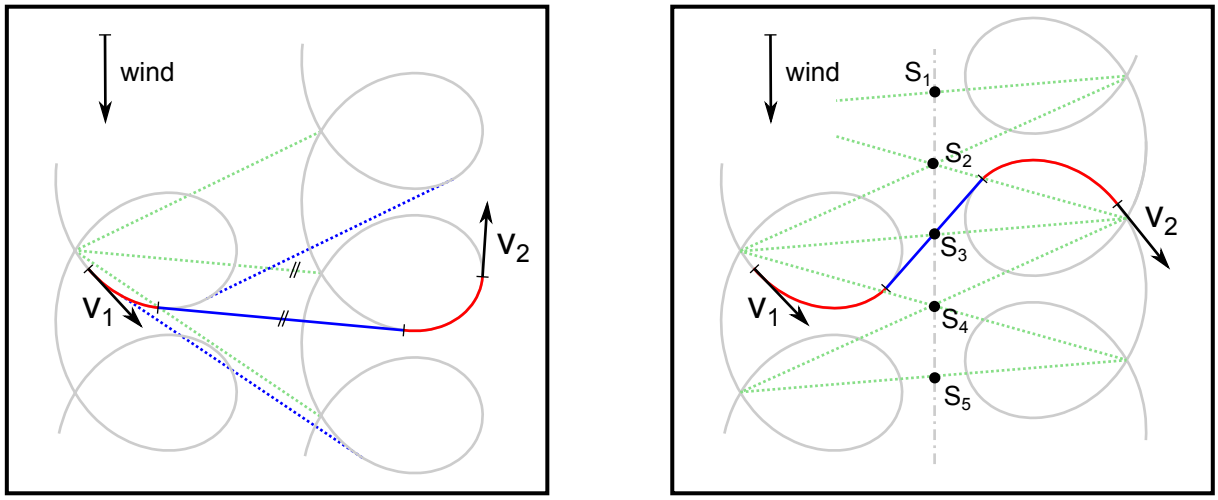
Same side turn – In this case, the trochoids are only translated one to the other so the direction of the straight segments connecting the trochoids must be parallel to one of the translation vectors, see Figure 8.5a.

Opposite side turn – In this case, the trochoids are centrally symmetric according to the points S_1,S_2,... and the straight maneuvers apparently need to pass through one of these points,



(a) $\frac{|\vec{w}|}{|\vec{v_a}|} = 0$ (b) $\frac{|\vec{w}|}{|\vec{v_a}|} = 0.25$

Figure 8.4: Modification of Dubins maneuver under the presence of wind.



(a) Same side turn - LSL (b) Opposite side turn - LSR

Figure 8.5: Connection of maneuvers. L/R means left/right side turns and S represents a straight flight maneuver.

see Figure 8.5b.

8.3.3 Modifications of Heuristic Function

The AA* algorithm uses Dubins maneuvers to estimate the heuristics for a given state [199]. However, McGee et al. [118] proved that for certain configurations in the presence of wind, non-Dubins maneuvers need to be applied for time-optimal planning. Thus, in some situations, it may not be possible to use the modified Dubins maneuvers for heuristics estimation and less informative methods would have to be used instead [62].

We proposed a so-called *irregular zone* around the current state to address this problem, see Figure 8.7. If the target configuration lies outside the zone, it is sure that our maneuvers

are time-optimal. For target configurations inside this zone, other than Dubins maneuvers may be necessary and thus we have to use some less informative heuristics. Another effect of this approach is that outside the *irregular zone*, only the CSC maneuvers can be time-optimal and thus the CCC maneuvers do not need to be considered.

Irregular zone – Using the idea of [118], in the presence of wind, the target configuration (called *virtual target*) moves and thus its trajectory needs to be checked for its presence in the *irregular zone*. The situation is depicted in Figure 8.6, where S is the initial state of the UAV and VT is the initial position of the virtual target. P_1 is the closest interception point reachable simultaneously by the UAV and the virtual target from their initial positions in the most convenient configuration (green). P_2 is the interception point in the case of the configuration that requires the longest optimal Dubins curve (red). Parameters d_{min} and d_{max} are distances, which the virtual target travels in the time necessary for the UAV to fly the distance l_{min} and l_{max}, respectively. For CSC maneuvers to be optimal, the whole line segment P_1P_2 needs to be outside of the circle with the radius 4ρ [27], see Figure 8.7. More details to this subject are presented in our previous work [172].

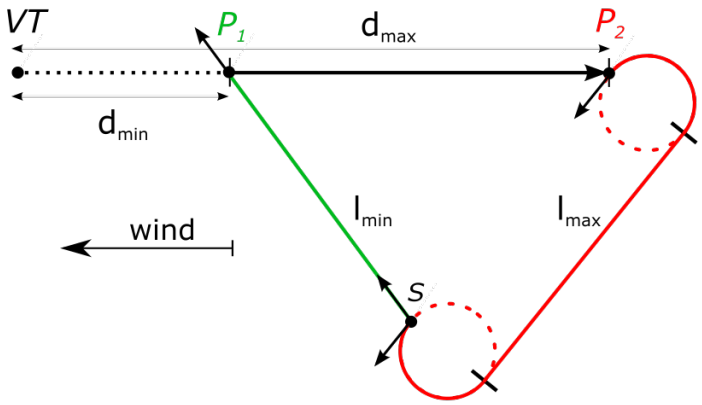


Figure 8.6: Trajectories of the moving virtual target and the aircraft in moving air frame.

8.4 Experimental Results

The proposed methods were verified by deployment on a real fixed wing UAV shown in Figure 8.8. Plan execution precision was compared in cases with plans generated by the original AA* algorithm and plans generated by the AA* modified for planning in wind. Tests were conducted in winds with the speed about 5 m.s^{-1} and different directions. The UAV's airspeed was 15 m.s^{-1}. The scenario consisted of three waypoints as shown in Figure 8.10. The blue curve represents the planned trajectory, the green curves show the real recorded trajectory, and black arrows emphasize a different turn radii at the particular cases. Red crosses are plan samples uploaded to the autopilot. The sampling period is proportional to the flight time in straight

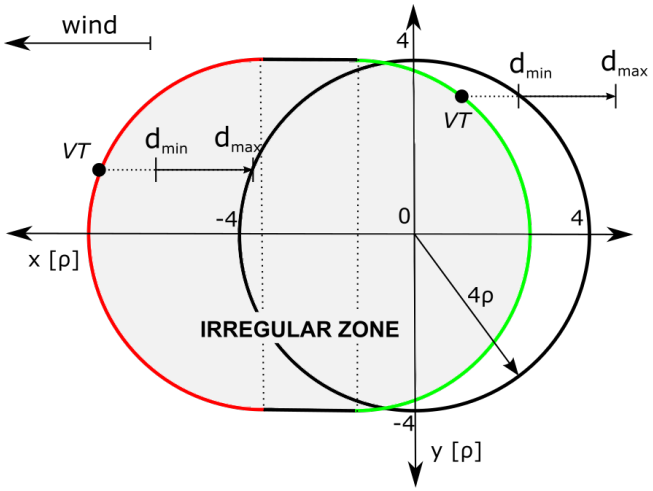


Figure 8.7: Construction of the irregular zone, outside of which we can use CSC Dubins maneuvers for heuristics estimation.

segments and to the turning rate in turn maneuvers. The reason why there is only a single real flight trajectory in Figures 8.10a and 8.10b is that the wind conditions were not stable for the whole period of the experiments.

Figure 8.8: Lockheed Martin Procerus fixed wing aircraft.

Trochoidal curves adapted to the wind speed and direction are followed much more precisely than the Dubins maneuvers. However, there are still errors caused mainly by wind gusts during the flight. The probabilities of the track following errors are shown in Figure 8.9.

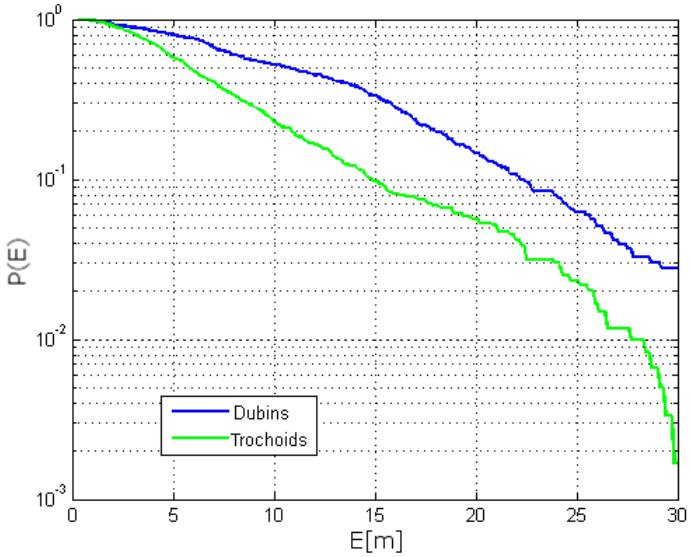
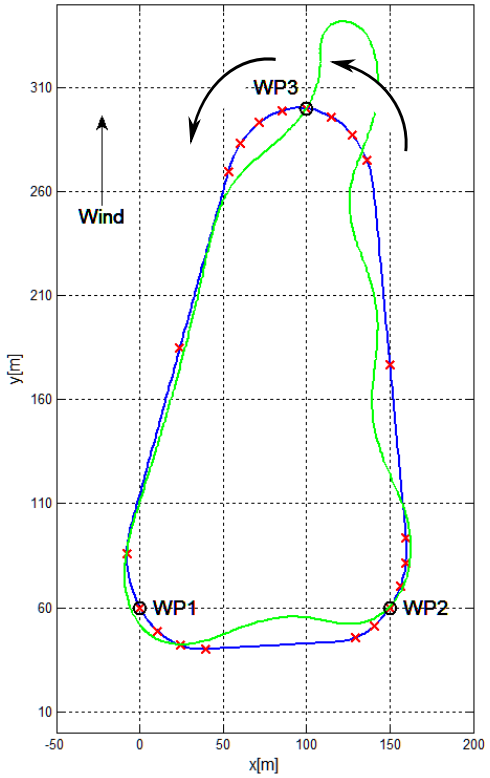
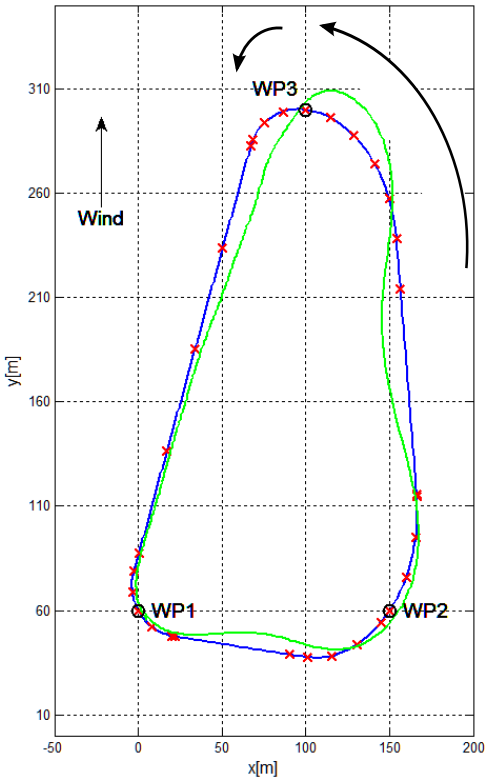


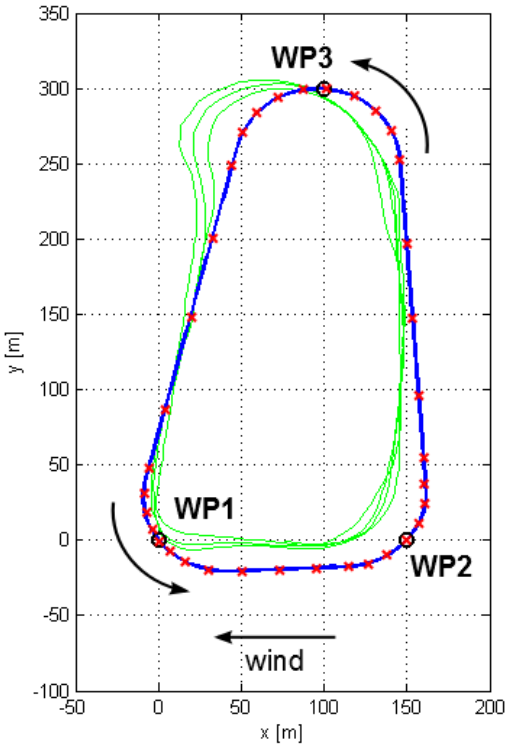
Figure 8.9: Probability distribution of the track following error.



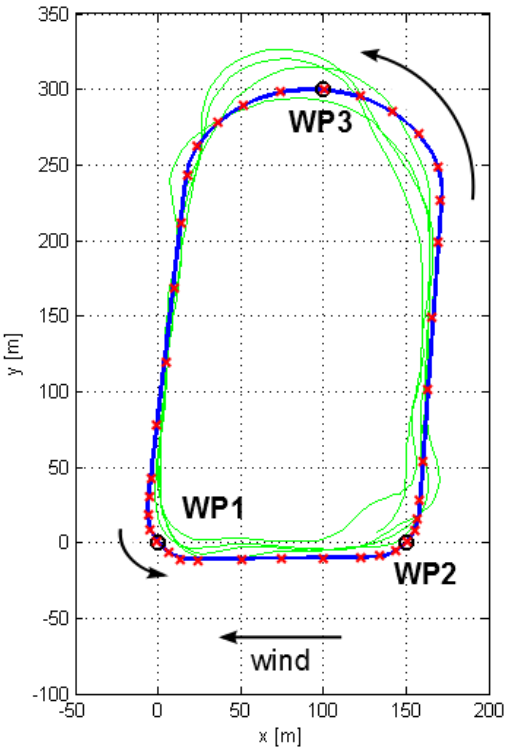
(a) Planning with Dubins curves in south wind



(b) Planning with trochoidal curves in south wind



(c) Planning with Dubins curves in east wind



(d) Planning with trochoidal curves in east wind

Figure 8.10: Planned and executed trajectories in different wind directions.

Chapter 9

Refining the Environment Model by Persistent Measurement

Significant source of imprecise results of simulations is the low fidelity of the environment model and its effect on the movement of entities and on other phenomena within the simulation (e.g., effect of wind on the movement of a toxic cloud, fire spreading, etc.). The wind modeling is one of the most significant sources of imprecision. In many open and flat environments, the models of the wind are relatively simple; however, in a case of more complex terrains, such as hills or urban areas, these models are difficult to be estimated precisely. Precise real-time models can be useful in time-critical situations like pollutant diffusion, forest fire forecasting, urban environmental prediction, or chemical leaking. This chapter addresses the problem of the refinement of the computational wind model by persistent wind measurements in urban environments.

Wind modeling in urban environments usable for forcasting has been an ongoing and difficult challenge [36], [77], [131]. Computational fluid dynamics (CFD) solvers can provide 3D wind field models of various environments; however, their precision and forcasting capability are strongly dependent on the quality of the initial conditions [144]. Models provided by the CFD simulations have limited accuracy [78] and can be significantly enhanced by the data assimilation, i.e., by incorporating real measurements into the state of a numerical model of the system [49], [85]. Real measurements can be acquired by static sensors; however, these fixed weather sensing systems are not practical to transport and install through dense urban settings, as they require significant infrastructure and workforce to support. Another option is to deploy mobile units, e.g., UAVs equipped with wind measurement sensors, which are much more versatile and allow for easy change of sensing locations if changes in the environment occur [115], [154].

The proposed approach addresses a problem of planning the best tours for a VTOL UAV with limited flight time for persistent measuring of a wind field in an urban environment. The task is to collect the maximum amount of the information per a single flight considering the spatiotemporal nature of the wind. The problem is illustrated in Figure 9.1. The proposed method can be used for the improvement of the wind field model of an urban environment

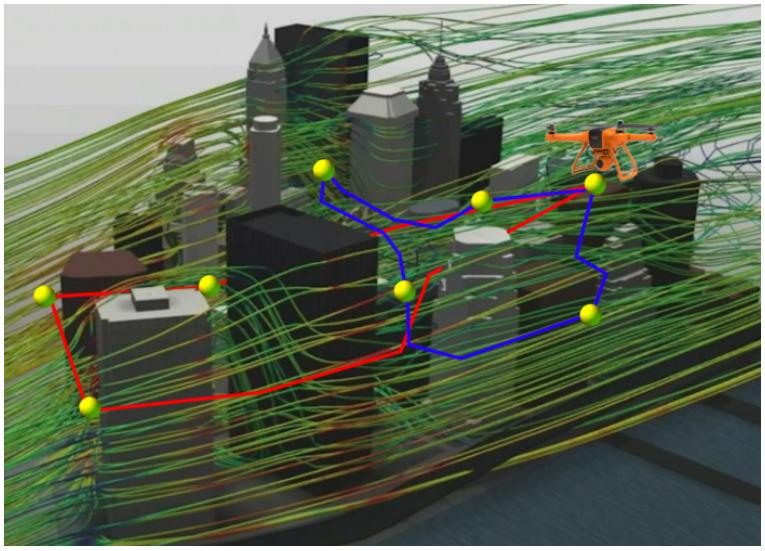


Figure 9.1: Illustration of the problem of repeated data collection of the spatiotemporal phenomenon in an urban environment. The task of the UAV is to collect the maximum amount of information per a single flight considering the spatiotemporal nature of the phenomenon. The background is borrowed from the presentation of Autodesk CFD solver.

created by a CFD solver. It should be noted that even though the herein proposed method is presented within the context of the wind measurement by a UAV, it can be generalized for any task aiming at a persistent observation of a spatiotemporal phenomenon. The solution of the problem consists of (i) a determination of a set of the measurement locations to address the spatial variability of the wind field; and (ii) a strategy for repeatedly searching for the best tours over these locations that would maximize the amount of the information gathered considering the local temporal variability of the wind field.

The proposed solution is based on the formulation of the problem as the *Repeated Orienteering Problem* (ROP) for capturing the temporal aspects of the wind field. ROP has three defining elements: (i) the target domain may be represented as a network of nodes with defined temporal variability of the observed phenomenon in them; (ii) the employed mobile robot (UAV) has a limited travel budget (e.g., a flight time) that is often insufficient for the total coverage of all the targets; and (iii) the previously performed data collection tours can affect the optimality of the sub-sequential tours. The third element arises from the observation that it may not be beneficial to visit the same node in two sub-sequential tours, but it may be beneficial to revisit it in later tours. Under these conditions, the goal is to repeatedly plan budget-limited tours for the robot(s) to maximize the information gain that is characterized by a function dependent on the temporal variability of the nodes, and on nodes visited in the previous tours. Within the context of the selected mission, the problem is to minimize the weighted information age of measurements over a long-term observation process, given the limited flight time and asymmetry of the problem given by flying in a wind field. The weights, in this case, reflect the need for more frequent visits at locations where the temporal variability of the wind vector is more significant

than at the other locations.

The rest of this chapter is structured as follows. Section 9.1 summarizes the related work. Next, the problem of persistent wind data collection in an urban environment concerning the spatial and temporal variability of the wind field is discussed in Section 9.2. The process of selecting the measurement locations addressing the spatial variability of the wind field is presented in Section 9.3 together with the proposed solution of ROP that targets the temporal properties of the wind field. The proposed solution addressing the particular part of the whole challenging problem is verified and evaluated for an artificial urban scenario, and the results are reported in Section 9.4. Section 9.5 gives final remarks on the subject of wind measurement.

9.1 Related Work

In this chapter, we address a problem of planning optimal tours for persistent measurement of a wind field with its spatiotemporal characteristics. It is a complex task that includes several subproblems, and to some extent, it is related to the sensor placement problem, the *Orienteering Problem* (OP), or the informative path and policy planning problems. Therefore an overview of the related work in these areas is reported in the following parts.

9.1.1 Sensor Placement

When monitoring spatial phenomena, choosing sensor locations is a fundamental task. The problem of sensor placement was addressed using multiple different approaches. The optimal sensor placement is usually calculated as the most informative locations by information theory criteria like entropy or mutual information, using spatial statistics, or as one that maximizes the spatial coverage. A common assumption is that the underlying phenomenon at one location can be modeled by a Gaussian distribution and the phenomenon over the target area can be thus considered as a *Gaussian Process* (GP).

The approaches that aim to select sensor locations to optimize the spatial coverage of the studied area are based on, e.g., locating sensors at the centroids of k-means clusters [203], or using Voronoi diagrams and Delaunay triangulation [10]. A survey on the sensor placement for applications in contamination detection is given by Hu et al. [84].

Approaches using geostatistical analysis often aim at sampling that minimizes the (mean) kriging error variance [45]. Konak [102] uses ordinary kriging to reduce the cost of active site surveys by estimating values of the phenomenon at locations where no measurement data is available using samples taken at other locations. Castello et al. [31] determined an optimal sensor placement scheme for environmental monitoring by minimizing the variance of the spatial analysis based on randomly chosen sensor locations. The authors used Monte Carlo analysis to optimize the selection of the locations.

The main drawback of the above mentioned approaches is that the spatial sampling is

adapted according to geometric criteria and does not take the characteristics of the monitored phenomenon into account. Some methods rely on ancillary data, such as aerial or satellite imagery, or climate information, which are directly correlated with the phenomenon of interest. Minasny et al. [124] proposed an algorithm for sampling with prior information on the environmental data. Their algorithm uses quadtree decomposition and successively divides the area such that each subarea has a more-or-less equal variation of the underlying phenomenon. Unfortunately, this approach is not applicable to more complex environments or environments with obstacles.

Information-theoretic approaches use entropy and mutual information to improve information quality by reducing uncertainty about the true state of the phenomenon. Guestrin et al. [73] proposed to use mutual information criteria, and they showed that it produces better placements than methods that place sensors at the points of the highest entropy (variance) in the GP model [98]. Du et al. [52] came with an optimal sensor placement scheme to measure the wind distribution with a limited number of sensors according to the entropy using heuristic algorithms. The authors employed CFD modeling to learn the spatial correlation of the wind in the presence of surrounding buildings.

Based on the literature review, the aforementioned approaches typically require sufficient prior knowledge of data distribution to train their models to capture pairwise correlations among different locations. In our proposed solution, we take inspiration of [52], and use a CFD simulation of the phenomenon to intercept its spatial properties.

9.1.2 Orienteering Problem

The *Orienteering Problem* (OP) is a routing problem in which for a given set of locations, each associated with a reward, it is requested to find a path that maximizes the sum of the collected rewards by visiting a subset of the given set of locations while the path length does not exceed the given travel budget [70]. There are several heuritic algorithms [37] for solving the ordinary OP, but also its variants [74], [194] including methods that consider path constraints [145] or even addressing the OP with Neighborhoods [58]. Besides, there is also a variant of the OP called the *Team Orienteering Problem* (TOP) [38] in which tours for multiple robots are planned. Therefore, various solutions of the OP exists and can be used for solving ordinary OP. The key difference between the OP and the herein introduced ROP is that the OP assumes only a single tour of the vehicle, while ROP deals with multiple sub-sequential tours.

9.1.3 Informative Path and Policy Planning

The problem addressed in this chapter is closely related to informative path and policy planning. Methods for dealing with a variety of persistent monitoring problems have been published, e.g., for graph-based settings [4], for the monitoring of a continuously evolving scalar field [181], or

for stochastic data harvesting [210]. Various application domains have been addressed such as aerial [121] and underwater [180] deployments.

Garg and Ayanian [65] developed an algorithm that maximizes entropy on a set of informative regions to monitor stochastic phenomena, where the underlying covariance structure changes sharply across the time. Stranders et al. [184] addressed persistent monitoring with submodular value function that used diminishing returns of observations, and that had a locality property where closer observations provide less information. Alamdari et al. [4] searched for infinite walks of robots that would minimize the maximum weighted latency of measurement at any vertex. These approaches; however, do not consider the limited flight time or necessity of take-offs and landings at particular places to take the measurements.

Other approaches use modifications of the *Traveling Salesman Problem* (TSP) formulation to model the limited flight times and other constraints. Ekici and Retharekar [55] used the *Multiple Agents Maximum Collection Problem with Time-Dependent Rewards* (MAMCP-TDR) to model the rewards and time properties of the solution. Yu et al. [209] modeled the problem as the Correlated Orienteering Problem (COP) where the correlation between the collection locations are taken into account for the reward estimation. Although these approaches are close to the herein address problem of UAV planning, they do not consider multiple sub-sequential measurements, and they provide only single tours.

Singh et al. [176] use informative path planning to address monitoring of spatiotemporal dynamics and maximize the amount of information collected while respecting the resource constraints such as a limited battery or limited amount of time to obtain measurements. This approach, however, does not allow for continuous monitoring, only such that is limited to the number of flights.

The approach proposed in this chapter addresses an efficient continuous persistent monitoring of a spatiotemporal phenomenon while taking prior information about the phenomenon into account. The monitoring is performed by a robot with limited travel budget that is requested to start and finish at the pre-specified locations. Based on the overview of the existing approaches, to the best of our knowledge, no approach addressing all these constraints has been presented yet.

9.2 Problem Statement

Spatiotemporal fields such as the wind field in an urban environment can be highly complex and dynamic, and it may change significantly over time. Since the underlying spatial structure (terrain, buildings, etc.) often does not change quickly, we can assume that even though the wind field can change over time, the nearby nodes have mostly time-invariant spatial correlations. Thus, in such an urban environment, areas of both high and low time variability can be created. The variation of the wind field in spatial and temporal domains can be observed from the CFD

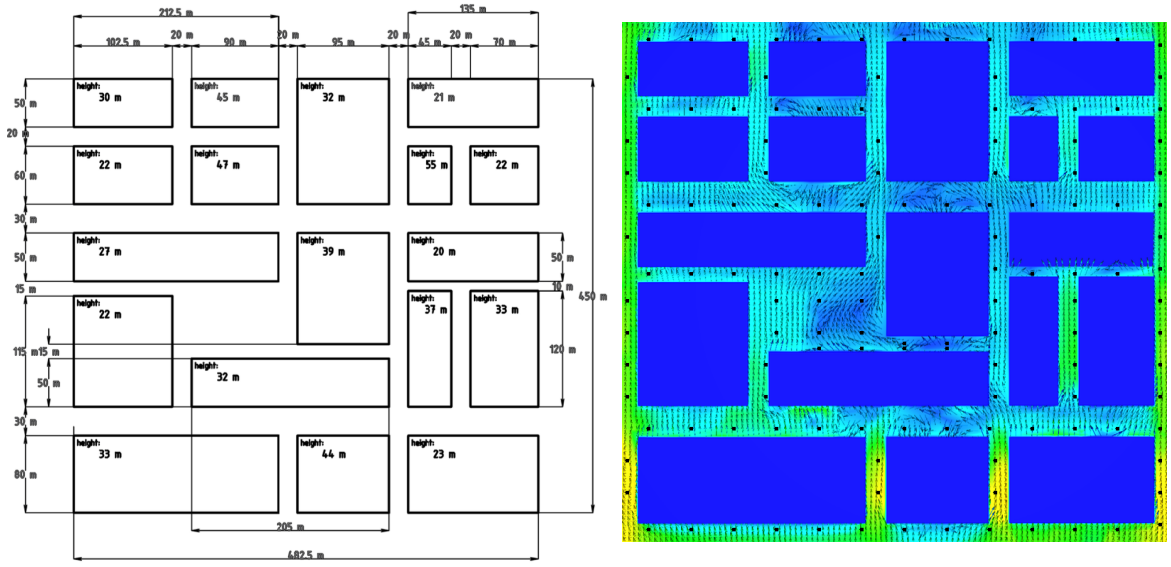


Figure 9.2: Environment model for the method verification showing proportions of obstacles and sampling grid.

models and can be taken into account during the data collection planning.

The studied data collection process is addressed in two stages to handle both the temporal and spatial variability of the wind. First, locations with the largest predictive value are selected into the measurement set. Second, a persistent data collection strategy is proposed that prioritizes visits of locations with a high temporal variability. Both the stages are further specified in the following text.

9.2.1 Selection of the Measurement Locations

A set of measurement locations need to be selected such that it maximizes the predictive features of the resulting computational model to address the spatial variability of the data. The urban area can be covered by a grid, a set of N possible measurement locations $p_i \in P \subset \mathbb{R}^2$, with omitted points inside buildings and obstacles, see Figure 9.2. Given limited flight time of UAVs and temporal variability of the wind, it is not possible to cover all N locations within relevantly short time. Therefore, the goal of the first stage of the proposed solution is to choose a subset Q of the fixed size $M < N$ of the measurement locations such that their predictive value about the phenomenon values in all N locations are maximized, i.e., the reconstruction error of the model is minimized. CFD models, however, are too complex and cumbersome to be used for this purpose. Thus, a much simpler model is needed to enable fast evaluation of the reconstruction error at multiple combinations of the measurement locations.

We assume that each point p_i is associated with a value z_i of interest and let u be a point

where the value z_u is unknown. The subset selection problem can be then expressed as

$$Q^* = \arg\min_{Q} \sum_{u \in P \setminus Q} (z_u - \hat{z_u})^2, \tag{9.1}$$

$$\|Q\| = M = const. \tag{9.2}$$

Since the value z_u of the measured phenomenon is unknown, it must be taken from the CFD model. The value $\hat{z_u}$ is estimated from the known values at locations from the set Q using a simpler model, in our case, using ordinary kriging which works as following.

Kriging analysis – Methods of geostatistics provide a simple model that can be used for fast estimation of the location set that minimizes the reconstruction error. A method of ordinary kriging is applied in the proposed approach. The ordinary kriging is an interpolation technique that gives the best linear unbiased prediction of the intermediate values of the measured phenomena [116]. The ordinary kriging models the interpolated values of a phenomenon by the Gaussian process governed by a prior covariance. The method assumes that each point p_i is associated with a value z_i of the interest and an unknown value z_u at the point u can be estimated as a weighted-linear combination of the known values in its neighborhood $V(u) = \{1, ..., M_u\} = Q$:

$$\hat{z_u} = \sum_{i \in V(u)} w_i^u z_i, \tag{9.3}$$

where the weights w_i^u satisfy $\sum_{i \in V(n)} w_i^u = 1$.

The ordinary kriging assumes constant mean in the local neighborhood of the point and minimizes the variance of estimation error, i.e., $Var(\hat{z_u} - z_u)$. The weights can be calculated as follows to ensure the minimal variance of the estimation error.

$$\begin{pmatrix} w_1^u \\ \vdots \\ w_{M_u}^u \\ \lambda \end{pmatrix} = \begin{pmatrix} \gamma(h_{1,1}) & \dots & \gamma(h_{1,1}) & 1 \\ \vdots & \ddots & \vdots & 1 \\ \gamma(h_{M_u,1}) & \dots & \gamma(h_{M_u,M_u}) & 1 \\ 1 & \dots & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \gamma(h_{1,u}) \\ \vdots \\ \gamma(h_{M_u,u}) \\ 1 \end{pmatrix}, \tag{9.4}$$

where λ is the Lagrange multiplier to minimize the kriging error and $\gamma(h_{i,j})$ is a semivariogram that is a function of the distance $h_{i,j}$ between the points p_i and p_j. $\gamma(h_{i,j})$ represents the spatial covariance between the locations with this distance. Before the estimation of the weights and the values of the phenomenon at not sampled locations, a meaningful distance measure and semivariogram function should be selected. This is discussed in Section 9.3.

9.2.2 Data Collection Strategy

Once the set of measurement locations is selected, we search for a strategy for persistent data collection over these locations. The proposed strategy prioritizes visits to the locations where the phenomenon varies more frequently to address the temporal variability of the spatiotemporal fields. This prioritization is represented by a reward received by the UAV for visiting particular measurement location. At this point, the problem of data collection can be modeled as an instance of the OP with the following assumptions.

Particularly, for the problem of measuring the wind field, we assume (i) a constant flight altitude of the UAV that reduces the problem to 2-dimensional one; (ii) one prevailing direction of the wind entering the urban environment which allows to use one single model from the CFD solver[1]; and we also assume that (iii) the wind characteristics do not change significantly during one data collection flight; thus, it is never beneficial for the UAV to visit a location multiple times during a single flight. Since the wind influences the travel time, the flight time from p_i to p_j is not necessary the same as from p_j to p_i. The data collection flights have to respect the maximal flight times and have to start and finish at the same location (base) where the batteries are exchanged.

The problem of data collection for a single UAV flight can be then stated as directed OP [194] on a graph $G = (V, A)$, where $V = \{v_1, ..., v_M\}$ is the vertex set corresponding to the location set Q, and $A = \{a_{1,2}, ..., a_{M-1,M}\}$ is the arc set. The start and end points coincide ($v_1 = v_M$) and are fixed. The time t_{ij} ($\neq t_{ji}$) needed to travel the arc $a_{ij} \in A$ from v_i to v_j is known for all vertices. A value of the wind field z_i can be measured once the UAV reaches v_i and performs measurements which takes time m_i. The goal of the OP is to determine a path with the flight time not exceeding the travel budget T_{max} and that visits some of the vertices to maximize the total collected rewards. The rewards, or scores S_i associated with each vertex $v_i \in V$ are assumed to be entirely additive. Each vertex can be visited at most once during a single flight.

Using the introduced notation, the OP can be formulated as an integer problem with decision variables $x_{ij} = 1$ if a visit of v_i is followed by a visit of v_j, and u_i being the position of v_i in the

[1] Aggregation of multiple wind models is discussed in Section 9.5.

final path. Formally, the problem can be defined as follows.

$$\max \sum_{i=2}^{M-1} \sum_{j=2}^{M} S_i x_{ij} \tag{9.5}$$

$$\text{s.t.} \tag{9.6}$$

$$\sum_{j=2}^{M} x_{1j} = \sum_{i=2}^{M} x_{i1} = 1 \tag{9.7}$$

$$\sum_{i=1}^{M-1} x_{ik} = \sum_{j=2}^{M} x_{kj} \leq 1; \forall k = 2, ..., M-1 \tag{9.8}$$

$$\sum_{i=1}^{M-1} \sum_{j=2}^{M} (t_{ij} + m_i) x_{ij} \leq T_{max} \tag{9.9}$$

$$2 \leq u_i \leq M; \forall i = 2, ..., M \tag{9.10}$$

$$u_i - u_j + 1 \leq (M-1)(1 - x_{ij}); \forall i, j = 2, ..., M \tag{9.11}$$

$$x_{ij} \in \{0, 1\}; \forall i, j = 1, ..., M \tag{9.12}$$

In the stated problem formulation, (9.5) is the objective to be maximized. The constraint (9.7) ensures that the path starts and ends at the vertex v_1, which represents the base. The constraint (9.8) guarantees that every vertex is visited at most once and the path is connected. The constraint (9.9) deals with the limited time budget, where apart from the travel time, also the measurement time m_i at the vertex v_i is considered. Finally, the constraints (9.10) and (9.11) are necessary to prevent subtours.

Persistent and repeated measurements target the temporal variability of the wind field. In the data collection task, the fact that some locations were not visited in the latest flight need to be taken into account. This is modeled as the *Repeated OP* (ROP) that minimizes the weighted age of the measured information. In ROP, the reward received for visiting a vertex is updated after each iteration of the solver, i.e., for each new flight, see (9.15). Several strategies for reward update together with the comparison of these strategies are discussed in Section 9.3.

9.3 Proposed Method

An existing CFD solver can be utilized[2] to get prior knowledge of the environment. For simplicity and w.l.o.g., a homogeneous wind stream entering the environment is used to get an approximate model of the wind field behavior inside the urban area of interest. The CFD model is then used throughout the whole solution method. The schematic description of the proposed method is shown in Figure 9.3. First, it uses the model from the kriging analysis to determine the set of measurement locations with the lowest predictive error through a series of iteration with

[2]In our case, we use the Autodesk CFD `https://www.autodesk.com/products/cfd`.

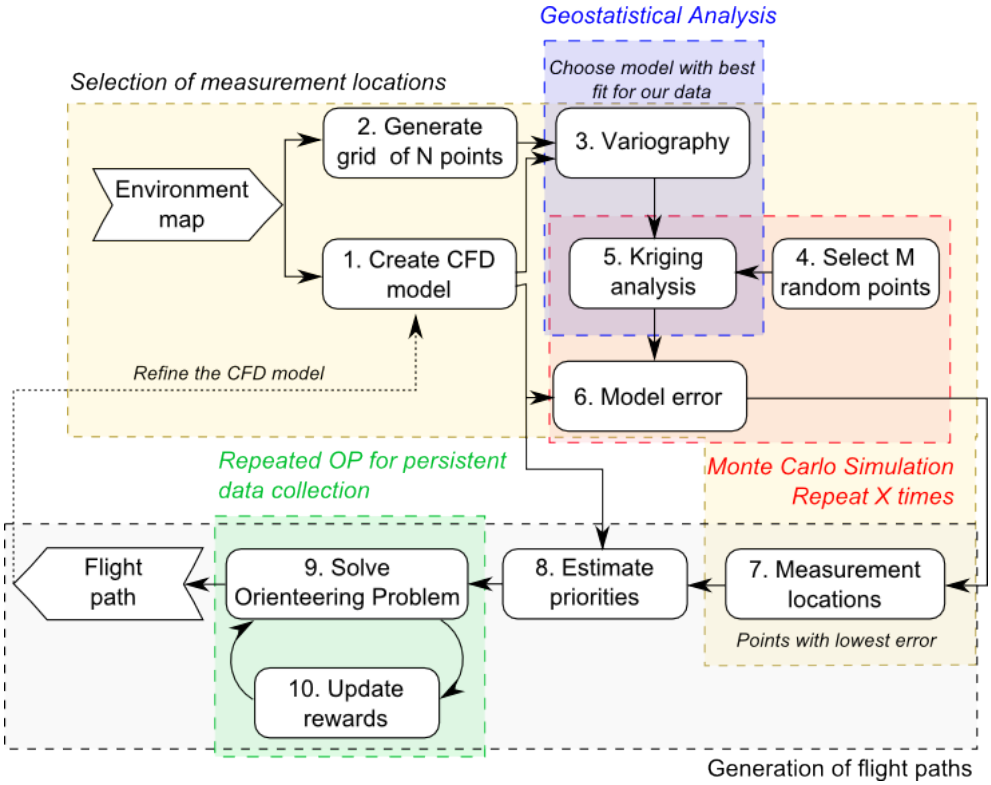


Figure 9.3: Scheme describing steps of the proposed wind data collection procedure.

randomly selected subsets of the grid points. Later, flight paths over these measurement locations are repeatedly generated by solving ROP.

9.3.1 Monte-Carlo Simulation for Selection of Measurement Locations

A meaningful distance measure and semivariogram function need be selected to calculate the kriging weights in (9.4) and estimate the $\hat{z_u}$ values in (9.3). For simplicity, we assume the air particles in the environment spread around obstacles and the distance between two locations is the length of the shortest path between these locations avoiding obstacles.

Although there is an infinite number of possible semivariogram functions, most commonly used semivariogram models, such as linear, exponential, and spherical models, provide good results for most data sets. We estimated the semivariogram model by fitting a curve to the data from the CFD solver, see Figure 9.4. Given the data, an exponential semivariogram is used with parameters chosen following the general guidelines for a semivariogram selection [16]. The selected exponential semivariogram can be expressed as

$$\gamma(h_{i,j}) = \begin{cases} 0 & h_{i,j} = 0 \\ C_0 + (C_1 - C_0)(1 - \exp(-3h_{i,j}/R)) & h_{i,j} > 0 \end{cases}, \tag{9.13}$$

where C_0 is the nugget effect, C_1 is the sill parameter, and R is the range parameter.

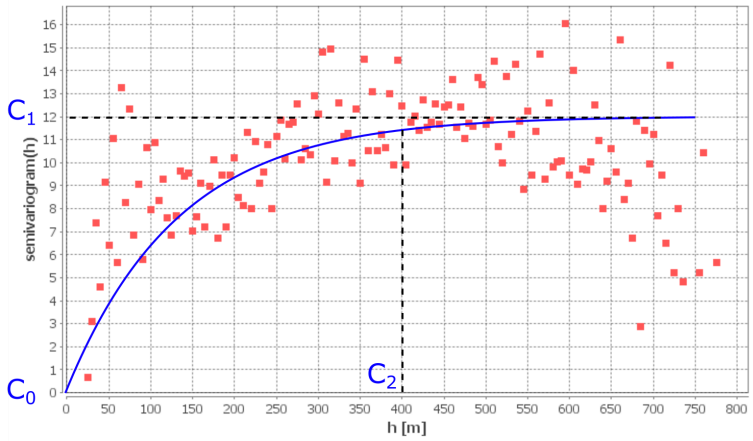


Figure 9.4: Empirical semivariogram with fitted model.

With the known semivariogram and CFD models, (9.4) and (9.3) can be evaluated. Then, the task is to select a set of $Q \subset P$ locations from the grid samples P that would produce the minimal predictive error after the kriging interpolation. The subset Q is determined using Monte-Carlo analysis. Unlike the method mentioned in [31], the knowledge of the CFD model is utilized to provide a better estimation of the predictive error. A random set of locations is selected, and the kriging analysis is used to estimate the wind field values at the remaining $P \setminus Q$ locations. These values are then compared with the values from the CFD model to get the predictive error of the model acquired from the set Q according to (9.14).

$$E_r = \frac{1}{\|P\| - \|Q\|} \sum_{i}^{P \setminus Q} |\hat{z}_u(i) - z_u(i)| \tag{9.14}$$

Different datasets (CFD models from various simulation times) are used to prevent overfitting of the kriging model and to estimate the kriging weights and the predictive error. The process is repeated several times for random subsets of P and Q^* with the smallest predictive error selected for the planning of data collection strategies. See the results of the Monte-Carlo analysis for the experimental environment in Figure 9.8.

9.3.2 Repeated Orienteering Problem

Each measurement location $q_i = (x, y)$, $q_i \in Q$ and its corresponding vertex v_i in the OP formulation are assigned a weight, the reward corresponding to the temporal variability of the phenomenon at the location q_i. The reward is set following the idea that more changes of the phenomenon at the given location, the more frequent measurements are required, and thus a higher reward is assigned to the corresponding vertex. Then, the problem of the persistent data collection is formulated and solved as an instance of ROP.

As the UAV visits only a portion of the measurement locations within a single flight, the momentary importance of these locations, and thus the reward for visiting them in the next

flight S^{vis} drops and at the same time, the momentary importance of the unvisited locations S^{non} increases.

Solving ROP is not utilized to determine a single solution to the problem. Instead of that, it is used to generate a series of sub-solutions for the steps $k = 1, 2, ...$, where for each $(k+1)$-th step, the problem is similar to the problem for the k-th step and only the rewards for visiting the particular vertices change depending on the sub-solution of the problem at the k-th step. The principle of the reward changes can be described by

$$S_i(k+1) = \begin{cases} S_i^{vis} & \text{vertex } v_i \text{ visited in the } k\text{-th step} \\ S_i^{non} & \text{vertex } v_i \text{ not visited in the } k\text{-th step} \end{cases}. \tag{9.15}$$

As the solution of ROP is time-step series, its evaluation needs to be based not on a single value but on behavior characteristics, such as the time needed to cover all the vertices at least once, periodicity, min/max values of the sub-solutions, the average weighted information age, etc. Based on the adopted reward update strategy, various behaviors of the ROP solutions can be observed. These behaviors also depend on the travel budget of the UAV and its relation to the size of the observed area.

Two types of the scenario can be distinguished where the ROP solutions differ significantly. In the first type, *large budget (LB) scenario*, all the locations can be visited in up to two iterations. In the second type, *small budget (SB) scenario*, the UAV need more iterations to visit all the locations. Evaluation of these strategies for score restarts and updates which can be either constant or weight-related is presented in the next part of this section.

9.3.3 Reward Update Strategies

Without a loss of generality, R weight classes of the locations are assumed $r \in \{1, ..., R\}$ where r_{min} is the lowest of them. Thus, each vertex has its weight that is constant, and the reward for its visit is updated according to the previous ROP solutions. The weight ratios correspond to the demanded ratios of the vertex visiting frequencies.

A strategy for ROP consists of two sub-strategies: (i) a strategy for a restart of the visited vertex score, S_i^{vis}; (ii) and a strategy for the update of the score of the non-visited vertex, S_i^{non}. These strategies are analyzed starting with the reward restart options. Three operations are reasoned about: (i) the constant operation; (ii) the operation with the original weight of the vertex; (iii) and the operation with a fragment of the original weight of the vertex.

- *Restart to a Constant, RtC* ($S_i^{vis} = C$) – This strategy is not suitable for the *LB scenarios* since after the first flight, all the visited nodes acquire the same score. In the *LB scenario*, some of them may be visited again on the subsequent flight, but the information about the nodes priorities would be lost. This behavior is demonstrated in Figure 9.5.

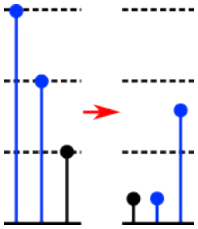


Figure 9.5: Scheme of the information loss problem in the case of *RtC* strategy in the *LB scenario*. Blue nodes are visited in the current iteration, the higher the node, the higher is its momentary score. A node with originally low weight can be selected over a more prioritized one in the later iterations.

- *Restart to the original Weight, RtW* ($S_i^{vis} = r_i$) – This strategy restarts the score values of the visited nodes to their original weights. Unlike the previous case, it preserves the priorities of the nodes over the subsequent iterations for *LB scenarios*. However, it is necessary to select the reward update strategy so that the low priority nodes are visited with respective frequency. The following reward update strategies can be used

  - *Update by Constant, UbC* ($S_i^{non} += C$) – If $C = r_{min} = 1$, the lowest weighted nodes are considered no sooner that after R ROP steps (R being the number of weight classes) when their reward starts to be close to the reward of just visited high weighted nodes. This would ensure that the locations with the weight R are on average visited R-times more often that those with the weights $r_{min} = 1$. Thus, this strategy generates solutions with the vertex visiting frequency ratios corresponding to the ratios of the weights of the vertices. Other values of C change the vertex visiting frequency ratios, i.e., setting $C = 2r_{min} = 2$ causes that the lowest weighted nodes are competitive with the highest weighted ones in only $R/2$ steps. Thus, these low weighted nodes are visited approximately two-times more often than required. Similarly, setting $C = r_{min}/2 = 0.5$ causes visiting the lowest weighted nodes approximately two-times less often than required.

  - *Update by Weight, UbW* ($S_i^{non} += Cr_i$) – This combination of the restart and update strategies can significantly delay the first visits of the lower weighted nodes in *LB scenarios* since the score disadvantage of these vertices is taken into account two times. However, the vertex visiting frequency ratio can be approximately correct as in the previous case, see Figure 9.6.

- *Restart to the original weight fragment, RtWF* ($S_i^{vis} = r_i/R$). This strategy restarts the score values of the visited nodes to a fragment of their original weights. It should preserve the priorities of the nodes over multiple iterations, and theoretically, it may decrease the time necessary for the first full coverage of all the nodes. The reason is that after the first visit, even the nodes with the highest weights acquire similar score values to those of the

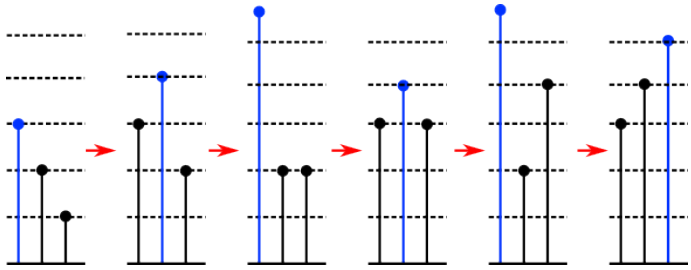


Figure 9.6: Scheme of the delayed visit in the case of *RtW–UbW* strategy in the *SB scenario*. The third node is visited only after multiple visits of the other nodes.

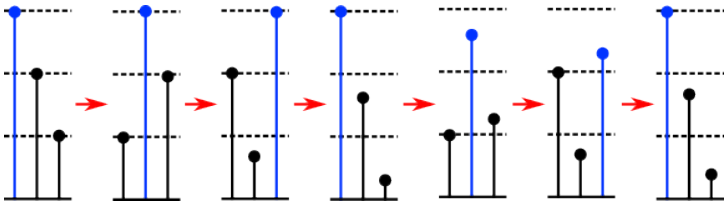


Figure 9.7: Scheme of the problem of non significant weights in the case of *RtWF–UbC* strategy. All nodes are selected with the same frequency.

least weighted nodes. The following reward update strategies can be used

- *Update by Constant, UbC* ($S_i^{non} += C$) – This strategy, however, renders all the priorities insignificant, see Figure 9.7.
- *Update by a Weight Fragment, UbWF* ($S_i^{non} += Cr_i/R$) – This is, in fact, a scaled version of the *RtW–UbW* strategy.
- *Update by the original Weight, UbW* ($S_i^{non} += Cr_i$) – In this case, the updates would be much more significant than the restarts.

9.4 Experimental Evaluation

The proposed data collection planning method has been verified in an artificial urban environment and using the CFD solver to model the wind field produced by a 10 m.s^{-1} south wind entering the area, see Figure 9.2. The model is covered by a sampling grid, from which only a subset of $Q = 30$ locations is selected as a trade-off between dense sampling and suitability for their fast coverage by a small number of UAV flights. The methods of geostatistics and Monte-Carlo analysis have been applied to lower the model predictive error according to the methodology described in Section 9.3 to select locations where measurements can be taken. Figure 9.8 shows the development of the predictive error E_r during the Monte-Carlo iterations and Figure 9.9 shows the best found set of the selected measurement locations.

The selected locations have been associated with the priority weights to deal with the temporal variability of the measured wind field. These weights reflected the required ratio of the

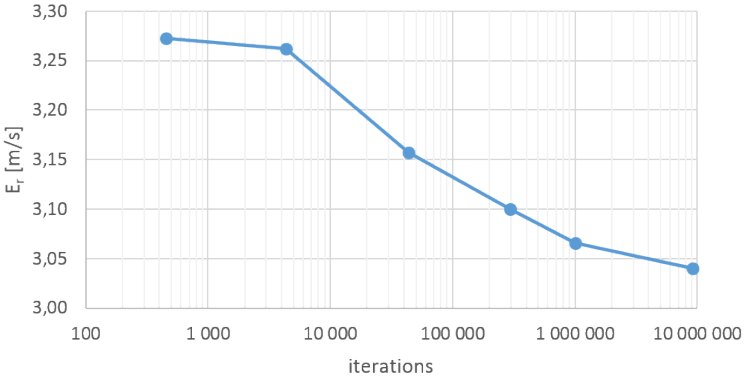


Figure 9.8: Improvement of the predictive error using Monte-Carlo analysis.

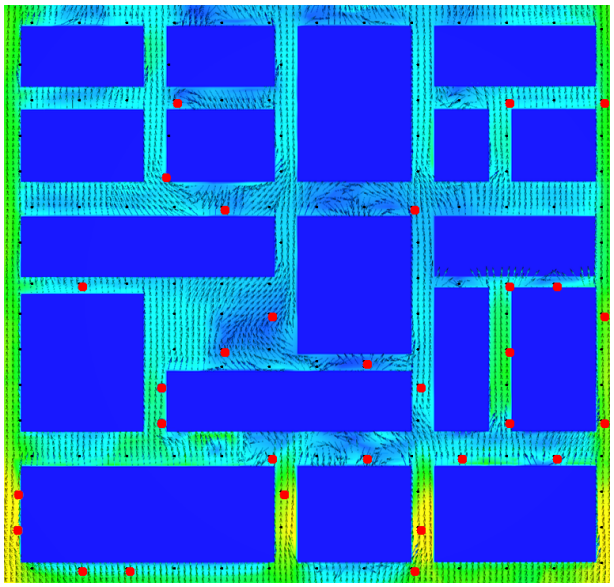


Figure 9.9: Set of selected measurement locations with the lowest predictive error E_r.

vertex visiting frequencies and would have been assigned based on the wind field variability at the particular locations acquired from the CFD model. Figure 9.10 shows the wind variability in the experimental environment estimated from the CFD model. However, the weights of these $Q = 30$ points have been assigned artificially to have a reasonable distribution of the weights among the measurement locations for experimental scenarios. The weights are divided into $R = 4$ weight classes according to the following distribution: 3 points with the weight 4, 6 points with the weight 3, 9 points with the weight 2, and 12 points with the weight 1.

The introduced ROP formulation has been used to repeatedly search for the best paths for the persistent wind data collection. Various score restart and update strategies, according to the discussion in Section 9.3.2, have been evaluated. In the following, we present the most significant ones, for better clarity also summarized in Table 9.1.

M1 *Restart* the reward of the visited vertices to zero, *update* the reward of non-visited nodes by the original weight, i.e., $S_i^{vis} = 0$, $S_i^{non} += r_i$.

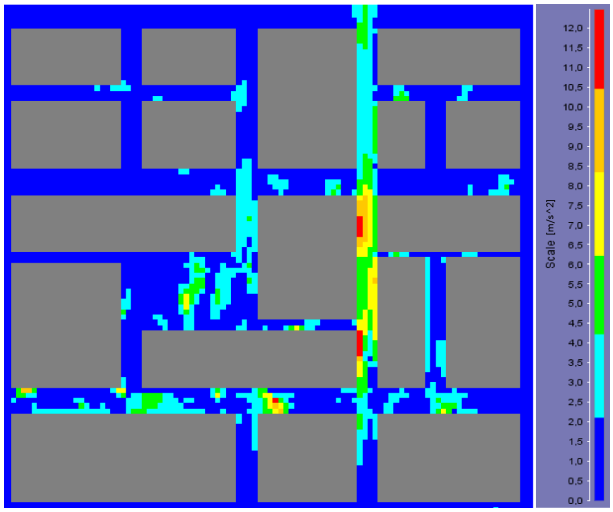


Figure 9.10: Visualization of the wind variability in the experimental environment estimated from the CFD model.

M2 *Restart* to the original weight, *update* by double the minimal weight, i.e., $S_i^{vis} = r_i$, $S_i^{non} \mathrel{+}= 2r_{min}$.

M3 *Restart* to the original weight, *update* by the original weight, i.e., $S_i^{vis} = r_i$, $S_i^{non} \mathrel{+}= r_i$.

M4 *Restart* to a fragment of the original weight, *update* by a fragment of the original weight, i.e. $S_i^{vis} = r_i/R$, $S_i^{non} \mathrel{+}= r_i/R$.

M5 *Restart* to a fragment of the original weight, *update* by the minimal weight, i.e., $S_i^{vis} = r_i/R$, $S_i^{non} \mathrel{+}= r_{min}$.

Table 9.1: Used score restart and update strategies in the presented methods

	Restart			**Update**		
	RtC	**RtW**	**RtWF**	**UbC**	**UbW**	**UbWF**
M1	✓			✓		
M2		✓		✓		
M3		✓			✓	
M4			✓			✓
M5			✓	✓		

The utilized UAV is assumed to be VTOL aircraft with the constant airspeed of 10 m.s^{-1}, and each iteration of ROP has been solved optimally using the IBM ILOG CPLEX solver version 12.5.1. Considering the effects of the wind on the flight produced non-symmetric trajectories, where a path from the vertex v_i to the vertex v_j may not be the same as from v_j to v_i. In the *LB scenario*, the maximal flight time of the UAV is set to $T_{max} = 1800$ s, and in the case of the *SB scenario*, the flight time is $T_{max} = 900$ s. The time required for the wind measurement at each location is set to $m_i = 60$ s.

Results of 50 iteration steps for all the update strategies for the *LB* and the *SB scenarios* are shown in Figure 9.11 and Figure 9.12, respectively. The observed characteristics are (i) the age of the measurements weighted by the classes weights, i.e., the weighted information age wIA; (ii) the average weighted information age $\overline{wIA}$; (iii) average information ages $\overline{IA}$ of individual weight classes; and (iv) the number of visits of individual weight classes over the duration of the experiment. Moreover, the following plots show the time required for each method for the first full coverage of all the measurement locations—the first time step, when wIA is non-zero. In this case, the method that allows the fastest full coverage is the M5. On the other hand, the slowest coverage time is provided by the method M4.

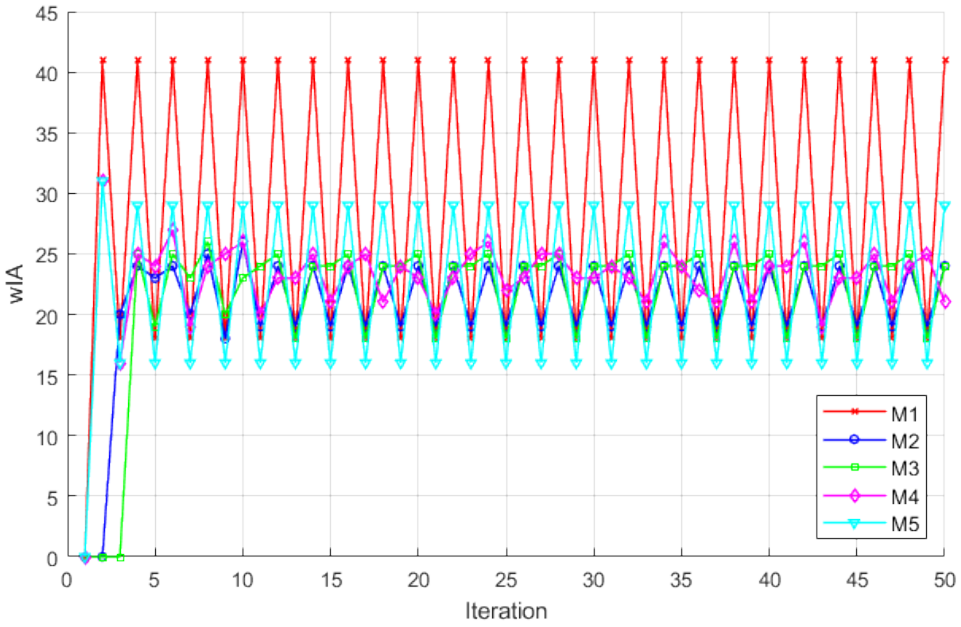


Figure 9.11: Evolution of the weighted information age wIA for the *LB scenario*.

The lowest average of the weighted information age $\overline{wIA}$ in both the *SB* and the *LB scenarios* is provided by the method M2, see Figure 9.13. On the other hand, the highest average of the weighted information age is produced by the method M1 in the *LB scenario* and by the method M5 in the *SB scenario*.

The best ratio of the average information ages for the individual weight classes that correspond with the ratio of the weight classes, is given by the method M4, see Figure 9.14).

The best ratio of the vertex visiting frequencies for the individual weight classes that correspond with the ratio of the weight classes, is given again by the method M4, see Figure 9.15.

9.5 Remarks

The proposed method can be used to refine the environmental model of MR simulations and increase their fidelity. Even though it is applied to the measurement of the wind field, we believe

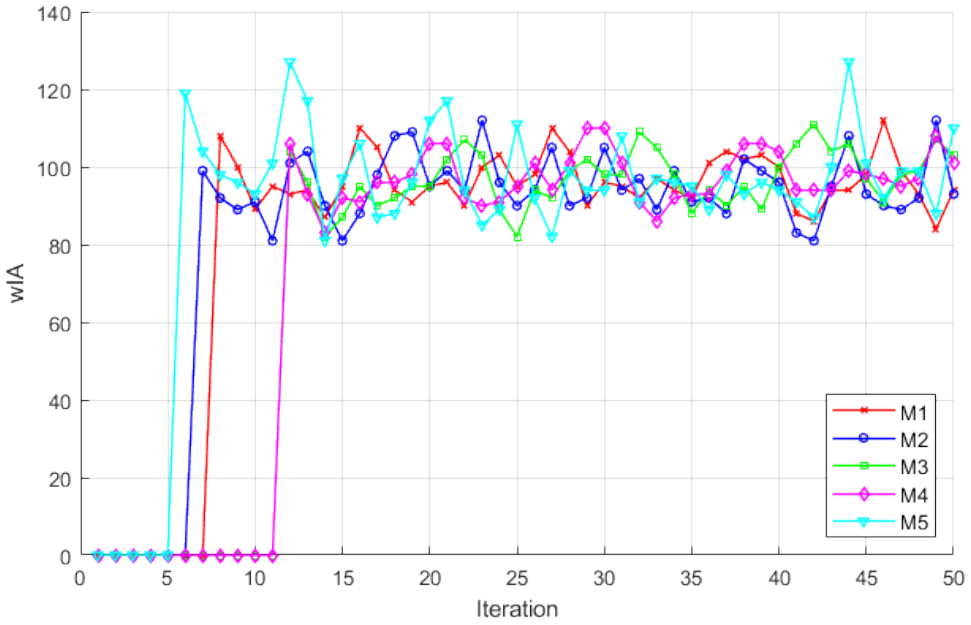
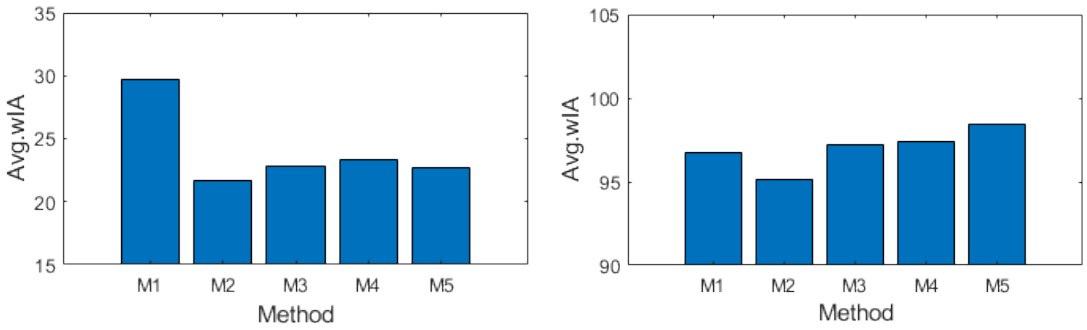


Figure 9.12: Evolution of the weighted information age wIA for the *SB scenario*.

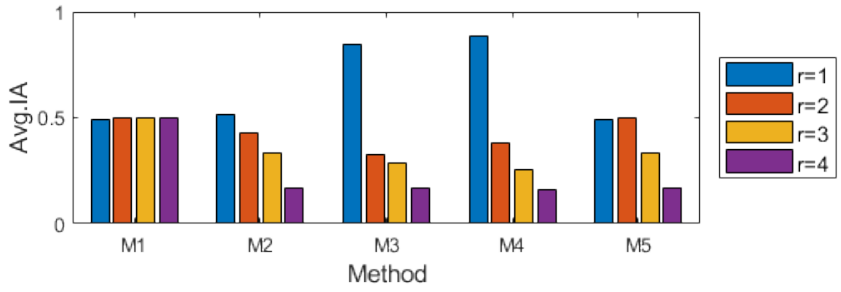


(a) LB scenario (b) SB scenario

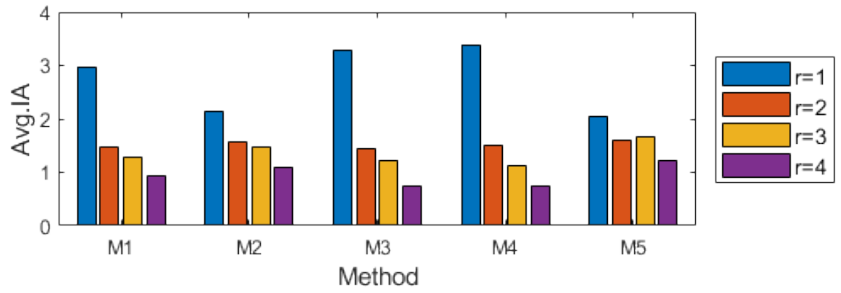
Figure 9.13: Average weighted information age $\overline{wIA}$ for various update methods in (a) *LB* and (b) *SB scenario*.

it can be generalized for data collection in arbitrary spatiotemporal fields.

For the future work, it would be interesting to consider different directions of the wind entering the urban area under the investigation and aggregate multiple wind models. An inspiration can be taken from the work of Du et al. [52] who came with a sensor placement scheme to measure the wind distribution with a limited number of sensors in environments where the target phenomena exhibited strong non-Gaussian yearly distribution. The results of all-weather seasons were combined to calculate the optimal sensor placement scheme for the whole year. The combination of sensors is necessarily sub-optimal since only static sensors are used. Employing unmanned aircraft and the herein proposed approach can solve this issue.

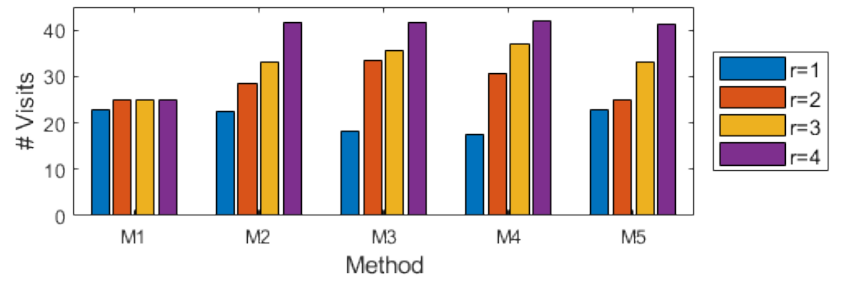


(a) LB scenario

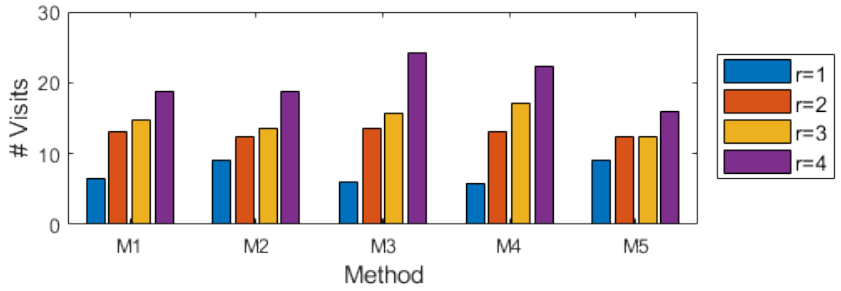


(b) SB scenario

Figure 9.14: Average information ages $\overline{IA}$ of the priority classes for various update methods in (a) *LB* and (b) *SB scenario*. The class with the lowest priority is in blue while the class with the highest priority is in purple.



(a) LB scenario



(b) SB scenario

Figure 9.15: Number of visits of the priority classes for various update methods in (a) *LB* and (b) *SB scenario*. The class with the lowest priority is in blue while the class with the highest priority is in purple.

Chapter 10

Specific Case Studies

In this chapter, we provide an overview of the specific use cases in which we employed the proposed MiRAMuR architecture for incremental development of multi-UAV systems with the use of MR testbeds. The architecture has been successfully employed during development of different aviation applications within research projects funded by Czech Ministry of Defence, U.S. Air Force Research Laboratory, and U.S. Army CERDEC, and also utilized for the development of an assistant system for optionally piloted light-sport aircraft pilots within a TAČR funded project. This multitude of applications shows the versatility of the presented method for research and development in UAS domain. An overview of the real aerial vehicles for which the method has been utilized is shown in Figure 10.1 and the use cases are as follows.

Figure 10.1: Aerial vehicles used in practical deployment of the proposed method.

The first presented use case is an unmanned system that provides C2 capabilities over a heterogeneous team of autonomous unmanned aircraft [165]–[167] and is in detail presented in Section 10.1. The second use case, presented in Section 10.2, is a MR testbed addressing the

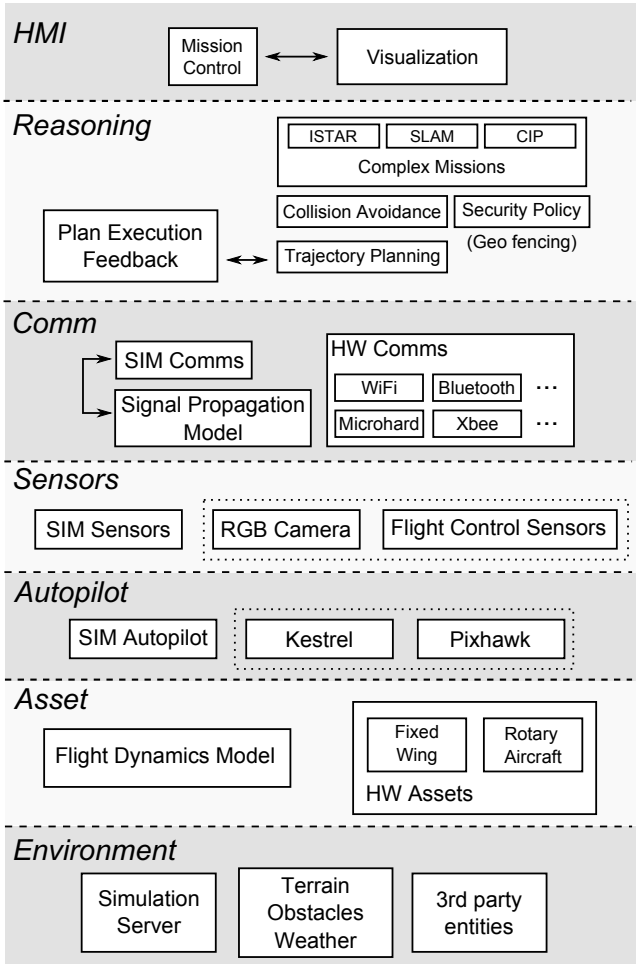


Figure 10.2: Architecture of the system for command and control of autonomous UAVs.

incremental deployment of the C2 system on Lockheed Martin Desert Hawk III aircraft, and training the system operators. Finally, the third use case is a system for providing increased flight safety in a domain of light-sport aircraft based on recommendation of the best collision-free trajectories for pilot or autopilot of the real light-sport aircraft. This system is presented in Section 10.3.

10.1 Heterogeneous Team of Autonomous UAVs

The MiRAMuR architecture presented in Chapter 4 has been used for development of unmanned system consisting of heterogeneous team of autonomous UAVs capable of performing complex tactical missions such as team area surveillance, target tracking or critical infrastructure protection and performing dynamic mission reconfiguration in case of change of the mission or available resources. Individual aspects of the system framework are presented in Chapter 5 and have been already published [165]–[167]. Individual layers of the MiRAMuR architecture and their inner content are shown in Figure 10.2.

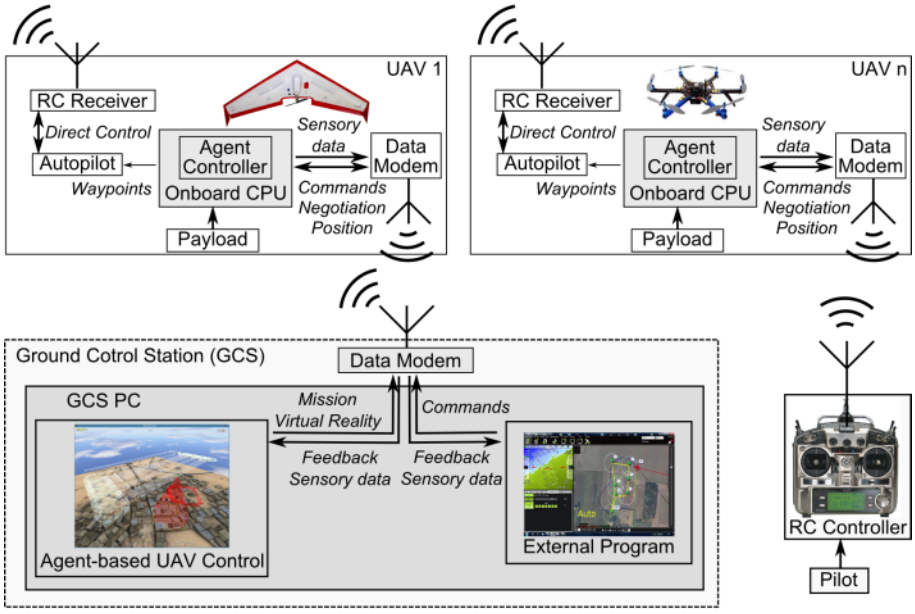


Figure 10.3: Scheme of multi-UAV system deployment.

We have performed series of experiments with fixed wing and VTOL rotary UAVs to validate the architecture. These experiments started from the pure simulation of all assets and environment. We run indoor HIL simulations with real modems and autopilots with simulated sensory inputs and flight dynamics. We performed field experiments in MR simulation with one hardware UAV and several virtual ones. Finally, we experimented with multiple fully deployed hardware assets, following the incremental development principle described in Section 4.1. This principle allowed less flight test hours for one and mainly two aircraft flying simultaneously. Also, it was advantageous because of the reduced deployment cost as the tests were executed once they were necessary for the verification of the particular virtualization levels.

10.1.1 System Architecture

The system was deployed on real hardware UAVs according to the scheme shown in Figure 10.3. It is divided into two parts: (i) on-board agent control and reasoning; and (ii) GCS with mission control, simulation, and visualization. The first part runs on an on-board computer that is connected to all sensory payload, directly to an autopilot to allow waypoint upload, and to the data modem for team coordination and communication with the operator. The GCS part runs all optional virtual UAVs together with the simulated environment if necessary, and is used for the mission assignment and visualization. Third party GCS software (Mission Planner, Virtual Cockpit, UgCS, etc.) can be connected for safety reasons and extended visualization options. For further improvement of the operation safety and for legal reasons the system allows to overtake control of each UAV by a pilot with the *Remote Controller* (RC).

Figure 10.4: Flight tests of multi-UAV system.

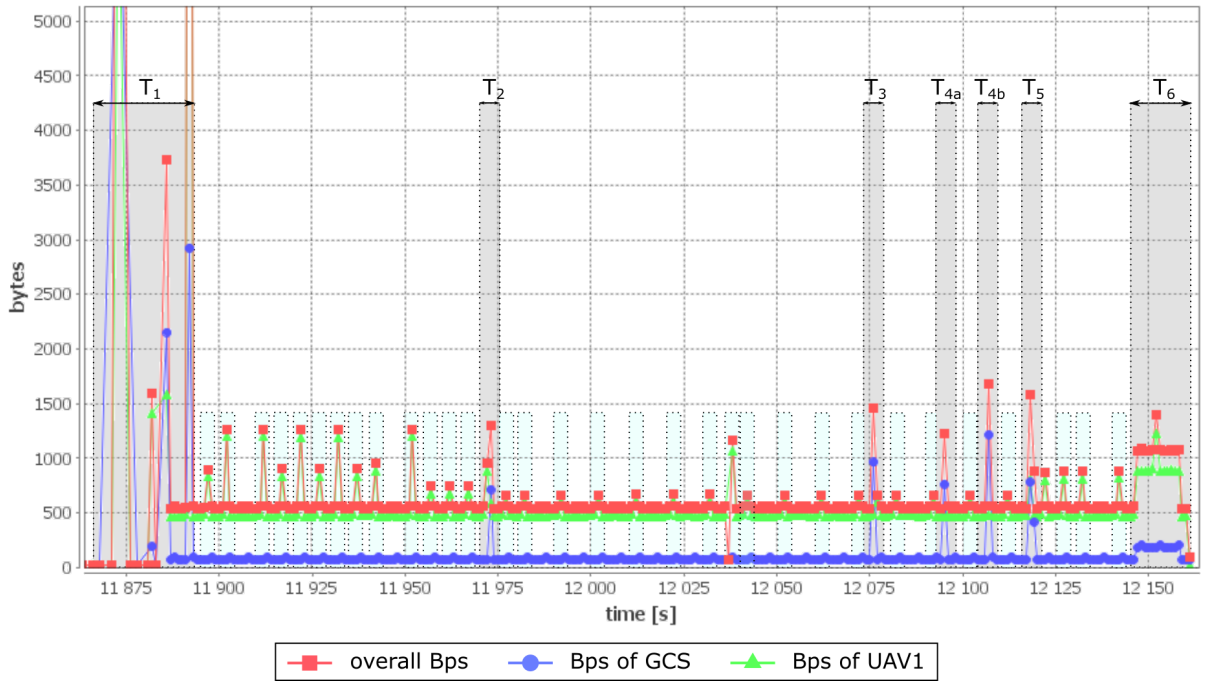
10.1.2 Identification of the Communication Requirements

One of the tests required for transition between individual development stages is validation of the communication bandwidth and delay. The bandwidth has to be wide enough to transmit plans, commands, telemetry, and other data required for a proper system behavior. Figure 10.5 depicts the required bytes in two scenarios. The transmission delay had to be small enough to provide good response time and well-timed collision avoidance. The minimal bandwidth and maximal delays were estimated in simulated scenarios with a variable number of UAVs and tasks performed.

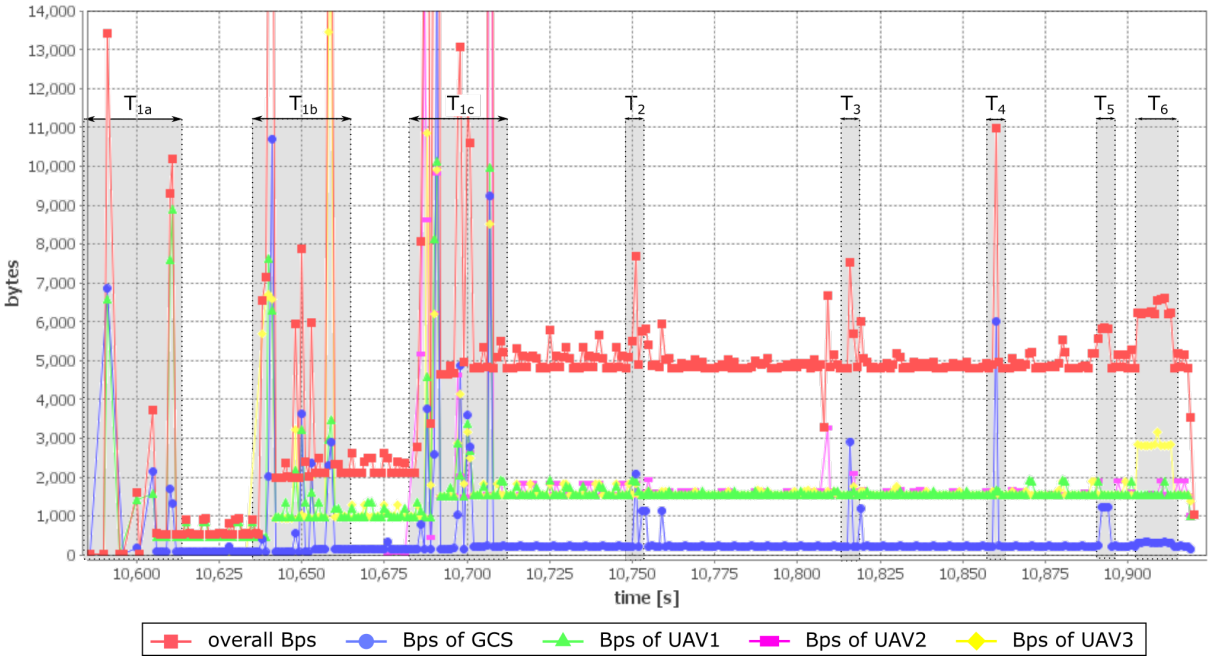
Experimental measurements of the signal strength have been performed to validate the used RF communication modems. Fixed wing aircraft with an omni-directional antenna secured on its left wing has been scheduled to fly square patterns at various distances from the GCS, see Figure 10.6.

During the first experiments, a basic XBee-PRO DigiMesh 2.4 RF modules have been used as the communication devices. These modules provided cheap and light-weight communication solution; however, since they allowed only half-duplex serial connection, even in cases of strong RSSI (*Received Signal Strength Indication*), the available communication bandwidth is less than approximately 5 kBps, which is not sufficient for the system needs. Therefore, the wireless modules have been replaced by the Microhard nVIP2400 broadband modems that apart from hypothetical data-rates of 54 Mbps provided also features of TCP/IP protocol and communication encryption.

The communication delay has an effect in two situations: (i) the delay between the issuance of a command and the start of the command execution; and (ii) the time from the moment a high-priority UAV successfully broadcasts its plan to the moment another low-priority UAV with colliding trajectory starts execution of its *Collision Avoidance* (CA) maneuvers. Variable throughput and delays have been caused by different channel properties and noise characteristics



(a) One UAV in the system



(b) Three UAVs in the system

Figure 10.5: Data exchanged between UAVs and GCS during various mission assignments. Parts T_1 represent data exchanged during initialization stage—registration of UAVs, time synchronization, and team establishment. T_2 represents assignment of surveillance mission and corresponding answers; T_3 represents setting 5 waypoints to each of the UAVs; T_4 represents assignment of 3 NFZs, and T_5 represents their removal. In T_6, UAV3 informs about a located ground target. The background data transmission consist of periodic advertisements, radar simulation, and UAV plan broadcasts and are marked by cyan background.

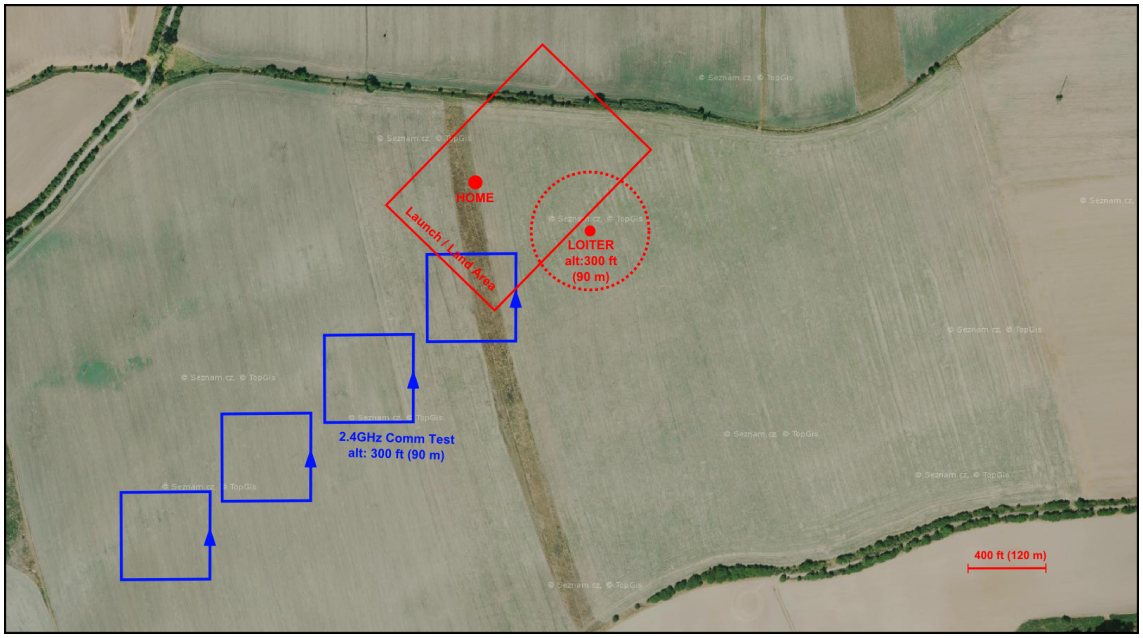


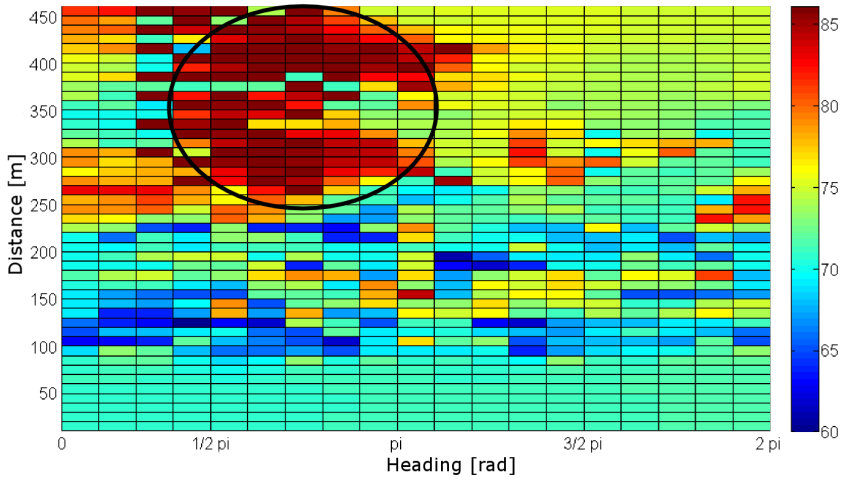
Figure 10.6: Setup for experimental measurement of the RF signal strength.

in every development stage. Together with the requirements on the bandwidth, and our available financial budget, these data have been used to select the most suitable communication hardware. Cheap XBee modules shown to be unsatisfactory because of their limited range and bandwidth, and therefore, Microhard nVIP2400 RF modems have been chosen. Their throughputs and latencies are shown in Table 10.1.

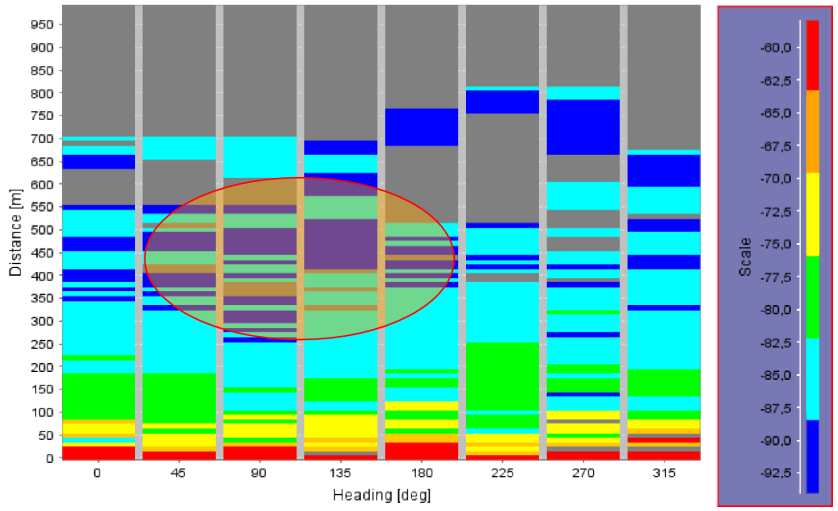
Table 10.1: Data-link quality in different development stages.

System	**Data-link spec**	**Throughput** [Mbps]	**Execution delay** [ms]	**CA delay** [s]
Sim	localhost TCP/IP sockets	650.0	10	0
HIL	RF modems in lab	13.0	300	2
HW	RF modems in the field	0.5	1500	8

During these experiments, an anomaly in the signal propagation appears. The signal quality is locally degraded in situations when the antenna points away from the GCS and the aircraft's body and motor is in the way (see the highlighted areas in Figure 10.7). This phenomenon has been observed by Meiser [119] and Mahdi et al. [11], but has not been studied in more detail yet. Very probably this RF signal hindering can be caused by aircraft's body and motor jamming. None of the existing wireless network simulators and no RF signal propagation models deal with this phenomenon, and thus the real-world experiments have been necessary to discover this



(a) XBee-PRO DigiMesh 2.4 RF modem



(b) Microhard nVIP2400 modem

Figure 10.7: Results of RSSI [dBm] measurements. The distance parameter is the distance between the aircraft and GCS and the heading parameter represents the relative heading attitude of the aircraft to the GCS. The highlighted area corresponds to the configuration where the UAV's antenna has been positioned away from the GCS. XBee-PRO measurements were adopted from [119].

particular issue. Based on the experimental evaluation, this phenomena can be integrated into the system simulation to significantly increase its fidelity.

10.1.3 Coordinated Flight Experiments

A set of field tests was performed to verify the features of the proposed C2 framework. Missions carried out included trajectory planning in dynamic environment, collision avoidance, surveillance, tracking and combined operations. The collision avoidance experiment illustrates benefits of proposed architecture for design and development of robust algorithms. The experiment consisted of three phases: (i) experiments with virtual UAVs; (ii) experiments in MR simulation; and (iii) experiments with two hardware UAVs.

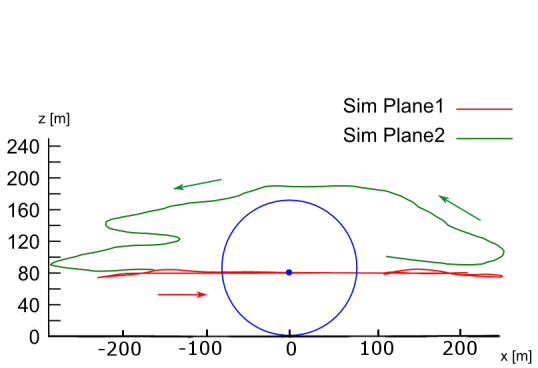
The trajectories from the the first phase experiments are depicted in Figure 10.8. In the second phase, the same collision avoidance algorithm was deployed on a physical UAV. In this case, the UAVs violated their predefined safety zones during the CA maneuver. During later analysis, a combination of several factors was identified as the cause of this behavior. The main reasons were: (i) environmental conditions—tailwind affected the flight of the physical UAV when changing its flight level; (ii) bounded computational resources of the onboard computer; (iii) a delay in communication; and (iv) an imprecise flight dynamics model used in the simulation. Flight trajectories from the second stage testing are shown in Figure 10.9.

We made necessary modifications in the control architecture and increased the UAVs' safety zones and proceeded to a third phase when the algorithms were deployed on two hardware assets. Subsequently the CA algorithm was successfully validated in the real-world scenario as shown in Figure 10.10.

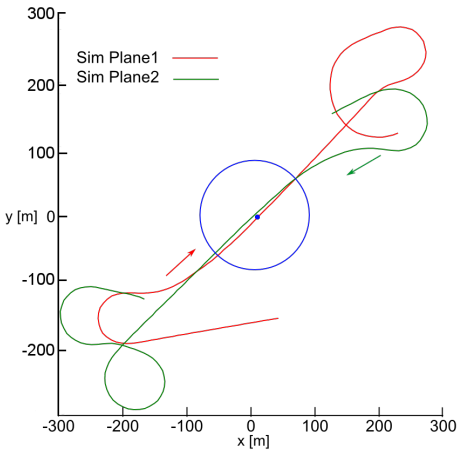
Moreover, the mixed-mode architecture proved to be useful especially during tests of dynamic mission reconfiguration when some of the existing assets were being removed from the team, e.g. due to landing or malfunction, or when new ones were introduced, as when new replacement asset is launched or an asset joins the team when its own sub-mission is finished. Because launch and landing of an asset is the most complicated part of the experiments, assets that were removed or introduced during the mission were simulated, while assets representing those being present during whole mission were the real ones. This approach helped us to carry out those experiments without unnecessary overhead.

10.2 System for Verification of Collision Avoidance Methods

The MiRAMuR architecture presented in Chapter 4 has been also utilized in a joint project with Quanterion Solutions Inc., U.S. Air Force Research Laboratory (AFRL), and NUAIR alliance. In this case, the aim has been to develop a robotic testing platform for verification of collision avoidance methods that would enable C2 of multiple fixed wing airplanes Desert Hawk III. The emphasis of the project has not been only on the flight safety in shared airspace and verification



(a) Side view



(b) Top view

Figure 10.8: Flight trajectories of 2 virtual UAVs. Planes heading toward each other used cooperative priority based collision avoidance algorithm [196] to prevent the collision. The safety zone around the high priority UAV Sim Plane1, should be avoided at all times. The blue circle marks the safety zone at the time when both UAVs are closest to each other.

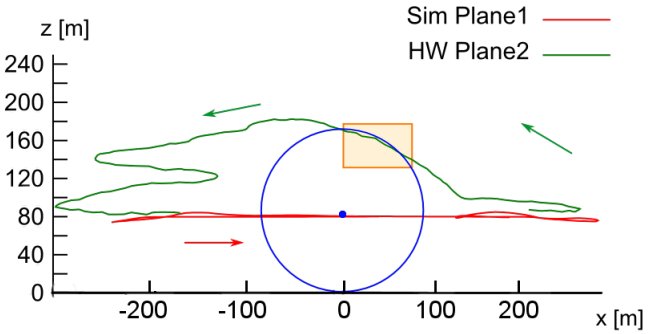


Figure 10.9: Side view of the flights trajectories of a virtual UAV and HW UAV in MR simulation. Planes heading toward each other used cooperative priority based collision avoidance algorithm [196] to prevent the collision. The orange rectangle emphasize the moment when the blue safety-zone around Sim Plane1 is violated by HW Plane2.

of cooperative collision avoidance among aircraft, but also on the training of operators of the final system.

The system is similar to the one presented in Section 10.1, and thus the individual architecture layers are similar to those in Figure 10.2. The increased safety requirements lead to an addition of extra RC and GCS to every single UAV which would be used in emergency situations. Further, a common GCS serving for multi-agent C2 has been provided. Also, a new autopilot control protocol—AFRL's *Common Communications Client for Payload Operations* (C3PO)—that provided increased level of control safety had to be implemented and tested. See Figure 10.11 for the overall scheme of the deployed system.

The system has not been directly deployed and operated by us, but by the members of the

Figure 10.10: Collision avoidance experiment with two hardware UAVs in the third phase of hardware deployment.

Quanterion Solutions and NUAIR alliance with our remote assistance only. Thanks to the system modularity and existing implementations of system configurations of various virtualization levels, the operators could have been trained iteratively. They started with training with a pure software simulation, then using various degrees of mixed reality setting, and finally training with the real Desert Hawk III airplanes. The fast and successful training of the operators refers to simplicity and portability of the resulting system. Moreover, it also supports the suitability of the incremental development method even for the training activities. An example of the Desert Hawk III launch and field test setting is shown in Figure 10.12.

During this project, there has been an extensive HIL testing phase. Not only for reasons of development and verification but also for system operators to get familiar with the system safely. These operators then provided feedback that resulted in changes in the HMI layer and integration of third party software for monitoring and visualization (Virtual Cockpit[1] software). The modularity of the system and possibility to easily change its virtualization level allowed for incremental training of the operators from pure simulation via HIL configuration to the final system deployed on the physical Desert Hawk III aircraft.

10.3 Safefly – System for Flight Safety of Light-Sport Aircraft

The last herein reported use case of the MiRAMuR architecture is the Safefly project that aimed to build a system for increasing flight safety of optionally piloted light-sport aircraft. Negotiation among aircraft, cooperative, and non-cooperative methods of collision avoidance (CA) and

[1] http://www.lockheedmartin.com/us/products/procerus/kestrel-autopilot.html

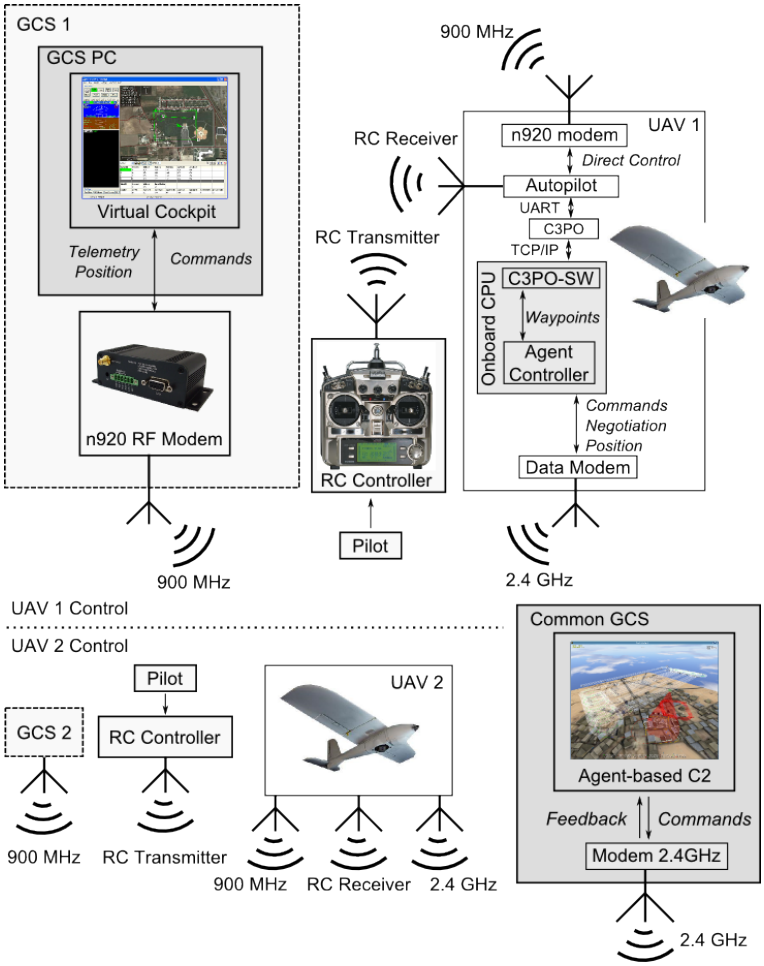


Figure 10.11: Scheme of the system deployment on Desert Hawks III.

Figure 10.12: Flight tests with the Desert Hawks III.

trajectory planning algorithms were employed to prepare and recommend the best collision-free trajectories for pilot or eventually upload them to an autopilot of light-sport aircraft. The system consists of multiple hardware parts that were selected to satisfy needs of the final system

on computational power, communication bandwidth, size, and weight.

- ***TL-1000 Integra*** – *An Electronic Flight Instrument System* (EFIS) that integrates all primary flight instruments, navigation, and 3-D terrain, together with various visualization options and autopilot options for pitch and roll control. Integra systems were integrated into two light-sport planes Tecnam P-2002 Sierra and connected with the other system parts. Apart from the two P-2002 Sierra aircraft, also Tecnam P92 Echo aircraft was used in the experiments for the comparison of different aircraft constructions.

- ***Zaon PCAS XRX*** – A passive collision avoidance system that uses amplitude and phase shift of other aircraft's transponder responses to estimate their presence and position. Since a great majority of civil aircraft is equipped with a transponder, this system provides non-cooperative[2] detection of the surrounding aircraft.

- ***Microhard n920F*** – An RF module for cooperative plan exchange and negotiation of evasion maneuvers among aircraft equipped with the Safefly system. It is also used to transmit mixed reality data and information to/from GCS.

- ***Gumstix Overo FireStorm*** – An onboard ARM computational unit that is connected to all the other system parts and runs the reasoning algorithms for planning, collision avoidance, and negotiation.

The respective layers of the system architecture and their inner content are shown in Figure 10.13. We omit the software models in the individual layers since they are similar to those in Figure 10.2.

Accordingly to the concept of the incremental development described in Chapter 4, a flight dynamics model of the Tecnam aircraft has been implemented based on their BADA performance characteristics [137]. The algorithms for the trajectory planning [199] and cooperative negotiation-based collision avoidance [197] have been first tested in a pure virtual simulation, see Figure 10.14. Furthermore, from the simulated encounter models and from the requirements on the safety, the bandwidth requirements on the communication equipment have been estimated, see Figure 10.15. These data-rates reflect the aircraft speed and should be large enough for the negotiation algorithms to provide collision-free trajectories at least 30 s before the collision time for 1.5 km separation between the airplanes.

Later, a model of the Zaon PCAS, based on its theoretical characteristics stated by the manufacturer has been integrated into the system to provide for non-cooperative CA. When the software model of the Integra EFIS has been obtained from its manufacturer, the interaction protocols for upload of plan waypoints into the EFIS have been implemented.

[2]The term non-cooperative is used in the sense that no negotiation nor plan-exchange methods are used for conflict detection and resolution.

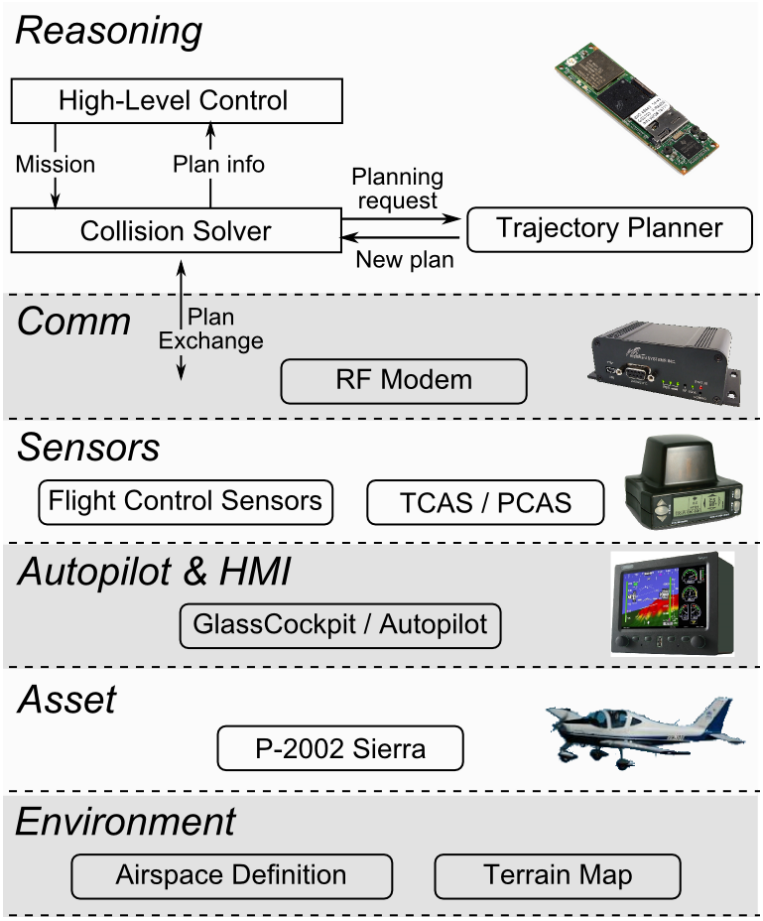


Figure 10.13: Architecture of the Safefly system for light-sport aircraft control with hardware components used for its realization.

Figure 10.14: Model of P-2002 Sierra utilizing BADA performance characteristics used for the MR simulation.

In the next stage, a series of field test experiments have been conducted to test the respective hardware modules for reliability, to measure the bandwidth and range of RF modems, and real behavior of the Zaon PCAS, see the obtained results depicted in Figure 10.16. There has been an intensive interaction with the aircraft pilots in various situations to verify the suitability of the proposed plans with the aim to increase the precision of the respective software models.

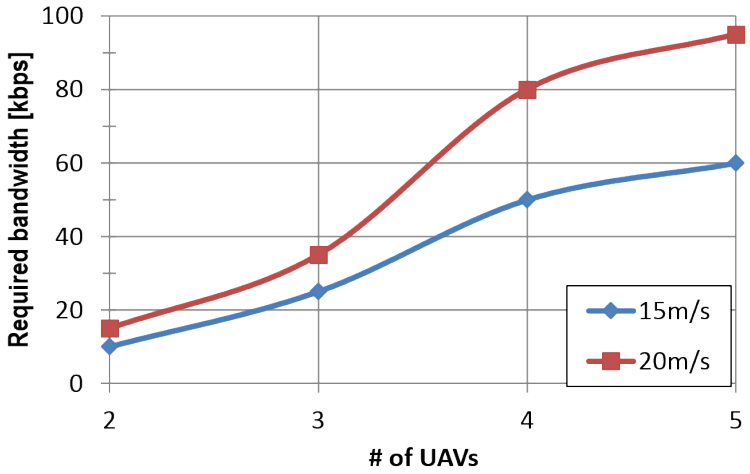


Figure 10.15: Estimation of communication requirements based on software simulation results.

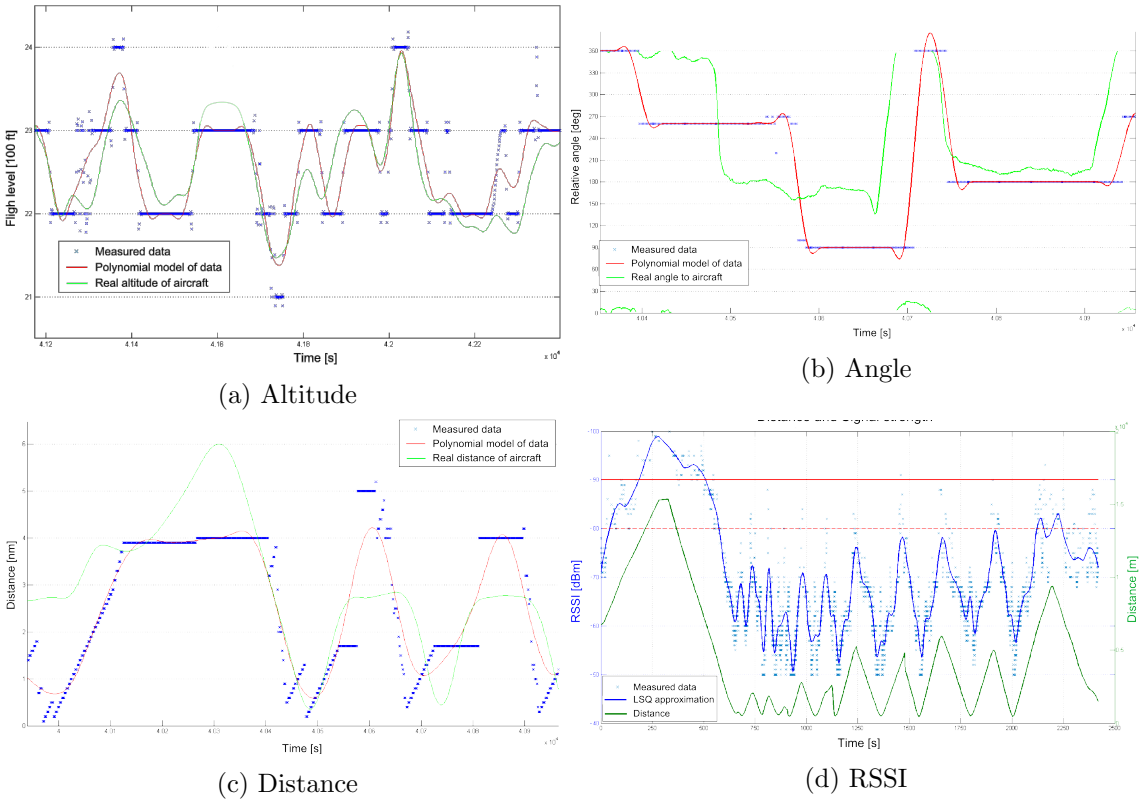
After a discussion with the pilots, a plan visualization widget has been designed that is shown in the EFIS together with other flight data. A snapshot of the widget is in Figure 10.17. It informs the pilots about aircraft in their neighborhood and helps them in following the provided best collision-free path. Note how the imprecision of the Zaon PCAS position estimation is reflected by larger dimensions of the red area.

The Safefly system has been then deployed on one of the experimental Tecnam planes, shown in Figure 10.18. It has been fed with the data of a virtual plane in the MR simulation and the pilot reactions on the plan recommendations have been observed. We integrated pilots' feedback together with their reaction times into the system and solved newly discovered hardware issues mostly related to the inter-connection of the EFIS Integra and Gumstix board. Finally, the system has been fully deployed on two experimental airplanes.

10.4 Lessons Learned

The incremental development strategy with the use of MR simulation proposed in Chapter 4 shown itself useful in all the presented projects. The MR simulation provided valuable insights into the systems that would be otherwise difficult to obtain in regular real world tests. The most valuable lessons learned are summarized in the following paragraphs.

Re-use of system modules – The modularity of the multi-level architecture enables to re-use many of the system modules in various projects even if they are of different type. For example the planning and collision avoidance algorithms, and also the communication protocols are usually very similar independently on the target application. In addition, it saves time and effort when repeatedly searching for hardware requirements since they may have been already estimated before. In particular, the communication device and onboard computers in the Safefly system have been selected based partially on our measurements of the system requirements from the



(a) Altitude

(b) Angle

(c) Distance

(d) RSSI

Figure 10.16: Results of Zaon PCAS measurements compared with real data.

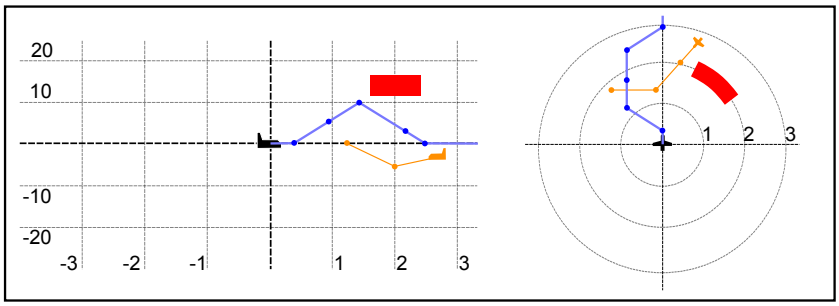


Figure 10.17: Plan visualization in Integra Glass Cockpit used for situation awareness. The red areas are the Zaon observations and the trajectory of near cooperating aircraft is shown in orange. Flight path recommendations are blue.

previous projects.

Interchangeable modules – Using the MR simulation with the incremental decreasing of virtualization level makes a replacement of individual system parts a straightforward task. Since all the modules on the same architecture layer share the same interface, they are easily interchangeable. This way, the virtualization level of the system can be quickly altered back and forth to provide the best testbed configuration level needed at the moment. Moreover, an integration of a new module to the system then requires just an implementation of the device interface according to the respective system layer and its verification in particular development

Figure 10.18: Aircraft pilot panel with the integrated Safefly system—build-in EFIS Integra, Zaon PCAS with the GPS on the flight panel. Rest of the system is integrated under the panel.

stages. This has been the case in the Safefly project when the manufacturer of Zaon XRX PCAS device stopped its production and support and some of its alternatives had to be used, i.e., the PowerFLARM.

Incremental training of system operators – Utilization of the MR simulations can significantly help target system operators to get familiar with the system without a risk of damage to hardware or third party property and also without any legal issues. The incremental decreasing of the virtualization level can be used in the learning process and teach the operators to use the system step by step. This is especially useful for aerial systems where the safety and legal issues are of the highest importance.

Localization of problems – The incremental development process helps in localization of problems that would not appear in the purely software simulation, and that would be difficult to isolate in the fully hardware deployed complex system. For example, this has been the case of the effect of the communication delays and limited data-rate.

Bridging development delays – In a case of a cooperation of multiple parties on a project, such as we experienced during the development of the Safefly project, the MR simulation showed to be very useful for substituting delayed hardware parts. During the project, there have been significant periods where software or hardware delivery from the project partners has been delayed and would disable further work of the rest of the team. In these cases, the virtualization and modeling of the respective system layers showed to be beneficial since these models can be utilized as a substitute for the hardware without the need to delay further system development.

Problematic wireless communication modeling – The MR simulations are difficult to be used for studying some of the issues related to wireless communication. The problem here lays in the fact that even when the virtual entities use hardware RF modems to communicate with the

physical entities, and thus contribute to the common bandwidth use, the fact that locations of their modems do not correspond to the assumed locations of the virtual entities cause incorrect assumptions on the RF signal propagation, and thus possibly incorrect simulation results. This issue is discussed in Chapter 6.

Chapter 11

Conclusion

This thesis addressed problems that are regularly faced during development and real world deployment of complex multi-UAV systems, where operation autonomy and mutual interaction among aircraft are required. A methodology of the incremental development using the mixed-reality (MR) simulation, where both the physical and virtual entities can co-exist and interact, has been proposed that allows addressing the problems efficiently without unnecessary costs and safety issues. For that purpose, a multi-layer system architecture is proposed and implemented into a framework for development and testing of various aspects of multi-UAV systems. The most crucial aspects of the MR simulation's architecture are addressed that account for its high fidelity simulation capability. In particular, the communication architecture for multi-UAV interactions, and methods for building, updating and using the environmental model have been addressed.

11.1 Addressing the Goals

In this thesis, we extended the concept of the MR simulations to the domain of autonomous UAVs and presented details of the necessary simulation system architecture. The proposed architecture allows reliable addressing and message exchange among the real and virtual entities and provides them with high-fidelity network simulation capabilities. Moreover, the thesis addressed methods for building and refinement of environmental model in MR simulations, and using this model for trajectory planning.

The thesis elaborated on the methodology for an incremental development of complex multi-UAV systems using MR testbeds. The seven-layer *MR simulation Architecture for development of Multi-Robot systems* (MiRAMuR) was proposed and the steps for decreasing the virtualization level of the developed system were specified. We provided the risk and cost analysis of such approach showing its efficiency compared to the full hardware deployment of the system in the later stages of the development.

The herein proposed methodology has been validated in the implementation of the command

and control system for a team of autonomous UAVs and used in UAV-based research and development projects funded by Czech Ministry of Defence, U.S. Air Force Research Lab, and U.S. Army CERDEC, and also utilized for the development of an assistant system for light-sport aircraft pilots in the project funded by the Technology Agency of the Czech Republic (TA ČR).

11.2 Other Achievements

Apart from the main goals of this thesis, that were specified in Chapter 3, we have presented several other significant results that were achieved during the research.

- An architecture for connecting a wireless network simulator for real-time simulation of the wireless communication features was proposed and implemented. Results from the deployment of our proprietary simulator were compared to an existing wireless network simulator and are presented in Chapter 6.
- During the research in the area of the representation of the MR environment, the classification of the MR environments and agents using MiRA Cube Taxonomy was extended with the ability of an agent to update the state of the simulation world. Details are presented in Chapter 7.
- A method of trajectory planning for aircraft with the limited bank angle in horizontal wind is presented in Chapter 8. This method has been developed to address a certain real-world problem, generates trajectories that are possible to be precisely executed by a real aircraft flying in the wind, and can be used for high-fidelity entity modeling.
- A method of persistent measurement of spatio-temporal phenomena (in our case, the wind) for refinement of the simulation's environmental model is proposed in Chapter 9. This method can be used for increasing the simulation's fidelity or for informed predictions of the progress of the measured phenomena (e.g., radiation, smog, toxic clouds, etc.). For the purpose of this measurements, we have defined the *Repeated Orienteering Problem* (ROP) and presented its solution.

11.3 Future Work

The development of complex multi-UAV systems will continue to be a source of interesting research problems, whether they are problems of interaction among the entities, such as coordination of their flight or cooperation for achieving mutual tasks; problems with sensors and actuators causing imprecise localization and plan execution, or problems with their deployment in real applications caused by legal and safety issues, limited flight time or high costs.

Our imminent goals related to the topics that were addressed in this thesis concern the problem of modeling the energy consumption of the aircraft, and utilization of this model for

the optimal segmentation of large areas and long-lasting missions into series of smaller tasks executable within a single flight.

The further plans for the future work are related to the sub-problems discussed in the thesis. First, to increase the fidelity of wireless channel simulation, some of the steps that would minimize the imprecisions and errors in the communication modeling discussed in Section 6.2.4 are necessary to be implemented and tested. These include using the dynamic transmit power adaptation of the GCS modem or using the multiple directional antennas. Besides, to provide even better accuracy of the flight modeling discussed in Chapter 8, the aircraft behavior and trajectory planning in arbitrary 3D wind field should be studied in the future work. Finally, as mentioned in Chapter 9, the further work can include the aggregation of multiple wind models for optimal selection of measurement locations in the urban environment and validation of the proposed method in a real application.

Appendix A

Publications

Related Publications

This section lists the author's publications related to the topic of this thesis. The number in parentheses shows the contribution of the author of the thesis.

Articles in journals, with IF:

1. **Selecký M.**, Faigl J. and Rollo M., "Communication architecture in mixed-reality simulations of unmanned systems." *Sensors*, vol. 18, no. 3, p. 853, 2018. **IF: 2.475** (**75%**)

2. **Selecký M.**, Faigl J. and Rollo M., "Analysis of using mixed reality simulations for incremental development of multi-UAV systems." *Journal of Intelligent & Robotic Systems*, pp. 1–17, 2018. doi: 10.1007/s10846-018-0875-8. **IF: 1.583** (**75%**)

3. **Selecký M.**, Váňa P., Rollo M. and Meiser T., "Wind Corrections in Flight Path Planning." *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, pp. 248–258, 2013. **IF: 0.952** (**40%**)

In proceedings, indexed by WoS:

1. **Selecký M.**, Faigl J. and Rollo M., "Mixed reality simulation for incremental development of multi-UAV systems." *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1530–1538, 2017. (**75%**)

In proceedings, indexed by Scopus:

1. **Selecký M.** and Rollo M., "Distributed control of heterogeneous team of autonomous UAVs." Proceedings of *EXPONENTIAL: Association for Unmanned Vehicle Systems (AUVSI)*, pp.707–717, 2016. (**75%**)

2. **Selecký M.**, Rollo M., Losiewicz P., Reade J. and Maida N., "Framework for incremental development of complex unmanned aircraft systems." *IEEE Integrated Communication, Navigation, and Surveillance Conference (ICNS)*, pp. J3–1, 2015. (**70%**)

3. **Selecký M.**, Štolba M., Meiser T., Čáp M., Komenda A., Rollo M., Vokřínek J., and Pěchouček M., "Deployment of multi-agent algorithms for tactical operations on UAV hardware." *Proceedings of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2013. (**12.5%**)

4. **Selecký M.** and Meiser T., "Integration of autonomous UAVs into multiagent simulation." *Acta Polytechnica*, vol. 52, no. 5/2012, p. 93–99, 2012. (**70%**)

Unrelated Publications

This section lists the author's publications unrelated to the topic of this thesis.

Articles in journals, with IF:

1. Čáp M., Novák P., Kleiner A. and **Selecký M.**, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots." *IEEE Transactions on Automation Science and Engineering*, 12(3), pp.835–849, 2015. **IF: 3.502** (**5%**)

In proceedings, indexed by WoS:

1. Čáp M., Novák P., **Selecký M.**, Faigl J. and Vokřínek J., "Asynchronous decentralized prioritized planning for coordination in multi-robot system." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3822–3829, 2013. (**15%**)

2. Rollo M., **Selecký M.** and Volf P., "Simulation of UAS integration into shared airspace for validation of impact on ATM systems." *IEEE Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pp. 6D2-1, 2017. (**33.3%**)

Patents

This section lists patents where the author of the thesis is part of the patent's authors and the patent is related to the topic of this thesis.

1. Rollo M., **Selecký M.**, Meiser T., Markovič M., Janečka A., Balda M., Jeřábek P., "Telemetrický systém pro zvýšení bezpečnosti provozu ultralehkých letounů." Czech Republic. Patent. CZ 305198. 2015-04-22. (**14.3%**)

Citations

Below we list all publications that received at least two citations (excluding auto-citations). The citation count (also excluding the auto-citations) for each publication was obtained from the Google Scholar database on 10th July 2018.

1. **Selecký M.**, Váňa P., Rollo M. and Meiser T., "Wind Corrections in Flight Path Planning." *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, pp. 248–258, 2013. **IF: 0.952** (**17 citations**)

2. **Selecký M.** and Meiser T., "Integration of autonomous UAVs into multiagent simulation." *Acta Polytechnica*, vol. 52, no. 5/2012, p. 93–99, 2012. (**6 citations**)

3. **Selecký M.**, Rollo M., Losiewicz P., Reade J. and Maida N., "Framework for incremental development of complex unmanned aircraft systems." In *IEEE Integrated Communication, Navigation, and Surveillance Conference (ICNS)*, pp. J3–1, 2015. (**3 citations**)

Bibliography

[1] H. Ahmed, S. Pierre, and A. Quintero, "A flexible testbed architecture for VANET", *Vehicular Communications*, vol. 9, pp. 115–126, 2017.

[2] E. Akim and D. Tuchin, "GPS errors statistical analysis for ground receiver measurements", *International Symposium on Space Flight Dynamics*, pp. 16–20, 2003.

[3] W. H. Al-Sabban, L. F. Gonzalez, R. N. Smith, and G. F. Wyeth, "Wind-energy based path planning for electric unmanned aerial vehicles using markov decision processes", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[4] S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations", *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, 2014.

[5] M. J. Allen and V. Lin, "Guidance and control of an autonomous soaring uav", *NASA Dryden Flight Research Center*, 2007, Technical Memorandum NASA/TM-2007-214611.

[6] Y. Allouche and M. Segal, "Cluster-based beaconing process for vanet", *Vehicular Communications*, vol. 2, no. 2, pp. 80–94, 2015.

[7] A. Alvarez, R. Orea, S. Cabrero, X. G. Pañeda, R. García, and D. Melendi, "Limitations of network emulation with single-machine and distributed ns-3", in *International ICST Conference on Simulation Tools and Techniques*, 2010, p. 67.

[8] P. Amstutz and A. Fagg, "Real time visualization of robot state with mobile virtual reality", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 241–247, 2002.

[9] AquaLab, *Swans++ - extensions to the scalable wireless ad-hoc network simulator*, online: `http://www.aqualab.cs.northwestern.edu/projects/143-swans-extensions-to-the-scalable-wireless-ad-hoc-network-simulator` [cited on 1 May 2018].

[10] M. Argany, M. Mostafavi, F. Karimipour, and C. Gagné, "A GIS based wireless sensor network coverage estimation and optimization: A voronoi approach", *Transactions on Computational Science XIV*, pp. 151–172, 2011.

[11] M. Asadpour, B. Van den Bergh, D. Giustiniano, K. Hummel, S. Pollin, and B. Plattner, "Micro aerial vehicle networks: An experimental analysis of challenges and opportunities", *IEEE Communications Magazine*, vol. 52, no. 7, pp. 141–149, 2014.

[12] N. Aschenbruck, A. Munjal, and T. Camp, "Trace-based mobility modeling for multi-hop wireless networks", *Computer Communications*, vol. 34, no. 6, pp. 704–714, 2011.

[13] A. Atamtürk, G. L. Nemhauser, and M. W. Savelsbergh, "A combined lagrangian, linear programming, and implication heuristic for large-scale set partitioning problems", *Journal of heuristics*, vol. 1, no. 2, pp. 247–259, 1996.

[14] D. Austin, L. Fletcher, and A. Zelinsky, "Mobile robotics in the long term-exploring the fourth dimension", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001, pp. 613–618.

[15] G. S. Avellar, G. A. Pereira, L. C. Pimenta, and P. Iscold, "Multi-uav routing for area coverage and remote sensing with minimum time", *Sensors*, vol. 15, no. 11, pp. 27 783–27 803, 2015.

[16] T. C. Bailey and A. C. Gatrell, *Interactive spatial data analysis*. Longman Scientific & Technical Essex, 1995, vol. 413.

[17] E. Bakolas and P. Tsiotras, "Time-optimal synthesis for the Zermelo-Markov-Dubins problem: The constant wind case", *American Control Conference (ACC)*, pp. 6163–6168, 2010.

[18] E. Balas and M. W. Padberg, "Set partitioning: A survey", *SIAM review*, vol. 18, no. 4, pp. 710–760, 1976.

[19] R. Baldessari, B. Bödekker, M. Deegener, A. Festag, W. Franz, C. C. Kellum, T. Kosch, A. Kovacs, M. Lenardi, C. Menig, *et al.*, "Car-2-car communication consortium-manifesto", 2007.

[20] M. Balduccini, D. N. Nguyen, and W. C. Regli, "Coordinating UAVs in dynamic environments by network-aware mission planning", in *Military Communications Conference (MILCOM)*, IEEE, 2014, pp. 983–988.

[21] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (FANETs): A survey", *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[22] S. Bloom, E. Korevaar, J. Schuster, and H. Willebrand, "Understanding the performance of free-space optics", *Journal of optical Networking*, vol. 2, no. 6, pp. 178–200, 2003.

[23] B. W. Boehm, "Seven basic principles of software engineering", *Journal of Systems and Software*, vol. 3, no. 1, pp. 3–24, 1983.

[24] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, *et al.*, "Advances in network simulation", *Computer*, vol. 33, no. 5, pp. 59–67, 2000.

[25] V. Brujic-Okretic, J. Guillemaut, L. Hitchin, M. Michielen, and G. Parker, "Telerobotic control using augmented reality", *Remote vehicle manoeuvring using augmented reality*, pp. 186–189, 2003.

[26] H. Bruyninckx, "Open robot control software: The OROCOS project", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2001, pp. 2523–2528.

[27] X. Bui, J. Boissonnat, P. Soueres, and J. Laumond, "Shortest path synthesis for dubins non-holonomic robot", *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2–7, 1994.

[28] A. Bujari, C. E. Palazzi, and D. Ronzani, "FANET application scenarios and mobility models", in *Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, ACM, 2017, pp. 43–46.

[29] A. Burkle, F. Segor, and M. Kollman, "Towards autonomous micro UAV swarms", *Journal of intelligent and robotic systems*, vol. 61, pp. 339–353, 2011.

[30] L. Castanet, T. Deloues, and J. Lemorton, "Channel modelling based on n-state markov chains for satcom systems simulation", *12th International Conference on Antennas and Propagation (ICAP 2003)*, vol. 1, 2003.

[31] C. C. Castello, J. Fan, A. Davari, and R. X. Chen, "Optimal sensor placement strategy for environmental monitoring using wireless sensor networks", in *Southeastern Symposium on System Theory (SSST)*, IEEE, 2010, pp. 275–279.

[32] P. Castillo-Pizarro, T. Arredondo, and M. Torres-Torriti, "Introductory survey to open-source mobile robot simulation software", *IEEE Robotics Symposium and Intelligent Robotic Meeting (LARS)*, pp. 150–155, 2010.

[33] A. Chakrabarty and J. W. Langelaan, "Energy maps for long-range path planning for small- and micro-UAVs", *AIAA Guidance, Navigation and Control Conference*, vol. 6113, 2009.

[34] A. Chakrabarty and J. W. Langelaan, "Flight path planning for UAV atmospheric energy harvesting using heuristic search", *AIAA Guidance, Navigation and Control Conference*, vol. 6113, 2010.

[35] F. Chang, A. Itzkovitz, and V. Karamcheti, "User-level resource-constrained sandboxing", *USENIX Windows Systems Symposium*, vol. 91, 2000.

[36] H. Chao and Y. Chen, "Surface wind profile measurement using multiple small unmanned aerial vehicles", in *American Control Conference (ACC)*, IEEE, 2010, pp. 4133–4138.

[37] I.-M. Chao, B. L. Golden, and E. A. Wasil, "A fast and effective heuristic for the orienteering problem", *European Journal of Operational Research*, vol. 88, no. 3, pp. 475–489, 1996.

[38] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem", *European Journal of Operational Research*, vol. 88, no. 3, pp. 464–474, 1996.

[39] H. Chen, O. Wulf, and B. Wagner, "Object detection for a mobile robot using mixed reality", *Interactive Technologies and Sociotechnical Systems*, pp. 466–475, 2006.

[40] I. Chen, B. MacDonald, and B. Wunsche, "Mixed reality simulation for mobile robots", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 232–237, 2009.

[41] I. Chen, B. MacDonald, and B. Wunsche, "Evaluating the effectiveness of mixed reality simulations for developing uav systems", in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, 2012, pp. 388–399.

[42] J. J. Chung, M. Angel, and S. Sukkarieh, "A new utility function for smooth transition between exploration and exploitation of a wind energy field", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4999–5005, 2012.

[43] A. Cobano, R. Conde, D. Alejo, and A. Ollero, "Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4429–4434, 2011.

[44] E. Costanza, A. Kunz, and M. Fjeld, "Mixed reality: A survey", *Human Machine Interaction*, pp. 47–68, 2009.

[45] N. Cressie, *Statistics for spatial data*. John Wiley & Sons, 2015.

[46] M. J. Cutler, T. W. McLain, R. W. Beard, and B. Capozzi, "Energy harvesting and mission effectiveness for small unmanned aircraft", *AIAA Guidance, Navigation and Control Conference*, 2010.

[47] M. Daily, Y. Cho, K. Martin, and D. Payton, "World embedded interfaces for human-robot interaction", *Annual Hawaii International Conference on System Sciences*, p. 6, 2003.

[48] M. Day, M. Clement, J. Russo, D. Davis, and T. Chung, "Multi-UAV software systems and simulation architecture", in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 426–435.

[49] D. Dee, S. Uppala, A. Simmons, P. Berrisford, P Poli, S Kobayashi, U Andrae, M. Balmaseda, G Balsamo, P Bauer, *et al.*, "The era-interim reanalysis: Configuration and performance of the data assimilation system", *Quarterly Journal of the royal meteorological society*, vol. 137, no. 656, pp. 553–597, 2011.

[50] S. Demers, P. Gopalakrishnan, and L. Kant, "A generic solution to software-in-the-loop", in *Military Communications Conference (MILCOM)*, IEEE, 2007, pp. 1–6.

[51] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of UAVs", in *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, IEEE, 2015, pp. 111–117.

[52] W. Du, Z. Xing, M. Li, B. He, L. H. C. Chua, and H. Miao, "Sensor placement and measurement of wind for water quality studies in urban reservoirs", *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 3, p. 41, 2015.

[53] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents", *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[54] D. J. Edwards and L. M. Silverberg, "Autonomous soaring: The montague cross-country challenge", *Journal of Aircraft*, vol. 47, no. 5, pp. 1763–1769, 2010.

[55] A. Ekici and A. Retharekar, "Multiple agents maximum collection problem with time dependent rewards", *Computers & Industrial Engineering*, vol. 64, no. 4, pp. 1009–1018, 2013.

[56] D. Evans and A. Twyman, "Flexible policy directed code safety", *IEEE Symposium on Security and Privacy*, pp. 32–45, 1999.

[57] Z. M. Fadlullah, D. Takaishi, H. Nishiyama, N. Kato, and R. Miura, "A dynamic trajectory control algorithm for improving the communication throughput and delay in UAV-aided networks", *IEEE Network*, vol. 30, no. 1, pp. 100–105, 2016.

[58] J. Faigl and R. Pěnička, "On close enough orienteering problem with dubins vehicle", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5646–5652.

[59] M. A. Farooque and J. S. Rohankar, "Survey on various noises and techniques for denoising the color image", *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, vol. 2, no. 11, pp. 217–221, 2013.

[60] *FIPA: foundation for intelligent physical agents*, online: `http://www.fipa.org` [cited on 1 May 2018].

[61] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics", *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[62] B. Garau, A. Alvarez, and G. Oliver, "Path planning of autonomous underwater vehicles in current fields with complex spatial variability: An A* approach", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2005, pp. 194–198.

[63] R. Garcia and L. Barnes, "Multi-UAV simulator utilizing X-Plane", in *Selected papers from the 2nd International Symposium on UAVs*, 2009, pp. 393–406.

[64] M. R. Garey and D. S. Johnson, ""strong"np-completeness results: Motivation, examples, and implications", *Journal of the ACM (JACM)*, vol. 25, no. 3, pp. 499–508, 1978.

[65] S. Garg and N. Ayanian, "Persistent monitoring of stochastic spatio-temporal phenomena with a small team of robots", in *Robotics: Science and Systems*, 2014.

[66] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems", *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[67] A. Goktogan, E. Nettleton, M. Ridley, and S. Sukkarieh, "Real time multi-UAV simulator", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2720–2726, 2003.

[68] A. H. Göktoğan and S. Sukkarieh, "A augmented reality system for multi-UAV missions", *Simulation Conference and Exhibition, SimTect'05*, pp. 1–6, 2005.

[69] A. H. Göktoğan and S. Sukkarieh, "Distributed simulation and middleware for networked uas", in *Unmanned Aircraft Systems*, Springer, 2008, pp. 331–357.

[70] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem", *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987, ISSN: 1520-6750.

[71] A. A. Gowardhan, E. R. Pardyjak, I. Senocak, and M. J. Brown, "A CFD-based wind solver for an urban fast response transport and dispersion model", *Environmental fluid mechanics*, vol. 11, no. 5, pp. 439–464, 2011.

[72] P. Groves, *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[73] C. Guestrin, A. Krause, and A. P. Singh, "Near-optimal sensor placements in gaussian processes", in *International Conference on Machine Learning*, ACM, 2005, pp. 265–272.

[74] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering Problem: A survey of recent variants, solution approaches and applications", *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.

[75] N. Gupta, A. Prakash, and R. Tripathi, "Medium access control protocols for safety applications in vehicular ad-hoc network: A classification and comprehensive survey", *Vehicular Communications*, vol. 2, no. 4, pp. 223–237, 2015.

[76] J. Härri, F. Filali, C. Bonnet, and M. Fiore, "VanetMobiSim: Generating realistic mobility patterns for VANETs", in *International Workshop on Vehicular ad-hoc Networks*, ACM, 2006, pp. 96–97.

[77] A. N. Hayati, R. Stoll, J. Kim, T. Harman, M. A. Nelson, M. J. Brown, and E. R. Pardyjak, "Comprehensive evaluation of fast-response, reynolds-averaged navier–stokes, and large-eddy simulation methods against high-spatial-resolution wind-tunnel data in step-down street canyons", *Boundary-Layer Meteorology*, pp. 1–31, 2017.

[78] M. A. Heath, J. D. Walshe, and S. J. Watson, "Estimating the potential yield of small building-mounted wind turbines", *Wind Energy*, vol. 10, no. 3, pp. 271–287, 2007.

[79] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan, "Effects of detail in wireless network simulation", *SCS Multiconference on Distributed Simulation*, pp. 3–11, 2001.

[80] T. R. Henderson, M. Lacage, G. F. Riley, C Dowell, and J Kopena, "Network simulations with the ns-3 simulator", *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[81] K. L. Hoffman and M. Padberg, "Solving airline crew scheduling problems by branch-and-cut", *Management science*, vol. 39, no. 6, pp. 657–682, 1993.

[82] T. Holz, A. Campbell, G. O'Hare, J. Stafford, A. Martin, and M. Dragone, "MiRA - mixed reality agents", *International Journal of Human-Computer Studies*, vol. 69(4), pp. 251–268, 2011.

[83] W. Honig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5382–5387.

[84] C. Hu, M. Li, D. Zeng, and S. Guo, "A survey on sensor placement for contamination detection in water distribution systems", *Wireless Networks*, pp. 1–15, 2016.

[85] K. Ide, P. Courtier, M. Ghil, and A. C. Lorenc, "Unified notation for data assimilation: Operational, sequential and variational (special issue data assimilation in meteology and oceanography: Theory and practice)", *Journal of the Meteorological Society of Japan. Ser. II*, vol. 75, no. 1B, pp. 181–189, 1997.

[86] IEEE 802.11 Working Group and others, "IEEE standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 6: Wireless access in vehicular environments", *IEEE Std*, vol. 802, no. 11, 2010.

[87] M. Jakob, M. Pěchouček, M. Čáp, P. Novák, and O. Vaněk, "Mixed-reality testbeds for incremental development of HART applications", *IEEE Intelligent Systems*, vol. 27, no. 2, pp. 19–25, 2012.

[88] R. Janovský, "Operator station for visualization and control of autonomous unmanned vehicles", online: `https://dspace.cvut.cz/handle/10467/68616` [cited on 1 May 2018], Master's thesis, Czech Technical University in Prague, 2017.

[89] V. Jayaraj, C Hemanth, and R. Sangeetha, "A survey on hybrid MAC protocols for vehicular ad-hoc networks", *Vehicular Communications*, vol. 6, pp. 29–36, 2016.

[90] F. Jensen and N. Petersen, *Burn-in: an engineering approach to the design and analysis of burn-in procedures*. Wiley, 1982.

[91] M. Jones, P. Leach, R. Draves, and J. Barrera, "Modular real-time resource management in the rialto operating system", *Workshop on Hot Topics in Operating Systems*, 1995.

[92] S. Joyeux, *Rock: The robot construction kit*, online at: `https://www.rock-robotics.org` [cited on 1 May 2018].

[93] A. B. Junaid, Y. Lee, and Y. Kim, "Design and implementation of autonomous wireless charging station for rotary-wing uavs", *Aerospace Science and Technology*, vol. 54, pp. 253–266, 2016.

[94] A. Kaadan, D. Zhou, H. H. Refai, and P. G. LoPresti, "Modeling of aerial-to-aerial short-distance free-space optical links", in *Integrated Communications, Navigation and Surveillance Conference (ICNS)*, IEEE, 2013, pp. 1–12.

[95] R. Kalman, "A new approach to linear filtering and prediction problems", *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[96] M. Khan, M. Yuksel, and G. Winkelmaier, "Gps-free maintenance of a free-space-optical link between two autonomous mobiles", *IEEE Transactions on Mobile Computing*, vol. 16, no. 6, pp. 1644–1657, 2017.

[97] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[98] C.-W. Ko, J. Lee, and M. Queyranne, "An exact algorithm for maximum entropy sampling", *Operations Research*, vol. 43, no. 4, pp. 684–691, 1995.

[99] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2149–2154, 2004.

[100] A. Komenda, P. Novák, and M. Pěchouček, "Domain-independent multi-agent plan repair", *Journal of Network and Computer Applications, vol. 37*, pp. 76–88, 2014.

[101] A. Komenda, J. Vokřínek, M. Čáp, and M. Pěchouček, "Developing multiagent algorithms for tactical missions using simulation", *IEEE Intelligent Systems*, vol. 28, no. 1, pp. 42–49, 2013.

[102] A. Konak, "Estimating path loss in wireless local area networks using ordinary kriging", in *Winter Simulation Conference*, 2010, pp. 2888–2896.

[103] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments", *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 964–977, 2017.

[104] D. Krajzewicz, M. Bonert, and P. Wagner, "The open source traffic simulation package sumo", *RoboCup 2006*, 2006.

[105] T. Kucner, J. Saarinen, M. Magnusson, and A. J. Lilienthal, "Conditional transition maps: Learning motion patterns in dynamic environments", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1196–1201.

[106] J. Kuffner and J. Latombe, "Fast synthetic vision, memory, and learning models for virtual humans", *Proceedings of Computer Animation*, pp. 118–127, 1999.

[107] E. Kuiper and S. Nadjm-Tehrani, "Mobility models for UAV group reconnaissance applications", in *International Conference on Wireless and Mobile Communications (ICWMC).*, IEEE, 2006, pp. 33–33.

[108] K. Kumar and P. Reel, "Analysis of contemporary robotics simulators", *IEEE Conference on Emerging Trends in Electrical and Computer Technology (ICETECT)*, pp. 661–665, 2011.

[109] N. R. Lawrance and S. Sukkarieh, "Autonomous exploration of a wind field with a gliding aircraft", *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 3, pp. 719–733, 2011.

[110] N. R. Lawrance and S. Sukkarieh, "Path planning for autonomous soaring flight in dynamic wind fields", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2499–2505, 2011.

[111] N. Le Sommer and F. Guidec, "A contract-based approach of resource-constrained software deployment", *International Working Conference on Component Deployment*, pp. 15–30, 2002.

[112] J. Lessmann, P. Janacik, L. Lachev, and D. Orfanus, "Comparative study of wireless network simulators", *Seventh International Conference on Networking*, pp. 517–523, 2008.

[113] B. Li, Y. Jiang, J. Sun, L. Cai, and C.-Y. Wen, "Development and testing of a two-uav communication relay system", *Sensors*, vol. 16, no. 10, p. 1696, 2016.

[114] B. Magyar, Z. Forhecz, and P. Korondi, "Developing an efficient mobile robot control algorithm in the webots simulation environment", *IEEE International Conference on Industrial Technology*, pp. 197–184, 2003.

[115] M. Marino, A. Fisher, R. Clothier, S. Watkins, S. Prudden, and C. S. Leung, "An evaluation of multi-rotor unmanned aircraft as flying wind sensors", *International Journal of Micro Air Vehicles*, vol. 7, no. 3, pp. 285–299, 2015.

[116] G. Matheron, "Principles of geostatistics", *Economic geology*, vol. 58, no. 8, pp. 1246–1266, 1963.

[117] P. Maybeck, *Stochastic models, estimation, and control*. Academic press, 1982.

[118] T. G. McGee, S. Spry, and J. Hedrick, "Optimal path planning in a constant wind with a bounded turning rate", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. 1–11, 2005.

[119] T. Meiser, "Distributed topology control in MANETs", Master's thesis, Czech Technical University in Prague, 2012.

[120] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and gazebo", in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Springer, 2012, pp. 400–411.

[121] N. Michael, E. Stump, and K. Mohta, "Persistent surveillance with a team of MAVs", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 2708–2714.

[122] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays", *IEICE Transactions on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[123] P. Milgram, A. Rastogi, and J. Grodski, "Telerobotic control using augmented reality", *IEEE International Workshop on Robot and Human Communication*, pp. 21–29, 1995.

[124] B. Minasny, A. B. McBratney, and D. J. J. Walvoort, "The variance quadtree algorithm: Use for spatial sampling design", *Computers & Geosciences*, vol. 33, no. 3, pp. 383–392, 2007.

[125] M. Montemerlo, N Roy, and S Thrun, *Carnegie mellon robot navigation toolkit*, Software package, download online: `http://carmen.sourceforge.net` [cited on 1 May 2018], 2002.

[126] R. Moseley, "Merged reality systems: Bringing together automation and tracking through immersive geospatial connected environments", *Science and Information Conference (SAI)*, pp. 732–735, 2014.

[127] F. Mutter, S. Gareis, B. Schatz, A. Bayha, F. Gruneis, M. Kanis, and D. Koss, "Model-driven in-the-loop validation: Simulation-based testing of UAV software using virtual environments", in *18th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*, 2011, pp. 269–275.

[128] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode, "Trafficview: Traffic data dissemination using car-to-car communication", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 3, pp. 6–19, 2004.

[129] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "RFC4861: Neighbor discovery for ip version 6 (IPv6)", *Standards Track*, 2007, online: `http://www.ietf.org/rfc/rfc4861.txt` [cited on 1 May 2018].

[130] S. Nassar, "Accurate INS/DGPS positioning using INS data de-noising and autoregressive (AR) modeling of inertial sensor errors", *Geomatica*, vol. 59, no. 3, pp. 283–294, 2005.

[131] M. Neophytou, A. Gowardhan, and M. Brown, "An inter-comparison of three urban wind models using oklahoma city joint urban 2003 wind field measurements", *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 99, no. 4, pp. 357–368, 2011.

[132] P. Neubert, N. Sünderhauf, and P. Protzel, "Superpixel-based appearance change prediction for long-term navigation across seasons", *Robotics and Autonomous Systems*, vol. 69, pp. 15–27, 2015.

[133] P. C. Ng, S. C. Liew, K. C. Sha, and W. T. To, "Experimental study of hidden node problem in ieee 802.11 wireless networks", *Sigcomm Poster*, vol. 26, 2005.

[134] J. Nguyen, N. Lawrance, R. Fitch, and S. Sukkarieh, "Energy-constrained motion planning for information gathering with autonomous aerial soaring", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3810–3816, 2013.

[135] N. Nikaein and C. Bonnet, "Topology management for improving routing and network performances in mobile ad hoc networks", *Mobile Networks and Applications*, vol. 9, no. 6, pp. 583–594, 2004.

[136] A Nuic, "User manual for the base of aircraft data (bada) revision 3.10", *Atmosphere*, vol. 2010, 2010.

[137] A. Nuic, D. Poles, and V. Mouillet, "Bada: An advanced aircraft performance model for present and future atm systems", *International Journal of Adaptive Control and Signal Processing*, vol. 24, no. 10, pp. 850–866, 2010.

[138] H. S. Nwana, L. C. Lee, and N. R. Jennings, "Coordination in software agent systems", *British Telecom Technical Journal*, vol. 14, no. 4, pp. 79–88, 1996.

[139] Ofcom, *Measuring mobile broadband performance in the uk - 4g and 3g network performance*, online: `https://www.ofcom.org.uk/research-and-data/telecoms-research/broadband-research/mobile-broadband-nov-14` [cited on 1 May 2018], 2014.

[140] A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the a&a meta-model for multi-agent systems", *Autonomous Agents and Multiagent Systems*, vol. 17(3), pp. 432–456, 2008.

[141] O. S. Oubbati, A. Lakas, F. Zhou, M. Güneş, and M. B. Yagoubi, "A survey on position-based routing protocols for flying ad hoc networks (FANETs)", *Vehicular Communications*, 2017.

[142] A. Pal, R. Tiwari, and A. Shukla, "Communication constraints multi-agent territory exploration task", *Applied intelligence*, vol. 38, no. 3, pp. 357–383, 2013.

[143] M. Park and Y. Gao, "Error and performance analysis of mems-based inertial sensors with a low-cost gps receiver", *Sensors*, vol. 8, no. 4, pp. 2240–2261, 2008.

[144] S.-Q. Peng and L. Xie, "Effect of determining initial conditions by four-dimensional variational data assimilation on storm surge forecasting", *Ocean Modelling*, vol. 14, no. 1, pp. 1–18, 2006.

[145] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1210–1217, 2017.

[146] A. R. Perry, "The flightgear flight simulator", in *USENIX Annual Technical Conference*, 2004.

[147] L. R. Pinto, A. Moreira, L. Almeida, and A. Rowe, "Characterizing multihop aerial networks of cots multirotors", *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 898–906, 2017.

[148] M. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters", *Journal of the American statistical association*, vol. 94, no. 446, pp. 590–599, 1999.

[149] I. Pizetta, A. Brandao, and M. Sarcinelli-Filho, "A hardware-in-the-loop platform for rotary-wing unmanned aerial vehicles", *Journal of Intelligent & Robotic Systems*, vol. 84, no. 725, pp. 725–743, 2016. DOI: 10.1007/s10846-016-0357-9.

[150] J. Prevost, M. Joordens, and M. Jamshidi, "Simulation of underwater robots using MS robot studio", *IEEE International Conference on System of Systems Engineering*, pp. 1–5, 2008.

[151] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, L. J., R. Wheeler, and Y. A. Ng, "ROS: An open-source robot operating system", *IEEE International Conference on Robotics and Automation (ICRA): workshop on open source software*, pp. 1–6, 2009.

[152] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*. Prentice Hall PTR New Jersey, 1996, vol. 2.

[153] L. Ricci, C. Castelfranchi, M. Piunti, and O. Boissier, "Mirror worlds as agent societies situated in mixed reality environments", *Workshop on Coordination, Organisations, Institutions and Norms (COIN 2014)*, 2014.

[154] L. Rodriguez Salazar, J. A. Cobano, and A. Ollero, "Small uas-based wind feature identification system part 1: Integration and validation", *Sensors*, vol. 17, no. 1, p. 8, 2016.

[155] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1321–1326.

[156] J. C. Rubio and S. Kragelund, "The trans-pacific crossing: Long range adaptive path planning for uavs through variable wind fields", in *The 22nd Digital Avionics Systems Conference (DASC)*, IEEE, vol. 2, 2003, 8–B.

[157] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjorungnes, "Hedonic coalition formation for distributed task allocation among wireless agents", *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1327–1344, 2011.

[158] D. Salomon, *Curves and surfaces for computer graphics*. Springer Science & Business Media, 2007.

[159] J. Sanchez-Lopez, J. Pestana, P. de la Puente, and P. Campoy, "A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-UAV systems: Simulation and experimentation", *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1–4, pp. 779–797, 2016.

[160] T. W. Sandhlom and V. R. Lesser, "Coalitions among computationally bounded agents", *Artificial intelligence*, vol. 94, no. 1-2, pp. 99–137, 1997.

[161] P. Santi, *Topology Control in Wireless Ad Hoc and Sensor Networks*. John Wiley & Sons, Ltd., 2005.

[162] P. Scerri, T. Von Gonten, G. Fudge, S. Owens, and K. Sycara, "Transitioning multiagent technology to UAV applications", *International Conference on Autonomous Agents and Multi-Agent Systems: industrial track (AAMAS)*, pp. 89–96, 2008.

[163] B. Schünemann, K. Massow, and I. Radusch, "Realistic simulation of vehicular communication and vehicle-2-x applications", in *International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 62.

[164] M. Selecký and T. Meiser, "Integration of autonomous UAVs into multi-agent simulation", *Acta Polytechnica*, vol. 52, no. 5, pp. 93–99, 2012.

[165] M. Selecký and M. Rollo, "Distributed control of heterogeneous team of autonomous UAVs", in *EXPONENTIAL 2016: Association for Unmanned Vehicle Systems (AUVSI)*, 2016, pp. 707–717.

[166] M. Selecký, M. Rollo, P. Losiewicz, J. Reade, and N. Maida, "Framework for incremental development of complex unmanned aircraft systems", in *Integrated Communication, Navigation, and Surveillance Conference (ICNS)*, IEEE, 2015, J3–1.

[167] M. Selecký, M. Štolba, T. Meiser, M. Čáp, A. Komenda, M. Rollo, J. Vokřínek, and M. Pěchouček, "Deployment of multi-agent algorithms for tactical operations on UAV hardware", *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1407–1408, 2013.

[168] M. Selecký, "Simulation of wireless communication networks", online: `https://cyber.felk.cvut.cz/theses/papers/91.pdf` [cited on 1 May 2018], Master's thesis, Czech Technical University in Prague, 2010.

[169] M. Selecký, J. Faigl, and M. Rollo, "Mixed reality simulation for incremental development of multi-uav systems", in *International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2017, pp. 1530–1538.

[170] M. Selecký, J. Faigl, and M. Rollo, "Analysis of using mixed reality simulations for incremental development of multi-UAV systems", *Journal of Intelligent & Robotic Systems*, pp. 1–17, 2018. DOI: `10.1007/s10846-018-0875-8`.

[171] M. Selecký, J. Faigl, and M. Rollo, "Communication architecture in mixed-reality simulations of unmanned systems", *Sensors*, vol. 18, no. 3, p. 853, 2018.

[172] M. Selecký, P. Váňa, M. Rollo, and T. Meiser, "Wind corrections in flight path planning", *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, pp. 248–258, 2013.

[173] T. B. Sheridan and W. L. Verplank, "Human and computer control of undersea teleoperators", Massachusetts Institute of Technology Cambridge Man-Machine Systems Laboratory, Tech. Rep., 1978.

[174] C. M. Silva, B. M. Masini, G. Ferrari, and I. Thibault, "A survey on infrastructure-based vehicular networks", *Mobile Information Systems*, vol. 2017, 2017.

[175] M. Simic, C. Bil, and V. Vojisavljevic, "Investigation in wireless power transmission for uav charging", *Procedia Computer Science*, vol. 60, pp. 1846–1855, 2015.

[176] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots", *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[177] D. Šišlák, M. Rehák, M. Pěchouček, M. Rollo, and D. Pavlíček, "A-globe: Agent development platform with inaccessibility and mobility support", in *Software agent-based applications, platforms and development kits*, Springer, 2005, pp. 21–46.

[178] C. C. Slama, C. Theurer, and S. W. Henriksen, *Manual of photogrammetry*. American Society of photogrammetry, 1980.

[179] R. Smith, *ODE: Open dynamics engine*, online: `http://www.ode.org` [cited on 1 May 2018], 2003.

[180] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution", *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

[181] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments", *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, 2012.

[182] S.-H. Song, D.-H. Kim, and C.-H. Chang, "Experimental reliability analysis of multi-UAV simulation with TMO-based distributed architecture and global time synchronization", in *IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW)*, IEEE, 2010, pp. 211–218.

[183] T. Stocky and J. Cassell, "Shared reality: Spatial intelligence in intuitive user interfaces", in *International Conference on Intelligent User Interfaces*, ACM, 2002, pp. 224–225.

[184] R. Stranders, E. M. De Cote, A. Rogers, and N. R. Jennings, "Near-optimal continuous patrolling with teams of mobile information gathering agents", *Artificial intelligence*, vol. 195, pp. 63–105, 2013.

[185] C. Sunshine, "Addressing problems in multi-network systems", *IEEE Infocom*, vol. 82, 1982.

[186] K. A. Suzuki, P. Kemper Filho, and J. R. Morrison, "Automatic battery replacement system for uavs: Analysis and design", *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 563–586, 2012.

[187] L. Techy and C. A. Woolsey, "Minimum-time path planning for unmanned aerial vehicles in steady uniform winds", *Journal of guidance, control, and dynamics*, vol. 32, no. 6, pp. 1736–1746, 2009.

[188] L. Techy, C. A. Woolsey, and K. Morgansen, "Planar path planning for flight vehicles in wind with turn rate and acceleration bounds", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3240–3245, 2010.

[189] E. Teng, J. D. Falcão, and B. Iannucci, "Holes-in-the-sky: A field study on cellular-connected UAS", in *International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2017, pp. 1165–1174.

[190] S. Thrun, "Particle filters in robotics", *Conference on Uncertainty in Artificial Intelligence*, pp. 511–518, 2002.

[191] S. Thrun *et al.*, "Robotic mapping: A survey", *Exploring artificial intelligence in the new millennium*, vol. 1, pp. 1–35, 2002.

[192] T. Toksoz, J. Redding, M. Michini, B. Michini, J. P. How, M. Vavrina, and J. Vian, "Automated battery swap and recharge to enable persistent UAV missions", in *AIAA Infotech@ Aerospace Conference*, 2011, pp. 1–10.

[193] M. Torres-Torriti, T. Arredondo, and P. Castillo-Pizarro, "Analysis of contemporary robotics simulators", *Robotica*, pp. 1–32, 2015.

[194] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey", *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[195] A. Varga, "Discrete event simulation system", in *European Simulation Multiconference (ESM)*, 2001.

[196] M. Čáp, P. Novák, M. Selecký, J. Faigl, and J. Vokřínek, "Asynchronous decentralized prioritized planning for coordination in multi-robot system", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3822–3829, 2013.

[197] D. Šišlák, P. Volf, A. Komenda, J. Samek, and M. Pěchouček, "Agent-based multi-layer collision avoidance to unmanned aerial vehicles", *Integration of Knowledge Intensive Multi-Agent Systems*, pp. 365–370, 2007.

[198] D. Šišlák, P. Volf, v. Kopřiva, and M. Pěchouček, "AGENTFLY: A multi-agent airspace test-bed", *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1665–1666, 2008.

[199] D. Šišlák, P. Volf, and M. Pěchouček, "Flight trajectory path planning", *International Conference on Automated Planning and Scheduling (ICAPS): Scheduling and Planning Applications Workshop (SPARK)*, pp. 76–83, 2009.

[200] J. Škaloud, *Optimizing georeferencing of airborne survey systems by INS/DGPS (Doctoral dissertation)*. University of Calgary, 1999.

[201] M. Štolba, M. Selecký, T. Meiser, M. Čáp, M. Rollo, A. Komenda, J. Vokřínek, and M. Pěchouček, "Agentfly-in-air: HW deployment, experimentation and testing of the results from the tactical agentfly and agentscout projects", CTU in Prague, FEE, Tech. Rep., 2012.

[202] R. Wahbe, S. Lucco, T. Anderson, and S. Graham, "Efficient software-based fault isolation", *14th ACM Symposium on Operating Systems Principles*, 1993.

[203] D. Walvoort, D. Brus, and J. De Gruijter, "An r package for spatial coverage sampling and random sampling from compact geographical strata by k-means", *Computers & Geosciences*, vol. 36, no. 10, pp. 1261–1267, 2010.

[204] J. Ware and N. Roy, "An analysis of wind field estimation and exploitation for quadrotor flight in the urban canopy layer", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1507–1514, 2016.

[205] M. T. Wolf, L. Blackmore, Y. Kuwata, N. Fathpour, A. Elfes, and C. Newman, "Probabilistic motion planning of balloons in strong, uncertain wind fields", *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1123–1129, 2010.

[206] M. Wooldridge and N. Jennings, "Intelligent agents: Theory and practice", *Knowledge Engineering Review*, vol. 10(2), pp. 115–152, 1994.

[207] M. Xie and C. Lai, "Reliability analysis using an additive weibull model with bathtub-shaped failure rate function", *Reliability Engineering & System Safety*, pp. 87–93, 1996.

[208] J. Young, E. Sharlin, and J. Boyd, "Implementing bubblegrams: The use of haar-like features for human-robot interaction", *IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 298–303, 2006.

[209] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks", *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, 2016.

[210] J. Yu, S. Karaman, and D. Rus, "Persistent monitoring of events with stochastic arrivals at multiple stations", *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 521–535, 2015.

[211] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A library for parallel simulation of large-scale wireless networks", in *Workshop on Parallel and Distributed Simulation (PADS)*, IEEE, 1998, pp. 154–161.

[212] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization", *IEEE Transactions on Wireless Communications*, 2017.

[213] F. Zhang and A. Knoll, "Systematic error modeling and bias estimation", *Sensors*, vol. 16, no. 5, p. 729, 2016.