End of Master's Degree Project

Automatic Control and Robotics

Micro Aerial Vehicles (MAV) Assured Navigation in Search and Rescue Missions

Robust Localization, Mapping and Detection

MEMORY

Author: Daniel Serrano López

Director/s: Juan Andrade-Cetto (director)
Maarten Uijt de Haag (co-director)

Call: November 2015

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

Abstract

This Master's Thesis describes the developments on robust localization, mapping and detection algorithms for Micro Aerial Vehicles (MAVs). The localization method proposes a seamless indoor-outdoor multi-sensor architecture. This algorithm is capable of using all or a subset of its sensor inputs to determine a platform's position, velocity and attitude (PVA). It relies on the Inertial Measurement Unit as the core sensor and monitors the status and observability of the secondary sensors to select the most optimum estimator strategy for each situation. Furthermore, it ensures a smooth transition between filters structures. This document also describes the integration mechanism for a set of common sensors such as GNSS receivers, laser scanners and stereo and mono cameras.

The mapping algorithm provides a fully automated fast aerial mapping pipeline. It speeds up the process by pre-selecting the images using the flight plan and the onboard localization. Furthermore, it relies on Structure from Motion (SfM) techniques to produce an optimized 3D reconstruction of camera locations and sparse scene geometry. These outputs are used to compute the perspective transformations that project the raw images on the ground and produce a geo-referenced map. Finally, these maps are fused with other domains in a collaborative UGV and UAV mapping algorithms.

The real-time aerial detection of victims is based on a thermal camera. The algorithm is composed by three steps. Firstly, a normalization of the image is performed to get rid of the background and to extract the regions of interest. Later the victim detection and tracking steps produce the real-time geo-referenced locations of the detections.

The thesis also proposes the concept of a MAV Copilot, a payload composed by a set of sensors and algorithm the enhances the capabilities of any commercial MAV. To develop and validate these contributions, a prototype of a search and rescue MAV and the Copilot has been developed.

These developments have been validated in three large-scale demonstrations of search and rescue operations in the context of the European project ICARUS: a shipwreck in Lisbon (Portugal), an earthquake in Marche (Belgium), and the Fukushima nuclear disaster in the euRathlon 2015 competition in Piombino (Italy).

Acknowledgements

I would like to thank my thesis director Dr. Juan Andrade Cetto, director of the Institut de Robòtica i Informàtica Industrial (IRI), for the great help and advice in putting this thesis together. And my thesis co-director Dr. Maarten Uijt de Haag, from the School of Electrical Engineering and Computer Science at Ohio University, who has been a great mentor on inertial and GNSS navigation and taught me that the enthusiasm for research makes things possible.

I am also indebted to Eurecat Technology Center for the opportunity and flexibility and to the European project ICARUS and the partners that have made this experience enriching and fruitful. A very special thank to my team at Eurecat during these month for their continuous support with hardware, research, dataset recording, etc. Without every each of you backing me up while I had to run to study or submit somthing, this would have not been possible. In particular for the 12.000 kilometers driven together all around Europe to attend the field trials and, of course, all the fun.

I would like to thank my family for your support and love throughout my life. You know everything. Any word here would not thank you half as much as you deserve. And to my friends that I hope still remember me and are not too angry about my long absence.

The deepest gratitude is to you Elena. I would have never enrolled the Master's degree if you had not encouraged me and I would have never finished it without your unconditional support even throughout pregnancy and after giving birth. Yes!, our "wee" daughter arrived during these studies and from the very first day, Bianca, you have been a motivation to me and have taught me everything about desire for self-improvement. I hope this effort will serve you as example in your future. Apologies for all the time taken from you both. We three made it.

Table of Contents

Table of Figures

1. Introduction

1.1. Motivation

In the event of large crises such as the earthquakes in Haiti and Japan or the flooding in Bosnia and Pakistan, a primordial task of the rescue services is the search for human survivors on the incident site. According to the most recent World Disaster Report between 2000 and 2009, 1,105,353 people were killed worldwide in 7,184 disasters with another 2,550,272 more directly affected, creating an estimated total cost of $986,7 billion [1].

The use of Micro Aerial Vehicles (MAVs) in search and rescue missions provides benefits for users due to their low cost, portability, and potential fields of use. Their use may improve the response time and coverage allowing search and rescue teams to systematically survey and perform mapping of areas with high level of details and accuracy at a resolution of up to 2-5 cm per pixel in real time and without any physical interaction within dangerous zones [2].

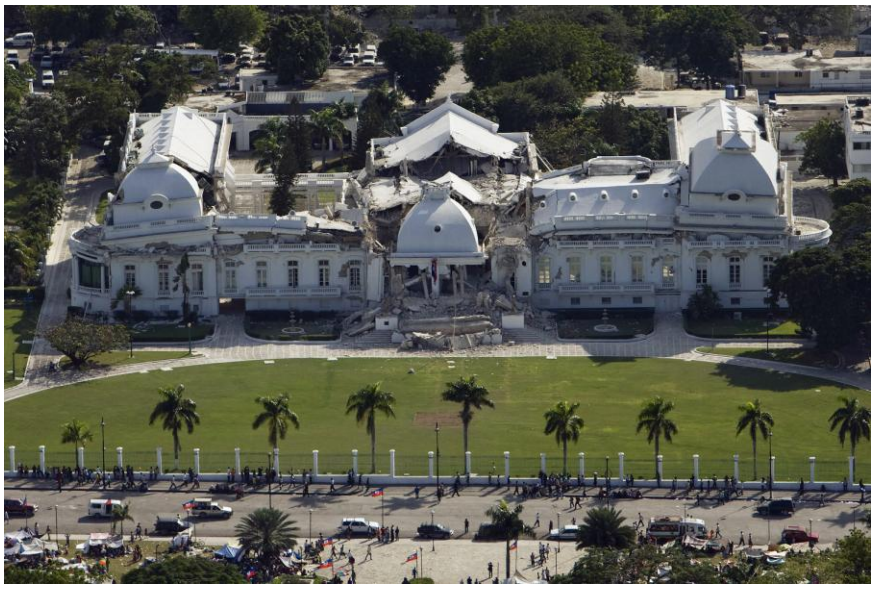


Fig. 1 Haitian national palace. 2010 Earthquake. Port-au-Prince (Haiti)

A worldwide report on interventions of MAVs in disasters states that, surprisingly, in all cases the system was tele-operated. Analyzing these experiences, the major open research topics in the field of disaster robotics are: autonomy for guarded motion, localization and mapping, better human-robot interaction and multi-agent coordination [1].

1.2. Objectives

The overall objective of this Master's Thesis is to enable any MAV system with robust localization, mapping and detection algorithms that allow safe and effective interventions in search and rescue missions.

The project shall identify the key sensors and components to meet the requirements of such scenario. A reference architecture shall provide the framework for the developments, covering both hardware and software components and interfaces. A high-performance MAV will serve as test-bench to validate the algorithms in real operations. This system must be adapted to satisfy the requirements of a search and rescue mission.

The specific technical objectives of the project are:

- robust MAV navigation in search and rescue scenarios,
- fast aerial mapping of a disaster area,
- real-time victim search using a thermal camera,
- validation in large-scale highly realistic scenarios.

1.3. Origin of the project

The opportunity to enrol the Master's degree in Automatic Control and Robotics arrived after quite some years working in the industry of software for mobile robots. My current employment is at the Eurecat Technology Centre, in Barcelona, where I lead the Unmanned Systems Group. The definition of this project is in line with the research interests of the group, in particular, with our contributions to the European project ICARUS (Integrated Components for Assisted Rescue and Unmanned Search operations) Feb.2012 - Feb.2016 [3] [4].

My involvement in this project has given me the opportunity to apply many of the expertises acquired during the Master's studies as reflected in this document.

1.4. Scope

This document is a report on the objectives, methods, implementation and experimental results generated in the context of this project.

The rest of the document has been organized as follows:

- **Section 2** describes the proposed MAV hardware and software architecture and the concept of the MAV Copilot.

- **Section 3** provides a detailed description of the seamless multi-sensor navigation framework, the navigation sensors considered, a experiment of GPS, inertial and laser navigation and how it applies to the Copilot sensors setup.

- **Section 4** describes the aerial fast mapping algorithm based on Structure from Motion.

- **Section 5** shows the aerial detection of survivors algorithm running on thermal images.

- **Section 6** illustrates three large-scale field validation of this project.

- **Section 7** lists some publications resulting from this work.

At Eurecat Technology Center, I am the team leader of the research group working on the ICARUS project and, as such, I have been deeply involved in every task of the project. Some of the work has obviously been done with the support and collaboration of other researchers. Within the general umbrella of the contribution of Eurecat to the ICARUS project, the contributions specifically developed in the scope of this thesis are:

- The MAV Copilot concept which is described in section □. For the sake of completeness, section 2 also provides details of the MAV platform and the hardware prototype of the Copilot concept developed by Eurecat.

- An indoor-outdoor multi-sensor localization scheme for robust MAV navigation which is described in section 3. This work is the result of an international collaboration with Ohio University (USA) and Dr. Maarten Uijt De Haag who is co-director of this thesis. My contributions focus specifically on the joint design of the multi-sensor integration architecture, the real-time implementation in ROS, the generation of stereo and mono correction measurements, dataset recording and experimental campaign with both cars and MAVs.

- A fast aerial mapping using Structure from Motion described in section 4. My contributions are the conceptualization and implementation of the mapping algorithm, the multi-domain (air and ground) collaborative mapping and the maps classification technique. This work was initially supported by the intern Jaume Serret who helped in the evaluation of the different open-source SfM packages and the first beta implementation of some of the steps; and later supported by the former Eurecat researcher Jesús Romero who helped with bug fixing, the blending technique and the

adaptation of the map classifier to a ROS service.

- The real-time victim search using a thermal camera which is described in section 5. My contributions are the overall victims detector and tracking. Again, this work was initially supported by Jesus Romero who implemented the fixed normalization method and integrated a general-purpose blob detector.

1.5. Analysis of the state of the art

1.5.1. History of disaster robotics: MAVs in search and rescue

This section describes the most relevant initiatives in the use of UAVs for disaster response. According to [1], rescue robotics is a relatively small discipline within corporate and academic circles. The first recorded speculation about the potential use of robots for search and rescue and other emergencies was in the 1980s [5]. The established academic research did not start until many years later. In 1995, two simultaneous disasters inspired the beginning of the two leading research groups on the field nowadays:

- The bombing of Alfred Murrah Federal building in Oklahoma (USA) inspired the beginnings of Dr. Robin Murphy, currently director of Center for Robot-Assisted Search and rescue (CRASAR) and Founder of Roboticists Without Borders.
- The Hanshin-Awaji earthquake in Kobe inspired the beginnings of the group of Prof. Satoshi Tadokoro currently at Tohoku University in Japan. He is currently the president-elect of the IEEE Robotics and Automation Society (RAS).

This was followed by two mobile robot competitions to engage the scientific community:

- Association for the Advancement of Artificial Intelligence Mobile Robots Competition (USA)
- RoboCup Rescue League (international): still ongoing

There is an annual gathering at the IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). This year it was the 13th edition and was held in Purdue University (Indiana-USA).

The European Commission acknowledged the need of research on the field and has funded several projects in the recent years. Namely NIFTi [6], ICARUS [7], Sherpa [8], TRADR [9] and euRathlon[10] . In particular, euRathlon is an outdoor robotics competition which invites teams to test the intelligence and autonomy of their robots in realistic mock emergency-response scenarios inspired by the 2011 Fukushima accident.

Fig. 2 CRASAR robot at the World Trade Center, 2011

As a general consensus, the first known use of rescue robots is the deployment of the DARPA Tactical Mobile Robots by the Center for Robot-Assisted Search and rescue (CRASAR) at the World Trade Center disaster in New York 2001. Regarding the use of UAVs, two military systems were used some years later:

- The Predator UAV in the aftermath of Hurricane Katrina (2005)
- The Honeywell T-Hawk UAV in the Fukushima Daiichi nuclear emergency (2011)

The EU-funded project NIFTI has demonstrated the first active deployment of robots in Search and rescue in Europe [11]. UAVs and UGVs were used for structural damage assessment. The UAVs were mostly tele-operated. A conclusion from this operations is that robot autonomy should be deployed to support humans and help them handle the cognitive load. This is to be ensured through user-centric methods.

Robin Murphy restricts rescue robots in [1] to "unmanned systems used as tactical, organic assets". Tactical means the robot is directly controlled by stakeholders who need to make fairly rapid decisions about the event. Organic means that the robot is deployed, maintained, transported and tasked and directed by the stakeholders. This is in contrast to a strategic asset, such as helicopter or high-altitude drones, which is deployed by an agency and then filters information to the tactical teams. Robin Murphy reports only 11 known deployments of UAVs:

Year	Event	Robot	Notes
2005	Hurricane Katrina (USA)	Like90 T-Rex	A micro rotorcraft used for Reconnaissance and Mapping
		AeroVironment Raven	A small fixed-wing used for Reconnaissance and Mapping

		iSENSYS IP-3	A small rotorcraft used for Structural Inspection
2005	Hurricane Wilma (USA)	Like90 T-Rex	A micro rotorcraft used for Structural Inspection
2007	Berkman Plaza II (USA)	iSENSYS IP-3	A micro fixed-wing used for Structural Inspection
2009	L'Aquila Earthquake (Italy)	Custom	A micro rotorcraft used for demonstration
2010	Haiti earthquake (Haiti)	Elbit Skylark	A small fixed-wing used for Reconnaissance and Mapping
2011	Christchurch earthquake (New Zealand)	Parrot AR drone	A micro rotorcraft used for Structural Inspection
2011	Tohoku earthquake (Japan)	Pelican	A micro rotorcraft used for Reconnaissance and Mapping
2011	Fukushima nuclear emergency (Japan)	Custom	A small fixed-wing used for Reconnaissance and Mapping
		Honeywell T-Hawk	A small rotorcraft used for Reconnaissance and Mapping and Structural Inspection
2011	Evangelos Florakis naval base explosion (Cyprus)	AscTec Falcon	A micro rotorcraft used for Structural Inspection
2011	GreatThaiand Flood (Thailand)	Unkown	A small fixed-wing used for Reconnaissance and Mapping
2012	Finale Emilia earthquake (Italy)	NIFTi 1 and 2	A small rotorcraft used for Reconnaissance and Mapping and Structural Inspection

Table 1 Known deployment of UAVs in disasters [1]

Fixed-wing have serve better to wide area reconnaissance. For instance, in the Hurricane Katrina, the UAV was used to detect roads blocked by trees, and Haiti earthquake, to assess the state of a distant orphanage. However, the table above shows that rotorcraft are the preferred platform. They can hover at low-altitudes (not possible for helicopters) and the data quality is comparable with fixed-wings. These are the gaps identified in these events:

- Lack of good optical acuity, image stabilization and enhancement to adapt to changes in illumination
- Inability to review photographs while the vehicle is in flight so that the mission is not repeated
- Lack of infrared sensors needed for night operations
- Lack of ability to fuse data from multiple sensors and a priori data (satellite)

1.5.2. Robust MAV navigation

To enable safe and reliable operation of MAVs in search and rescue, at any time in any environment, a position and velocity estimation is required that is robust and not solely dependent on the Global Navigation Satellite System (GNSS). In aerial robotics, it is required to estimate the platform's pose (position and attitude) in three dimensions (3D) and therefore solve for 6 degrees-of-freedom (6DOF): $(x, y, z, \phi, \theta, \psi)$. GPS provides measurements suited for estimation of the position and the IMU provide measurements that can be used to estimate all 6 unknowns.

Alternative navigation technologies may include (a) the integration of inertial sensors with imagery and laser scanners [12], (b) beacon-based navigation (i.e. pseudolites, UWB) [13], (c) or navigation using signals of opportunity [13].

Two-dimensional (2D) laser scanners have been used extensively in robotics to estimate the 3 degrees-of-freedom (3DOF), (x, y, ψ), in 2D navigation; for example, [14] and [15] describe methods that extract features such as line segments and points from the laser scans and use these features to estimate the position and heading of the robot or aid an inertial navigator.

Alternatively, a large amount research papers have addressed laser scanner-based 3DOF Simultaneous Localisation And Mapping (SLAM) methods such as the grid-based SLAM method described in [16] or methods that use some form of scan-matching such as the Iterative Closest Point (ICP) approach [17].

Rather than using a 2D laser scanner, one could choose to use sensor that produces a 3D point cloud such as a 3D laser scanner or even a 3D imaging sensors. Examples of the latter are the Swissranger, PMD or the Kinect. In that case, features could be extracted from the resulting 3D point cloud and used for 3D pose estimation. For example, in [18] and [19] planar features are extracted from the point cloud and used to obtain 6DOF estimates either with or without IMU. Alternatively, the translational and rotational motion (6DOF) of the platform can be estimated by performing ICP on consecutive point clouds [20].

2D laser scanners can be used for 3D pose estimation, as is done in the algorithm presented in this paper. The 2D sensor could be turned into a 3D sensor by using an additional motor to rotate the laser scanner or an additional rotating mirror [20]. The method described in [21] uses the aerial platform itself to rotate the 2D laser scanner. The method furthermore assumes a structured environment and uses planar feature to estimate the 3D pose of the platform.

Recently, SLAM methods have been develop that use a 2D laser scanner on a small-size indoor aerial platform. In [22] the authors present an approach that combines a 2D SLAM

method with a 3D navigator based on an IMU. They use a 2D occupancy grid map and use a bilinear interpolator to achieve better scan matching performance. This method has been implemented in ROS as part of the Hector SLAM package. [23] describes another implementation using a modified 2D laser scanner. Like the previous paper, a 2D map is being generated, however, this time support is included for multiple floors and loop closure. ICP is used to perform scan matching with a grid-based map. A mirror is added to deflect a section of the 2D laser field-of-view (FoV) to obtain an altitude measurement(z). The remaining 2 degrees of freedom (ϕ, θ) are obtained from the IMU. The authors in [24] utilize the same modified 2D laser scanner as used in [23] and similarly, compute the pitch and roll (ϕ, θ) directly from the IMU, reducing the problem to a 4DOF incremental motion estimation problem. In contrast to [22] and [23], the authors introduce the concept of multilevel-SLAM which uses multiple 2D maps associated with potential level changes (i.e. tables) defined at discrete altitudes. The deflected laser altitude measurement is used in combination with the vertical velocity estimate are used to identify the level of each point of the point cloud.

In terms of multi-sensor frameworks the work in [25] [26] proposes a time-delayed single and multi-sensor fusion based on Extended Kalman Filter for 6DOF pose estimation including intrinsic and extrinsic sensor calibration. This filter is available in ROS in [27].

1.5.3. Aerial mapping

Aerial mapping in 2D and 3D is currently a popular research topic. The growing availability of affordable MAV systems and high-resolution cameras, in combination with the progress of aerial mapping software (i.e. Pix4D, Photoscan, etc) has enabled its application in multiple scenarios. A survey of the current status has been described in [28] and target applications ranging from forestry and agriculture, cultural heritage, environmental monitoring and 3D reconstruction are also analysed.

A standard mapping pipeline is described in [29] which relies on i) the optimized matching of pair of images pre-selected based on the telemetry of the UAV, ii) scene reconstruction based on the bundler block adjustment such as for instance the one described in [30], iii) triangulation and densification of the resulting sparse point cloud to build a DEM and iv) projection of the images to generate ortho-photos and 3D models.

MAV based mapping competes with the traditional manned flights and satellite imagery primarily in cost, but also in performance. For instance, an evaluation of the performance of Pix4D software in [31] shows a mean deviation between the Pix4D surface and the ground-truth generated with a LIDAR of about 3 cm.The results are robust to errors in the estimation of the MAV position and orientation typically cause by the considerably lower cost components integrated on the MAV autopilots. A geo-referenced ortho-mosaic and Digital Surface Model (DSM) can be obtained in principle without the need for ground control points.

However more accurate geo-referencing does require Ground Control Points.

The application of aerial mapping to search and rescue differs slightly from the other applications identified above. Requirements such as speed, integration on the coordination chain, multi-stage assessment of different levels of interest, etc. must influence the approach to produce the mapping. In [32], the requirements, adaptations and integration needs of general-purpose UAV mapping to be useful in search and rescue operations are described.

1.5.4. Victim detection

People detection and tracking has been subject to extensive research. [33] provides a summary of relevant initiatives in the fields of computer vision and pattern recognition. A classical reference is the work on [34] that combines weak features such as Haar combined by AdaBoost to provide a strong classifier. These approaches focus on body parts detection and, as supervised algorithms, require training datasets.

These works have been transferred to aerial people detection and tracking using visible-light images from MAVs. A similar approach to the work done in [34] has been used in [35], where due to the rapid platform motion, image instability and the relatively small size of the object of interest signatures, the detection requires a secondary confirmation with thermal imagery.

Tracking across multiple frames has been used to provide better performance [36]. Kalman filter can be used to predict the motion of the features as described in [36][37].

Classical methods based on colour and texture cannot be directly applied on thermal images. Thermal images have a much lower resolution and are corrupted by thermal noise. Since humans most often have a different body temperature than the surrounding background, background subtraction method offers a first and fast selection method to truncate the detector search space to image regions containing humans [38].

1.6. Methodology

In order to build upon existing body of work, the project relies, when possible, on commercial off-the-shelf (COTS) low-cost components and open-source software. In general, an iterative approach. At the beginning of the project, there was neither available functional prototypes nor a-priori knowledge on the specific algorithms. Most of the tasks followed this scheme:

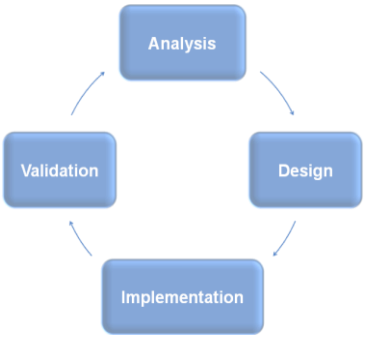


Fig. 3 Iterative model

During the analysis, information about the problem was gathered. For instance, studying the state of the art in specific algorithms such as aerial mapping or sensor-based navigation. Or surveying the market for open-source autopilots, thermal cameras or operator interfaces.

During the design phases, key decisions were made. Depending on the type of task, it could imply the evaluation of an autopilot board or a software middleware, or the actual design of the mapping algorithm. After the implementation of the design, validation will confirm if the outcomes were valid, should be discarded or could lead to corrective actions.

Following this scheme, several frames and autopilots were evaluated and key decisions throughout the process were made. More details are provided in section 2.1 and 2.2. Another example of the iterative process usage of this scheme is the design of the cameras configuration of the sensor payload. An initial study of the state of the art would hint that a nadir configuration will increased the quality of the maps, which is true for orthophotos, but during the first field trials, it was clear than a oblique camera configuration will give both better victim detections and buildings 3D reconstructions while it will also allow to detect obstacle in front of the MAV, which is a trade-off for the configuration

The MAV relies on several open-source components and tools. It is equipped with an Onboard PC running Ubuntu 14.04. The Robot Operating System (ROS) has been chosen as the onboard Software middleware for all the software developments. Most of the software nodes running onboard are freely available in ROS. Octave was use for prototyping some of the software routines as well as OpenCV has been used for computer vision. Some other open-source libraries has been used for specific functionality, for instance, SwiftNav library was used to convert from global to local coordinates, GDAL to generate raster files, etc.

The weight and power consumption of the sensors and onboard electronics is a critical parameter affecting the performance of the MAV. Not having the capacities to miniaturize the onboard components, all integration have been based on standard interfaces (USB, Ethernet, etc). The mechanical integration has used some open-source 3D models adapted to our needs and printed with the Eurecat 3D-printer even though the author has not been directly involved in the electronic and mechanical integration.

2. MAV Architecture

Ready-To-Fly (RTF) is the commercial term used to refer to completely assembled platforms that are available commercially of the shelf. They require no adaptation prior to flight. In the context of this document, RTF is used to refer to the MAV platforms that provide the basic functionality such as tele-operation, altitude control, hover and mission mode, independently of whether they were bought as a RTF system or were built and adapted.

This section proposes a conceptual design that provides a RTF MAV with the required capacities to address the abovementioned challenges, including hardware and software. It also describes a payload specifically conceived for Search and rescue MAV missions and proposes a software architecture and an approach to assure safe and reliable navigation.

2.1. MAV platform

The MAV platform used in this project is an outdoor coaxial quadrotor. An electric Mini-UAV with Vertical Take Off and Landing (VTOL) developed at Eurecat along with all the other activities referred in this project. The current platform is called LIFT IV and is an adaptation of previous designs that has been modified to satisfy the needs of Search and rescue operations. LIFT IV has a 855 mm diameter and weights 4.3 Kg. The propulsion chain has been optimized to maximize the payload capacity (3.8 Kg), endurance (35 minutes approx.) and wind resistance (up to 35 Km/h), allowing it to support different scenarios and configurations. These modifications consist of:

- the upgrade of the avionics (see 2.2),
- integration of the Search and rescue payload concept (see □),
- integration of a survival kit delivery mechanism,
- water proofing.

Fig. 4 Search and rescue MAV (Eurecat-Ascamm)

2.2. MAV avionics

2.2.1. Autopilot

Several autopilots were evaluated throughout the project. The first attempt was to develop our own autopilot and was quickly discarded when open-source autopilot started to become widely available and adopted.

The first choice was the UAV development board (UDB4) [39]. The UDB4 comes populated with a dsPIC33FJ256 CPU, and the MPU-6000, a MEMS 3-axis gyroscope and 3-axis accelerometer which is the exact same architecture and sensors selection of our own autopilot. Unfortunately, the rest of the required sensors (barometer and magnetometer) had to be integrated. *MatrixPilot* is the open-source firmware which was designed for planes, and had to be adapted to multi-rotors.

Under the incredible explosion of use, performance and reliability of the ArduCopter community, the autopilot was upgraded to APM 2.5 [40]. Some experiments helped us understand that the limited hardware resources in this board could lead to catastrophic incidents and therefore the system was upgraded to the Pixhawk autopilot board [41] but Arducopter 3.2.1 [42] remains as the firmware of choice due to their advanced user functionality and to ensure compatibility with some of the previous developments. Pixhawk integrates a 32bit STM32F427 Cortex M4 core with FPU, ST Micro L3GD20H 16 bit gyroscope, ST Micro LSM303D 14 bit accelerometer / magnetometer, the Invensense MPU 6000 3-axis accelerometer/gyroscope and MEAS MS5611 barometer. All interfaces and connectors are fully integrated and available off-the-shelf.

2.2.2. MAVLINK interface protocol

Most open-source autopilots implement the MAVLink protocol [43] for the interface with any external subsystem. MAVLink or Micro Air Vehicle Link is a protocol for communicating with small unmanned vehicle. It is designed as a header-only message marshaling library. MAVLink was first released early 2009 by Lorenz Meier under LGPL license.

The autopilot has been modified to enable a second telemetry link. The main telemetry link connects the autopilot with the GCS, and therefore the operator. This link usually builds upon wireless systems such as the Xbee family that uses IEEE 802.15.4 networking protocol for fast peer-to-peer networking. A secondary telemetry link allows the access to the autopilot from an onboard computer, called *Companion computer* in the Pixhawk argot. This link builds upon a serial interface or serial-over-USB. Both these links are running MAVLink.

The bridge between MAVLink and ROS is the *mavros* node (http://wiki.ros.org/mavros). The

list of MAVLink message definitions is fully documented in [43]. They can be grouped in the following categories:

- Minimum messages set: the minimum set of messages required from the autopilot (heartbeat, attitude, position, battery, etc) and the GCS.
- Parameter protocol defines the messages used to change the configurations.
- Mission protocol defines the messages required to plan and execute a flight plan.
- Robotics/companion protocol defines the messages that are available to the onboard computers in order to take control. For instance:
    - COMMAND_INT to send commands to the MAV
    - SET_ATTITUDE_TARGET sets the vehicle attitude and body angular rates
    - SET_POSITION_TARGET_LOCAL_NED sets vehicle position, velocity and acceleration setpoint in local frame.
    - SET_POSITION_TARGET_GLOBAL_INT sets vehicle position, velocity and acceleration setpoint in the WGS84 coordinate system

Some other relevant MAVLink messages used in this project are:

- RC_CHANNELS_OVERRIDE to override the safety pilot RC radio from a gamepad or joystick
- GLOBAL_VISION_POSITION_ESTIMATE and VISION_SPEED_ESTIMATE to push an external estimate of the current position

2.3. MAV Copilot

A *copilot* is the second pilot in an aircraft. The *copilot* increases safety and executes the instructions delivered by the official pilot, who supervises the operation and keeps the ultimate responsibility. This concept applies also to autonomous systems. It has been used to name speed controllers and GPS navigation systems for cars. SeeByte Ltd. calls one of their product SeeTrack CoPilot [44], a Dynamic Positioning system for Remote Operated underwater Vehicles (ROV).

This section proposes a MAV Copilot, a payload that upgrades any commercial RTF multi-rotor to a fully autonomous intelligent MAV. The Copilot includes an On-board PC, interfaces to the autopilot and the operator on the ground and a set of integrated sensors, depending on the target application. It may provide the following functionality:

- Assured navigation based on multi-sensor data fusion
- Thermal and visible digital downlink
- Obstacle detection and avoidance
- Target tracking
- Autonomous infrastructure inspection
- Fast mapping (close to real time)

- 3D reconstruction
- Fleet Collaboration

The following figures show the prototype developed at Eurecat and used in this Master's thesis:

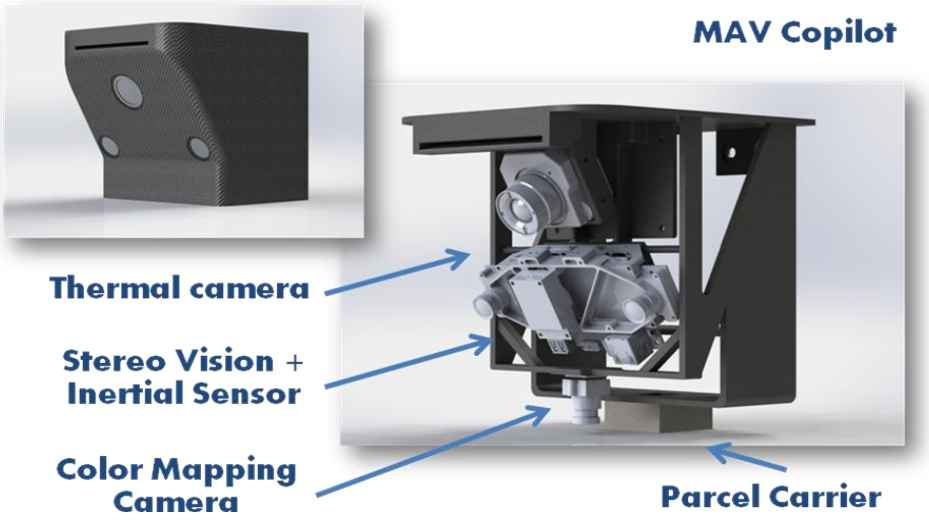


Fig. 5 Search and rescue Copilot Concept

Fig. 6 Prototype of the MAV Copilot

2.3.1. Sensor selection and configuration

The payload kit integrates several sensors.

It integrates a Vision-Inertial sensor [45], developed by ETH Zurich and Skybotix AG, to enable vision based navigation and mapping. This sensor provides fully time-synchronized and factory calibrated IMU- and stereo-camera data streams. The stereo pair is composed by two Aptina's MT9V034, a 752H x 480V pixels camera. The IMU is an ADIS16448 from

Analogue Devices. Augmenting with inertial sensors enhances the performance and limitations of algorithms based on pure vision.

To use stereo cameras instead of lasers is a design decision: they are lighter, provide more information, and under certain distance and circumstances can also provide range estimations to potential obstacles.

Fig. 7 Vision-Inertial Sensor [45]

The MAV Copilot also integrates a thermal imager, FLIR Tau2 [46]. It is used for mapping, real-time victim detection and to support night operations. The FLIR Tau2 is a Long Wave Infrared (LWIR) Thermal Imaging Camera. LWIR sensors measure thermal emissions and therefore do not require external lighting. The FLIR Tau2 provides 640 × 512 images at 30Hz. For multirotor applications, flying at low altitude and in proximity to the targets, wide angle lenses are better suited. The 9mm lenses have been mounted providing a Field of View (FOV) of 69° (horizontal) × 56° (vertical). This camera enabling tracking during night and in other situations where the image quality of visible-light cameras is limited.

Fig. 8 FLIR Tau2 LWIR Sensor [46]

The current orientation of these sensors is oblique, with 60º pitch up from the vertical axis. This is the optimum setup since it enables navigation, obstacle detection and mapping. Furthermore, for victim detection it provides a much better perspective than others angles.

An extra color camera is mounted in nadir configuration (pointing downwards) to support mapping in color. It is a Pointgrey Firefly FMVU-03MTM-CS with 752 x 480 pixels.

Fig. 9 Pointgrey Firefly FMVU-03MTM-CS

All these images are captured from the onboard computer and geotagged with the current position and attitude estimation.

This payload is sometimes complemented by high-resolution cameras to provide higher fidelity maps. A compact Canon IXUS 130 digital camera with 14.1 M provides conventional RGB images. A Sony α4000 with 24 megapixels has also been used in some experiments. These cameras can be mounted in two different configurations: downwards looking for aerial mapping, and oblique (45 degrees) for better buildings 3D reconstruction. The images from this camera are stored on the camera and only post-processed after landing.

2.3.2. Camera calibration

The intrinsic and extrinsic calibration of the cameras and the IMU is done with the Kalibr toolbox [47][48] developed by ETH Zurich. Kalibr is a toolbox that solves the spatial and temporal calibration of an IMU with respect to a camera system. This section lists the calibration parameters from each of the cameras.

2.3.2.1. Intrinsic camera calibration

The intrinsic calibration computes the focal length and principal point and gets represented as the intrinsic camera matrix *K:*

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(Eq. 2.1)}$$

The intrinsic calibration matrix transforms 3D camera coordinates to 2D homogeneous image coordinates. This perspective projection is modeled by the ideal pinhole camera. The focal length (f_x, f_y) is the distance between the pinhole and the image plane and is measured in pixels. The principal point (c_x, c_y) represents the center of the image.

2.3.2.2. Extrinsic camera calibration

The extrinsic calibration computes the coordinate system transformations from 3D world coordinates to 3D camera coordinates. In our case, the extrinsic parameters define the mapping of a point in camera frame into the IMU frame, expressed as a rotation and a translation. It is represented as the extrinsic camera matrix:

$$[R \mid t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \mid t_1 \\ r_{21} & r_{22} & r_{23} \mid t_2 \\ r_{31} & r_{32} & r_{33} \mid t_3 \end{bmatrix} \quad \text{(Eq. 2.2)}$$

2.3.2.3. Camera calibrations

Cam0 (Left camera on the stereo pair)

Full resolution in pixels: 752(w) X 480 (h).

$$K_{cam0} = \begin{bmatrix} 467.27 & 0 & 377.52 \\ 0 & 465.88 & 239.07 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{cam0}^{imu} = \begin{bmatrix} 0.064 \\ -0.008 \\ 0.005 \end{bmatrix} \quad \text{(Eq. 2.3)}$$

$$q_{cam0}^{imu} = [0.0004 \quad -0.0009 \quad -0.7081 \quad 0.7060]$$

Cam1 (Right camera on the stereo pair)

Full resolution in pixels: 752(w) X 480 (h).

$$K_{cam1} = \begin{bmatrix} 468.08 & 0 & 319.94 \\ 0 & 466.84 & 247.21 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{cam1}^{imu} = \begin{bmatrix} -0.0438 \\ -0.0078 \\ 0.0094 \end{bmatrix} \quad \text{(Eq. 2.4)}$$

$$q_{cam1}^{imu} = [-0.0021 \quad 0.0028 \quad -0.7101 \quad 0.7040]$$

Tau3

Full resolution in pixels: 640(w) X 512 (h). Calibrating a thermal camera is not straight forward. There are several approaches. A special calibration board is needed. In our case, we used a board heated up by a lamp that was not continuously picked up by calibration toolbox. It could correctly calibrate the intrinsic, but it didn't get enough lock to calibrate the extrinsic. A future work to do is to repeat the calibration step using a special calibration board with enough reflective material to be visible on the thermal.

$$K_{tau} = \begin{bmatrix} 523.91 & 0 & 343.36 \\ 0 & 522.38 & 262.27 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(Eq. 2.5)}$$

Mapping

Full resolution in pixels: 640(w) X 480 (h). The mapping camera was mounting downwards in NADIR configuration during the euRathlon competition 6.3, being already on the field. Since time was limited and the transformation from the camera to the IMU was not critical for this camera, only the intrinsic parameters were calibrated for this camera.

$$K_{mapping} = \begin{bmatrix} 631.06 & 0 & 289.03 \\ 0 & 629.19 & 236.63 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(Eq. 2.6)}$$

2.4. Software architecture

Autonomous systems are highly complex. They need to execute several tasks concurrently, while monitoring and managing unexpected events. For instance, sensors are not perfect and they induce noise, and therefore errors, in the measurements. Thus, perception and estimation are subject to uncertainty. Writing software for robots is difficult, particularly as the scale and scope of robotics continues to grow. All this complexity leads to the need of well-designed robotics architectures. The term robot architecture is often used to refer to how a system is divided into subsystems and how those subsystems interact. It can also refer to the computational concepts that underlie a given system, for instance, the communication method, the event modeling, etc.

Different types of robots can have quite varying hardware, making code reuse nontrivial. The need for common robot architectures and frameworks to share and reuse software and algorithms seems convenient. The *Handbook of Robotics* [49] refers to some reference architectures and principles that serve as inspiration.

Even though this project has focused on the Search and rescue domain, the challenge of the

architecture design is to ensure abstraction and versatility for future applications. The following diagram shows the robotics software architecture defined at the beginning of this project. It is a three-tier architecture. This clear separation in layers encourages different levels of abstractions in the design. Some principles behind this architecture:

- The functional layer provides abstract interfaces to the robot basic functionality such as motion control, state estimation and navigation. This layer interacts with the hardware drivers.
- The executive layer implements the plan received by higher layers that could come directly from the operator. It is a logical layer in charge for instance of consuming the set of waypoints that compose a flight plan or executing a given behaviour, for instance, inspect traverse, execute flight plan, inspect this wall, etc.
- The deliberative layer is the brain of the system. It is responsible to interpret the world and select the next behavioral strategy or plan.

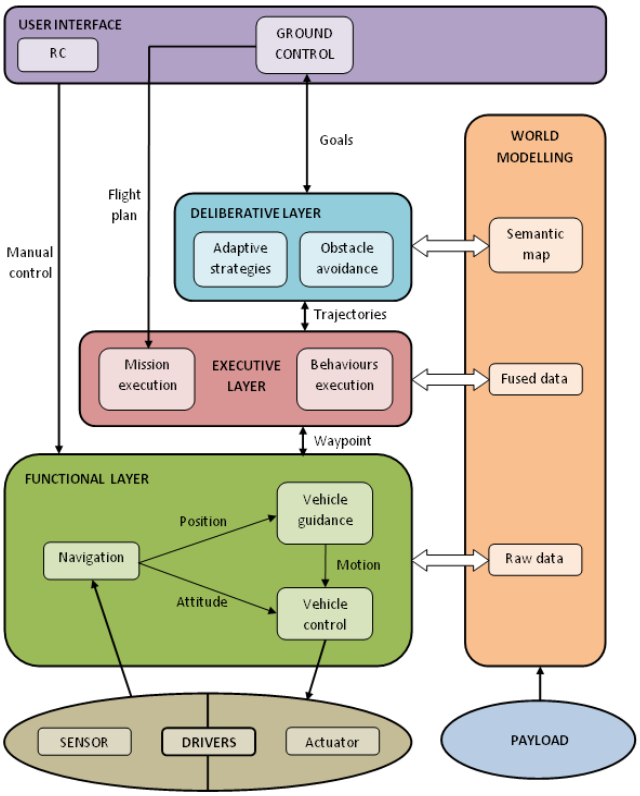


Fig. 10 Conceptual Reference Software Architecture

In the case of a MAV, the functional layer is typically provided by the autopilot. Even the executive layer is also partially provided by the autopilot, at least the functionality not related to perception, just pure mission execution. The Pixhawk autopilot provides access at different control levels. This flexibility allows multiple strategies to implement the concepts drawn above.

All the software running on the onboard computer is based on ROS [50]. ROS is an open source robot operating system. It provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license.

Some of the concepts proposed in the reference architecture, such as abstraction, encapsulation, etc., are already implicitly provided by ROS. The following diagram shows the deployment of our onboard ROS-based software:

- Hardware drivers are responsible for accessing the sensors and autopilot and provide standard ROS messages to the rest of the System
- The Assured Navigation provides multi-sensor pose estimation and obstacle detection.
- The data captured in ROS is available on the GCS through the WiFi link and serves to augment the information displayed to the operator in RVIZ.
- Specific functionality such as victim detection and aerial mapping

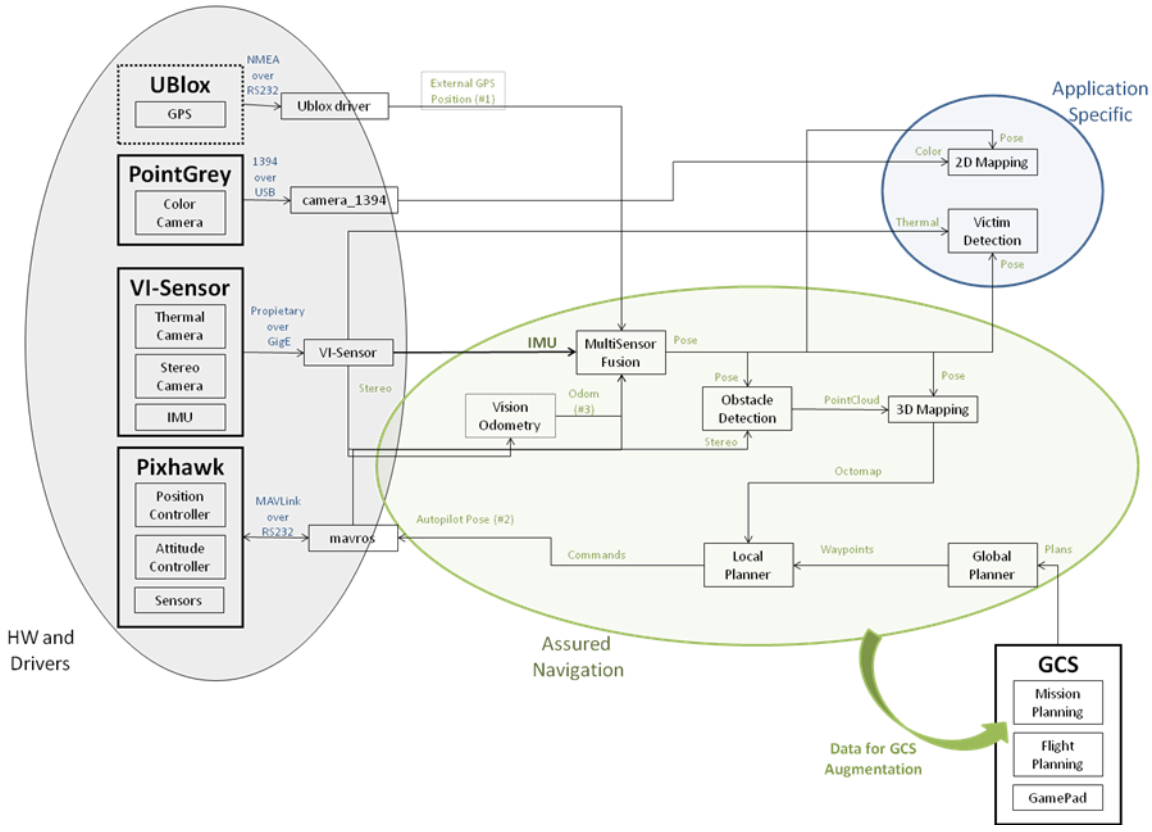


Fig. 11 Conceptual Deployment Diagram of the ROS nodes

2.4.1. Interoperability

Interoperability may be defined as the ability of robots to operate in synergy towards the execution of assigned missions. Having a multitude of robotic systems able to assist with disaster management operations is potentially very useful, but if the interoperability between these heterogeneous agents is missing, then it will not help much in practical operational situations, where these robotic assets need to collaborate and share information.

To ensure interoperability, this project proposes the standardization of the interfaces. Interoperability standards provide a common framework for multi-robot missions. There exist several predominant initiatives for interoperability of unmanned systems. Harmonization among them is not yet a fact.

To make matters worse, each autopilot manufacturer will choose their own software interface. MAVLink (see 2.2.2) is widely adopted in the open-source community related to MAV developments. The system is automatically compliant with most of the open GCS (i.e. QGroundControl, Mission Planner, APM planner, etc). Since the Copilot is using ROS, the system will be automatically compliant with any ROS-based GCS.

However, MAVLink is originally design for MAVs and ROS is not considered a standard (rather a middleware). If we consider the cooperation with other domains, such as maritime and ground robots, an alternative standard for interoperability is the Joint Architecture for Unmanned Systems (JAUS). A ROS-JAUS bridge has been implemented and enables this MAV Copilot to be interface from any JAUS compliant GCS.

3. Seamless Multi-Sensor Navigation

The work described in this section has been published as:

- "*Seamless Indoor-Outdoor Navigation for Unmanned Multi-Sensor Aerial Platforms*" published on the journal The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-3/W1 , 2014 [51]
- and a different paper with the same title "*Seamless Indoor-Outdoor Navigation for Unmanned Multi-Sensor Aerial Platforms*" published on the Position, Location and Navigation Symposium - PLANS 2014, 2014 IEEE/ION [52].

This section proposes an approach to provide a MAV with robust multi-sensor localization. This algorithm is capable of using all or a subset of its sensor inputs to determine a platform's position, velocity and attitude (PVA). To enable safe and reliable operation of MAVs in search and rescue, at any time in any environment, a position and velocity estimation is required that is robust and not solely dependent on the Global Navigation Satellite System (GNSS). Particularly in Urban Search and Rescue (USAR), when the MAV is required to operate in urban canyons, close to structures, a GNSS position capability may not be available due to shadowing, significant signal attenuation and multipath caused by buildings, or due to interference, denial or deception. To improve availability and guarantee continuity of service in these environments, GNSS is typically integrated with an Inertial Measurement Unit (IMU) and several other type of sensors. Note that integration of multiple sources of data may not only improve the accuracy of the position and attitude estimate, but also add integrity, continuity and availability to the solution. [53]

In the integration approach proposed here, an Inertial Measurement Unit (IMU) is chosen as the core sensor since it is self-contained and, therefore, does not depend on “external” infrastructure such as a feature-rich environment, or radio-frequency (RF) signals. However, when an Inertial Navigation System (INS) is used in a standalone manner the position and attitude estimates will drift over time. Integration with secondary sensors is, therefore, required to mitigate the drift error by periodically “resetting” the inertial errors [53].

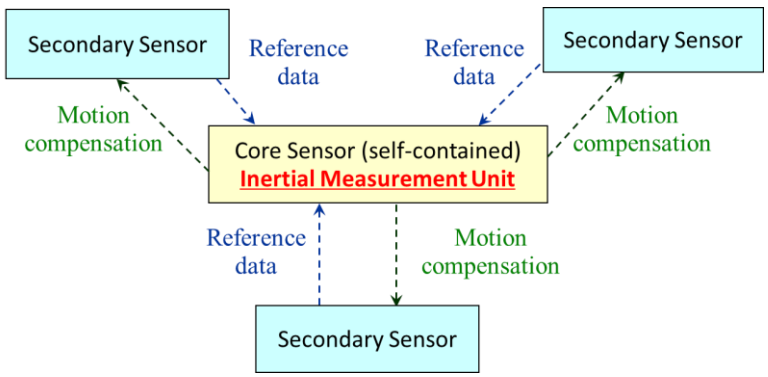


Fig. 12 Strategy for assured positioning

The system input set may include data from GNSS receivers, data from inertial sensors, grayscale and color cameras, and laser range scanners. However, at any time during operation one or more sensors can be removed (unplugged) or added (plugged in) without interruption of the estimation as long as the remaining sensors can achieve enough observability allowing the operator to change the sensor configuration during operation.

3.1. Multi-Sensor Integration Architecture

The current proposed architecture for the plug-and-play sensors integrator consists of five major components: the connection monitor, the calibration unit, the feature extraction unit, the PVA estimator and mapping unit, and the integrity monitor. The architecture is shown here:

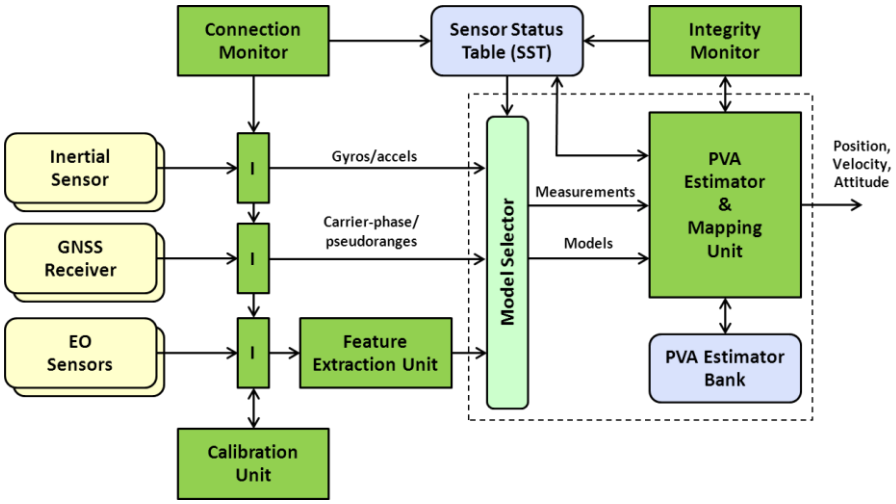


Fig. 13 Reconfigurable navigation solution architecture

The connection monitor detects sensor connect and disconnect events and keeps track of the current status of each of the devices in a Sensor Status Table (SST). In case of a connect event, the monitor determines if the sensor in question requires online calibration and, if so, dispatches the sensor device to the calibration unit for calibration. An online calibration of intrinsic and extrinsic parameters may be required for sensors such as the 2D imaging sensors. In the event that no calibration is required or the calibration has been completed, a feature extraction unit is assigned to those sensors that require one, and the sensor is connected to the PVA estimator and mapping unit. In case of a disconnect event the device will be removed from the SST.

The feature extraction unit extracts the necessary features from the sensors such as planar surfaces, lines and points from 2D imagery and Ladar. Finally, the integrity monitor detects if any off-nominal condition does exist in the available sensor data and passes that information to the connection monitor, which can then remove it from the SST temporarily or permanently.

3.2. PVA estimator

The PVA estimator (and mapping) unit uses the Sensor Status Table to determine the structure of the PVA estimator that should be used for the current sensor configuration, and performs an observability analysis to determine if a PVA estimate can still be obtained with that configuration. This unit furthermore, performs a smooth transition from one filter to the next using a filter structure similar to an Interacting Multiple Modeling (IMM) filter.

Currently, various estimator strategies are used in the filter structure to support the various sensor configurations including complementary extended Kalman filters and Self-Localization and Mapping (SLAM) algorithms.

The following figure shows a detailed block diagram for the proposed PVA estimator and mapping unit.

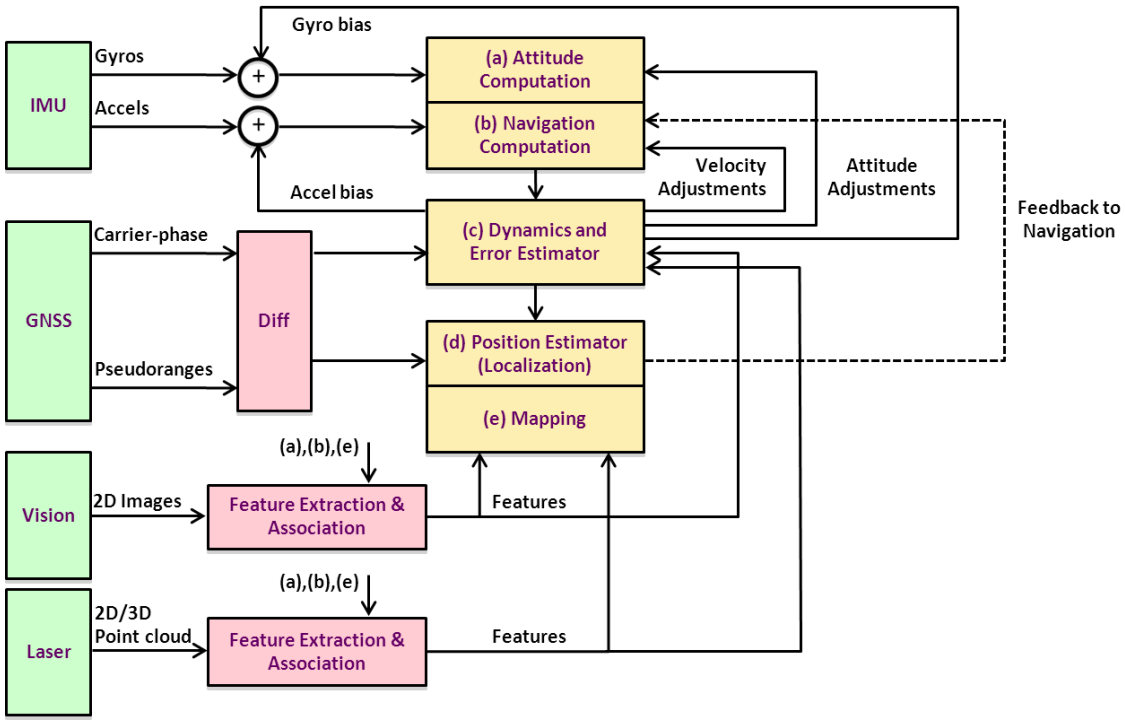


Fig. 14 PVA Estimator block diagram

The top part, blocks (a) and (b), show the inertial mechanization. The shown mechanization implements the low-cost inertial mechanization described in detail in references [54]. The mechanization is split into two parts: the computation of the attitude and the computation of position and velocity in the navigation frame.

The block (c) illustrates the estimation of the dynamics of the system. It estimates the acceleration, velocity and orientation of the system. Furthermore, it is also responsible to estimate the acceleration bias, and velocity and attitude errors that are used to correct the

current inertial estimation. Block (d) represents the estimation the position. Optionally, block (e) is responsible to do mapping.

The estimation strategy used in blocks (c) and (d) depends on the current SST. The following sections 3.4, 0 and 3.8 show examples of the most common sensors used to complement the inertial navigation, illustrating a subset of complementary filters that can be used within our framework. It is important to note that the PVA estimator can still be configured in various ways. In one approach the H matrices derived from all available features are used in one big complementary Kalman filter. An alternative approach would be to use an IMM that has separate filters for each of the secondary sensors and combines the separate filters using a Markov model. A more detailed discussion of these various filters and filter configuration as well as a completely difference approach using a Marginalized SLAM approach will be the focus of a future paper.

3.3. Strapdown Inertial Navigation

An Inertial Measurement Unit (IMU) has three integrated gyroscopes that measure the angular rate $\boldsymbol{\omega}_{ib}^{b}$. and three accelerometers that measure the specific forces $\mathbf{f}^{b}$, the non-gravitational accelerations.

Integrating these measurements we obtain the change on angle and velocity:

$$\Delta\vartheta = \int_{t-\Delta t}^{t} \boldsymbol{\omega}_{ib}^{b}\, dt \qquad \text{(Eq. 3.1)}$$

$$\Delta v = \int_{t-\Delta t}^{t} \mathbf{f}^{b}\, dt \qquad \text{(Eq. 3.2)}$$

In an inertial navigation system, we want to compute the current velocity and position of the sensor in our navigation frame. Therefore, the strapdown navigation needs to compute the rotations and compensations from factors such as the previous known orientation, the Coriolis acceleration, the gravity and centripetal acceleration, etc. All this computation is summarized in the following diagram:

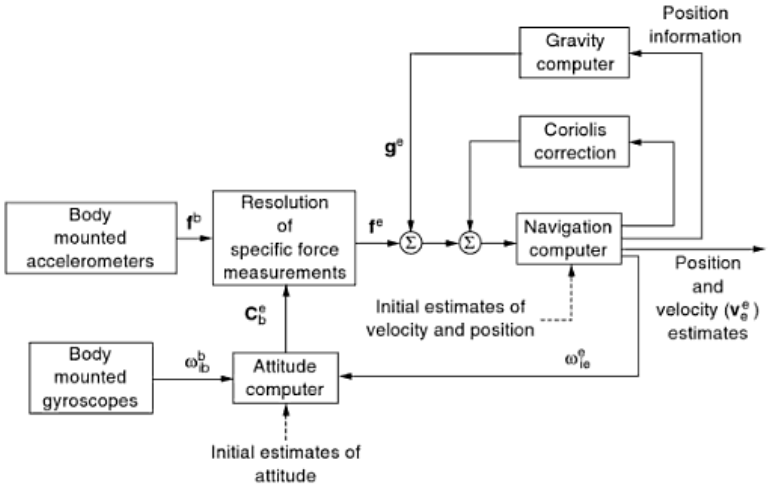


Fig. 15 Strapdown Inertial Mechanization [54]

The current estimate of the orientation, velocity and position is calculated by iteratively integrating the changes in angle, accelerations and integrated velocities. An inertial system is therefore a dead reckoning system. The main disadvantage is that any error on the measurement is integrated on the estimation and therefore the uncertainty of grows over time as the error is accumulated.

Therefore, a system relying only on inertial will show drift. These are the identified errors on a strapdown inertial mechanization:

- $\delta\mathbf{r}$: position displacement error,
- $\delta\mathbf{v}$: velocity error,
- $\delta\boldsymbol{\Psi}$**:** tilt or miss-orientation
- $\delta\boldsymbol{\omega}$: gyro bias, and
- $\delta\mathbf{f}$: accelerometer bias,.

To remove these errors, they have to be estimated with the support of a secondary sensor. They can be related to the input variables of the INS and can be taken out if their values can be estimated.

3.4. Extended Kalman Filter (EKF)

Kalman filter is a recursive state estimator for partially observed non-stationary stochastic processes. It gives an optimal estimate in the least squares sense of the actual value of a state vector from noisy observations [56].

The linear Kalman filter considers the system a discrete-time linear system with process and sensor zero-mean white Gaussian noises.

Kalman filter models the system with equations. The general system model equation estimates the state vector, x_k, and is shown in equation 3.3. The general measurement

model equation, displayed in equation 3.4, is used to estimate the measurement vector, z_k.

$$\boldsymbol{x_k} = \boldsymbol{\Phi_{k-1} x_{k-1} + w_k} \quad \textit{(Eq. 3.3)}$$

$$\boldsymbol{z_k} = \boldsymbol{H_k x_k + v_k} \quad \textit{(Eq. 3.4)}$$

The process noise, w_k, the error variance, e_k, and the measurement error variance, v_k, must be estimated and are assumed to be Gaussian due to the central limit theorem [55].

The measurement matrix, $\boldsymbol{H_k}$, and the system dynamics matrix, $\boldsymbol{F}$, are designed in the following sections and define the filter structure to be used in the different scenarios. Using these variables, an initial measurement covariance matrix, $\boldsymbol{R_k}$, an initial error covariance matrix, $\boldsymbol{P_k}$, a transition matrix, $\boldsymbol{\Phi}$, and a system noise covariance matrix, $\boldsymbol{Q}$, can be obtained:

$$\boldsymbol{R_k} = E\left[v_k^- v_k^{-T}\right] \quad \textit{(Eq. 3.5)}$$

$$\boldsymbol{P_k} = E\left[e_k^- e_k^{-T}\right] \quad \textit{(Eq. 3.6)}$$

$$\boldsymbol{\Phi_k} = exp^{\boldsymbol{F}\Delta t} \quad \textit{(Eq. 3.7)}$$

$$\boldsymbol{Q} = E\left[w_k^- w_k^{-T}\right] \quad \textit{(Eq. 3.8)}$$

The Kalman filter follows an iterative process. Given the initial conditions X_0 and P_0, the complete recursion in the filter is compute as follows:

Prediction

- Predict the a priori next state vector based on the current estimated state and the calculated transition matrix:

$$\boldsymbol{\hat{x}_{k+1}^-} = \boldsymbol{\Phi_k \hat{x}_k} \quad \textit{(Eq. 3.9)}$$

- Projects the error covariance matrix forward:

$$\boldsymbol{P_{k+1}^-} = \boldsymbol{\Phi_k P_k \Phi_k^T} + \boldsymbol{Q} \quad \textit{(Eq. 3.10)}$$

- Predict the observation estimates:

$$\boldsymbol{\hat{z}_{k+1}^-} = \boldsymbol{H \hat{x}_{k+1}^-} \quad \textit{(Eq. 3.11)}$$

Update

- The Kalman gain K is computed on each iteration with the following equation:

$$\boldsymbol{K_k}=\boldsymbol{P_k^-}\boldsymbol{H_k^t}(\boldsymbol{H_k}\boldsymbol{P_k^-}\boldsymbol{H_k^T}+\boldsymbol{R})^{-1} \quad \textit{(Eq. 3.12)}$$

- The state vector is updated using the current measurements:

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}}=\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}+\boldsymbol{K_k}(\boldsymbol{z_k}-\hat{\boldsymbol{z}}_{\boldsymbol{k}}^{-}) \quad \textit{(Eq. 3.13)}$$

- The error covariance matrix must also be updated in this process:

$$\boldsymbol{P_k}=(\boldsymbol{I}-\boldsymbol{K_k}\boldsymbol{H})\boldsymbol{P_k^-} \quad \textit{(Eq. 3.14)}$$

For systems that are considered non-linear, the Extended Kalman Filter (EKF) provides a solution by linearizing the process about the current state, and linearizing the measurement model about the predicted observation. In our navigation solution, we assume that the system is non linear, therefore, we use an EKF. It is important to note that the linearization of the nonlinear process and measurement model may be only reliable for systems that are almost linear on the time scale interval.

In our navigation solution, we assume that the system is non linear, therefore, we use an EKF. We assume that both process models in equations 3.3 and 3.4 remain valid. The modeling process is done using all of the same equations and methods except for the derivation of the geometry matrix[55]**.** **H** is no longer a constant term, instead, it is derived using equation (3.15). In this equation the Jacobian of the observation matrix ,**h**, is taken with respect to the state vector. The iterative process remains the same, but using the continuously updated geometry matrix, the rest of the update and prediction equations remain unchanged:

$$\boldsymbol{H}_k=\frac{\partial \boldsymbol{h_k}}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}} \quad \textit{(Eq. 3.15)}$$

If sensor readings are very uncertain, and the state estimate is relatively precise, then the Kalman gain has little impact on the update of the state estimate in Eq. 6, and the system relies heavily on the system model. If, on the other hand, the uncertainty in the measurement is small and that in the state estimate is large, then K is also large, thus trusting more in sensor measurements for the correction of the state estimate. Thus, the Kalman filter is typically used to fused data from two complementary sensors, typically inertial measurements which are accurate in short term but drift, with other type of signals like GNSS which are more uncertain but do not degrade over time.

3.5. Complementary EKF implementation

Based on the generic description of a EKF provided in the previous section, this section illustrates how to implement a EKF for our particular navigation problem. The remaining work is to defined the state vector, the measurements vector and observation equations.

The block diagram in Fig. 14 shows a separated approach in which the dynamics and position components are separated in a manner similar to the one described in [12]. In our case, the dynamics and position estimator, and therefore, the state vector includes also information about the position. They take the form of a complementary Kalman filter (CKF). A complementary filter takes error measurements as input and estimates error state vector rather than absolute states:

$$x = \left[\delta\mathbf{r}^n \quad \delta\mathbf{v}^n \quad \delta\boldsymbol{\psi}_{nb} \quad \delta\boldsymbol{\omega}_{ib}^b \quad \delta\mathbf{f}_b\right]^T \qquad \textit{(Eq. 3.16)}$$

where

δr^n is the position error in the NED frame,

$\delta\mathbf{v}^n$ is the velocity error in the NED frame,

$\delta\boldsymbol{\psi}_{nb}$ is the is the miss-orientation attitude vector,

$\delta\boldsymbol{\omega}_{ib}^b$ is the gyro bias vector,

$\delta\mathbf{f}_b$ is the specific force bias error.

The rest of the thesis will use the augmented state vector, containing the error in the position. The corresponding continuous-time state transition matrix:

$$F = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\lfloor \mathbf{f^n} \times \rfloor & \mathbf{0} & -\mathbf{C_b^n} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{C_b^n} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\tau}_\mathbf{g}^{-1}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\tau}_\mathbf{g}^{-1}\mathbf{I} \end{bmatrix}^T \qquad \textit{(Eq. 3.17)}$$

After the CKF update step, the dynamic filter error states are fed back to the attitude and navigation computation blocks to correct the drift.

The measurement vector and observation equation varies depending on the secondary sensor chosen and the integration strategy. The different combinations with secondary sensors are described in the following sections. The process noise, w_k, the error variance, e_k, and the measurement error variance, v_k, must be estimated as part of the filter design and are assumed to be Gaussian [55].

There exist three different strategies to the integration:

- Loose integration is commonly performed in the position domain. It use processed data rather than raw measurements.
- Tight integration is commonly performed in the measurement domain
- In ultra-tight or deep integration the inertial is used to aid the GPS tracking loops.

For instance, in a GPS/ Inertial loose integration, the position estimates from a GPS receiver and from the INS algorithm are computed independently and later integrated by the filter. As long as GPS has a position solution this integration is very practical and effective, but it will coast on inertial as soon as the GPS solution drops below 4 satellites.

In GPS/Inertial tight integration, raw measurements from a GPS are integrated with position estimates or measurements from the inertial. In this scheme, the filter still works when an incomplete GPS solution. This reasoning applies also to other sources for correction such as cameras, lasers, etc. Theoretically, a well designed filter could combine sparse features from the sensors with few satellites and the inertial and provide a complete solution in a situation where none of the algorithms using a single sensor such as visual odometry, SLAM, etc will succeed.

Even thought the loose integration is simpler and will work in most of the scenarios, this thesis will delve into the tight integration scheme. For each of the secondary sensors in mind, the following sections describes the features that could be extracted, the observability provided, and how to relate them to the state vector. There are essentially two types of features [55]:

- Features providing full observability can be used directly to estimate errors on the inertial estimates using the difference between observed and synthesized features. Examples are GNSS pseudoranges, planar surfaces and 3D points and corners. Examples are provided in sections 3.6, 0 and 3.8.

- Other features such as 3D lines or 2D points are used to develop constraints equations that can be used as well to estimate errors on the inertial sensors. An example is provided in section 3.9.

3.6. GNSS receiver as secondary sensor

In the presence of a GNSS receiver as a secondary sensor, the tight GNSS/Inertial integration strategy allows estimation of inertial errors even in the presence of less than four satellites. The latter operational scenario must be expected in challenging environments like urban canyons. Various integration methods exist for GNSS and inertial sensors:

- GPS pseudoranges with inertial position

- GPS pseudoranges/carrier phases with inertial position
- GPS pseudoranges with inertial Δv, $\Delta\theta$ measurements
- GPS pseudoranges/carrier phases with inertial Δv, $\Delta\theta$

The fourth approach has been implemented within our framework to estimate the inertial errors. In that implementation the carrier-phase measurements are combined with the inertial measurements to obtain an estimate of the dynamics of the system while pseudoranges are used to help estimating the position. Pseudoranges are more robust, provide an absolute measurement but are less accurate. Carrier phases are known to be more accurate, but less robust. Therefore, this strategy exploits the benefits of both.

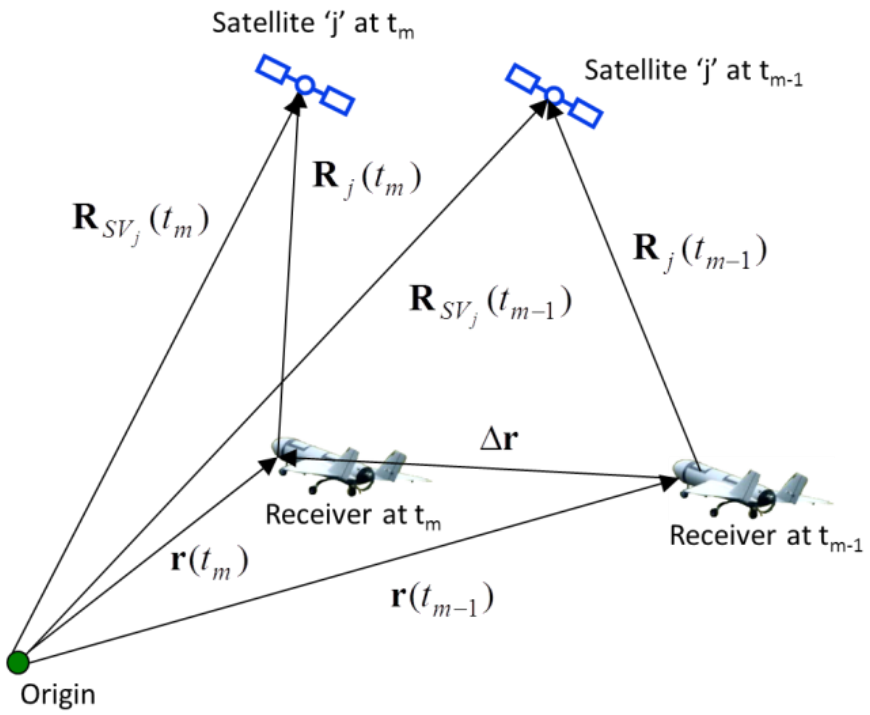


Fig. 16 GNSS Sequential difference geometry

Also, a clock drift term could be added. As shown in Fig. 14, the error states are fed back to the attitude and navigation computation blocks as soon as they are available.

The dynamics part of the filter filter in Fig. 14 is set up as a complementary Kalman filter with a measurement vector consisting of:

$$z_{j,GNSS} = \mathbf{h}_j \Delta\mathbf{r}^n_{INS} - \Delta\phi_j + a_j + b_j + \delta t_{rcvr} \qquad \textit{(Eq. 3.18)}$$

where $\Delta\phi_j$ is the sequential carrier-phase difference for satellite 'j', $\Delta\mathbf{r}^n_{INS}$ is the change in user position as computed by the inertial, $\mathbf{h}_j$ consists of the transpose of the line-of-sight

vector to satellite 'j', $\mathbf{h}_j \equiv \mathbf{e}_j^T = \mathbf{R}_j^T / |\mathbf{R}_j|$, a_j, b_j are two compensation terms for geometry and Doppler change correspondingly, δt_{rcvr} is the sequential clock drift error.

Note that equation (3.19) can be extended to a difference-of-difference measurement by taking the difference of the sequential difference of satellite 'j' and the sequential difference of a key satellite 'k'. This new difference term will remove the receiver clock drift error from the state vector.

The measurement matrix, **H**, which relates the error state in (3.16) to equation (3.19) can be derived by expanding and evaluating the sensitivity of the inertial term in (3.19):

$$\mathbf{h}_j(t_m)\Delta\tilde{\mathbf{r}}_{INS} = \mathbf{h}_j(t_m)\int_{t_{m-1}}^{t_m} \tilde{\mathbf{v}}_e^n dt + \mathbf{h}_j(t_m)\left[\tilde{\mathbf{C}}_b^n(t_m) - \tilde{\mathbf{C}}_b^n(t_{m-1})\right]\mathbf{d} \qquad \textit{(Eq. 3.19)}$$

where **d** is the lever arm between the IMU and the GNSS receiver. In equation (3.19) the tilde '~' indicates parameters affected by the inertial errors. Evaluation of (3.20) results in the following measurement matrix:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1(t_m)\left[\mathbf{I}_{3x3} \mid \mathbf{0}_{3x12}\right]\left[\int_{t_{m-}}^{t_m} \mathbf{\Phi}(\tau, t_{m-1})d\tau\right]\mathbf{\Phi}^{-1}(t_m, t_{m-1}) \\ \vdots \\ \mathbf{h}_N(t_m)\left[\mathbf{I}_{3x3} \mid \mathbf{0}_{3x12}\right]\left[\int_{t_{m-}}^{t_m} \mathbf{\Phi}(\tau, t_{m-1})d\tau\right]\mathbf{\Phi}^{-1}(t_m, t_{m-1}) \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3x3} & \left[\delta\boldsymbol{\xi}\times\mathbf{h}_1^T(t_m)\right] & \left[(t_m - t_{m-1})\boldsymbol{\xi}(t_{m-1})\times\mathbf{h}_1^T(t_m)\right] & \mathbf{0}_{3x3} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{3x3} & \left[\delta\boldsymbol{\xi}\times\mathbf{h}_N^T(t_m)\right] & \left[(t_m - t_{m-1})\boldsymbol{\xi}(t_{m-1})\times\mathbf{h}_N^T(t_m)\right] & \mathbf{0}_{3x3} \end{bmatrix} \qquad \textit{(Eq. 3.20)}$$

where $\xi(t_m) = \mathbf{C}_b^n(t_m)\mathbf{d}$, $\delta\boldsymbol{\xi} = \xi(t_m) - \xi(t_{m-1})$ and $\mathbf{\Phi}(t_2, t_1)$ is the discrete-time state transition matrix from time epoch t_{m-1} to time epoch t_{m-2} derived from **F** in (3.11).

The position part of the estimator is a has an extremely simple form that includes a forcing function from the dynamics part. A forcing function does not depend on the measurements, the dynamics obtained from the inertial are directly used to update the current position:

$$\mathbf{r}_{INS}^n(t_m) = \mathbf{r}_{INS}^n(t_{m-1}) + (vector\ forcing\ function) \qquad \textit{(Eq. 3.21)}$$

Using the inertial estimate of the position, pseudoranges from the inertial are predicted

(synthesized):

$$\rho_{INS,j} = \sqrt{\left(x_{INS} - x_j\right)^2 + \left(y_{INS} - y_j\right)^2 + \left(z_{INS} - z_j\right)^2} \qquad \textit{(Eq. 3.22)}$$

The measurements vector are the differences between the measured pseudoranges and the synthesized pseudoranges computed in equation 3.22:

$$z_{Pos,j} = \rho_{GPS,j} - \rho_{INS,j} \qquad \textit{(Eq. 3.23)}$$

And the linearized measurement equation is:

$$H_j = \left[\begin{bmatrix} \frac{x - x_j}{Rj} & \frac{y - y_j}{Rj} & \frac{z - z_j}{Rj} \end{bmatrix} I_{3x3} \quad 0_{3x3} \quad 0_{3x3} \quad 0_{3x3} \quad 0_{3x3}\right] \qquad \textit{(Eq. 3.24)}$$

3.7. Ladar or 3D imager as secondary sensor

In case a 3D imaging (Ladar) sensor is available, features, such as planar surfaces, can be extracted from the point cloud data directly using, for example, the ROS PCL. In that case a change in position and attitude can be directly observed from the change in closest distance, ρ_i, from the imager origin to the planar surface and the normal vector, **n**, of that surface expressed in the camera frame at time epochs t_m and t_{m-1} following:

$$\begin{bmatrix} \mathbf{n}_1^T \\ \vdots \\ \mathbf{n}_N^T \end{bmatrix} \Delta\mathbf{r} = \begin{bmatrix} \rho_1(t_m) - \rho_1(t_{m-1}) \\ \vdots \\ \rho_N(t_m) - \rho_N(t_{m-1}) \end{bmatrix} \Rightarrow \mathbf{H}\Delta\mathbf{r} = \Delta\boldsymbol{\rho}_L \qquad \textit{(Eq. 3.25)}$$

Using the information provided by the IMU (estimated translational and rotational motion), expressions of the change in the shortest distance to the planar surfaces can be drawn and used to estimate the velocity and position errors in the IMU. More details and a derivation of the attitude estimator can be found in [56].

The complementary filter is formed by differentiating the change in the shortest distance calculated using the Ladar and the IMU data and can be set up with the following measurement. This will yield the necessary information to correct position, but does not observe attitude errors. A second equation has been derived to measure the error in attitude

of the normal vector of each plane:

$$\begin{aligned} z_{j,Lp} &= \Delta\rho_{j,L} - \Delta\rho_{j,INS} \\ \mathbf{z}_{j,La} &= \mathbf{n}_{j,L} - \mathbf{n}_{j,INS} \end{aligned} \qquad \textit{(Eq. 3.26)}$$

where $\Delta\rho_{j,INS}$ is the change in closest distance derived from the inertial mechanization and $\mathbf{n}_{j,INS}$ is the normal computed from the previous Ladar output and the inertial position and attitude estimates:

$$\begin{aligned} \Delta\tilde{\rho}_{j,INS} &= \tilde{\mathbf{n}}_j^T \tilde{\mathbf{C}}_n^b \Delta\mathbf{r}^n \\ \tilde{\mathbf{n}}_{j,INS}(t_m) &= \mathbf{C}_n^b(t_m)\mathbf{C}_b^n(t_{m-1})\tilde{\mathbf{n}}_{j,L}(t_{m-1}) \end{aligned} \qquad \textit{(Eq. 3.27)}$$

The derivation of the measurement equation that relates equation (3.23) to the augmented state vector (3.18) given the erroneous parameters as defined in (3.24) can be found in [53]. One row of each of these **H** matrices is provided here:

$$\begin{aligned} \mathbf{h}_{j,Lp} &= \begin{bmatrix} \mathbf{n}_j^T(t_{m-1})\mathbf{C}_n^b(t_{m-1}) & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} \end{bmatrix}^T \\ \mathbf{h}_{j,La} &= \begin{bmatrix} \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \left(\mathbf{C}_n^b(t_m)\mathbf{C}_b^n(t_{m-1})\mathbf{n}_j(t_{m-1})\times\right)^T & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} \end{bmatrix}^T \end{aligned} \qquad \textit{(Eq. 3.28)}$$

Often only a 2D Ladar (laser scanner) is used since it is more accurate, lighter and cost-effective. In that case 2D features such as lines can be extracted and mapped on 3D space. However, estimation of position and attitude for more complex 3D motion will not be possible. An alternative has been proposed and evaluated successfully in [56]. The method in that paper exploits existing or intentional motion of the platform to increase the field-of-view of the Ladar resulting in point cloud measurements in 3D rather than 2D. Note that the introduced motion must be taken into account carefully.

Other features that have been considered are 3D points and 3D lines. A detailed discussion and derivation on the measurement equations for those features is in [53].

3.8. A stereo camera as secondary sensor

Calibrated stereo cameras can easily estimate the feature depth by comparing the pixel location of feature matches. The difference on the location of the feature seen in both camera is called disparity 'd'. In combination with the known baseline 'b' and the focal length 'f', the following equation provides and accurate estimation of the depth:

$$d = x_1 - x_2$$
$$\frac{d}{b} = \frac{f}{z} \qquad \textit{(Eq. 3.29)}$$

The 3D point can also be estimated based on the coordinates from the previous and the inertial estimation of translation and rotation. The detailed elaboration of these expressions is skipped here but can be found in [53]. It results on the following measurement equation for a single feature:

$$z_{i,p_{3d}} = C_n^b(t_k)\delta r^n - \left[\Delta t C_n^b(t_k) p_{INS,i}^n(t_{k-1}) \times C_b^n(t_k)\right]\delta\omega + \varepsilon_i(t_k) \qquad \textit{(Eq. 3.30)}$$

Therefore, through these features, position displacement error, δr, is directly observable. This yields an observation matrix for 3D corners and points where the contribution of one observed feature is shown in equation:

$$\mathbf{H}_{i,p_{3d}}(t_k) = \left[\mathbf{C}_{\boldsymbol{n}}^{\boldsymbol{b}}(t_k) \quad \mathbf{0}_{3x3} \quad \mathbf{0}_{3x3} \quad -\left[\Delta t \mathbf{C}_{\boldsymbol{n}}^{\boldsymbol{b}}(t_k)\mathbf{p}_{\boldsymbol{INS,i}}^{\boldsymbol{n}}(t_{k-1}) \times \mathbf{C}_{\boldsymbol{b}}^{\boldsymbol{n}}(t_k)\right] \quad \mathbf{0}_{3x3}\right] \qquad \textit{(Eq. 3.31)}$$

3.9. 2D imager as secondary sensor

Unlike the GNSS- and Ladar-based integration approaches, no direct measurement can be derived for the 2D image features. Instead, a constraint-based (or implicit) complementary Kalman filter will be used. In this filter the input to the filter is a constraint equation rather than the difference between a measurement and a quantity synthesized using outputs from the inertial mechanization. While various features can be considered for 2D imagery, only point features are shortly considered here. For a discussion on other 2D imagery features the reader is referred to [53].

An example of such a constraint (the so-called epipolar constraint) can be derived given the geometry shown in Figure 5. The 3D point **p** is observed at two time epochs in the respective body frames of the camera resulting in two unit-length pointing vector $\mathbf{e}_I^b(t_{m-1})$ and $\mathbf{e}_I^b(t_m)$. Note that the actual vectors ($\mathbf{p}_I^b(t_{m-1})$ and $\mathbf{p}_I^b(t_m)$) are unknown due to the unknown length (the "depth" problem).

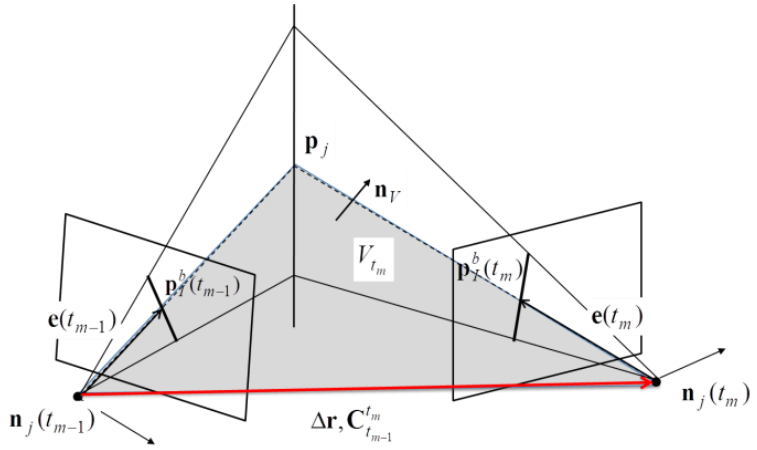


Fig. 17 Point feature constraints

Now the normal vector (expressed in the "navigation" frame) of the plane spanned by $\mathbf{e}_I^n(t_{m-1})$ and $\Delta\mathbf{r}^n$ is given by $\mathbf{n}_V = \mathbf{e}_I^n(t_{m-1}) \times \Delta\mathbf{r}^n$. As can be seen in Fig. 17, this vector is perpendicular to $\mathbf{e}_I^n(t_m)$, resulting in the following constraint equation:

$$\begin{aligned} &\mathbf{e}_I^n(t_m) \cdot \left(\mathbf{e}_I^n(t_{m-1}) \times \Delta\mathbf{r}^n\right) = 0 \Rightarrow \\ &\left(\tilde{\mathbf{C}}_b^n(t_m)\mathbf{e}_I^b(t_m)\right) \cdot \left(\tilde{\mathbf{C}}_b^n(t_{m-1})\mathbf{e}_I^b(t_{m-1}) \times \Delta\tilde{\mathbf{r}}^n\right) = 0 \end{aligned} \qquad \textit{(Eq. 3.32)}$$

Another constraint that can be observed from Fig. 17 is given by:

$$\begin{aligned} &\mathbf{p}_I^n(t_m) = \Delta\mathbf{r}^n - \mathbf{p}_I^n(t_{m-1}) \Rightarrow \\ &\tilde{\mathbf{C}}_b^n(t_m)\mathbf{p}_I^b(t_m) - \Delta\tilde{\mathbf{r}}^n + \tilde{\mathbf{C}}_b^n(t_{m-1})\mathbf{p}_I^b(t_{m-1}) = 0 \end{aligned} \qquad \textit{(Eq. 3.33)}$$

Off course, this constraint will not equal zero due to errors in all components of equation. Note that the tilde is used in both equation (3.29) and equation (3.30) to indicate the use of an erroneous estimate for the INS mechanization. The use of erroneous inertial values (in addition to the noise that may be present on the derived features themselves), allows for the derivation of an appropriate H matrix. Because of the complexity of this matrix it is not given here, but can be found in [53].

The "constraint measurement" results as:

$$\begin{aligned} z_{j,P_{2D}} = \; & -\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_{\boldsymbol{n}}^{\boldsymbol{b}}(t_k)\delta\mathbf{r}^{\boldsymbol{n}} - \left[\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_{\boldsymbol{n}}^{\boldsymbol{b}}(t_k)\Delta\mathbf{r}\times\right]\boldsymbol{\psi}^{\boldsymbol{n}} + \\ & \left[\rho_i(t_{k-1})\Delta t\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_{\boldsymbol{n}}^{\boldsymbol{b}}(t_k)\mathbf{e}_i^n(t_{k-1}) \times \mathbf{C}_{\boldsymbol{b}}^{\boldsymbol{n}}(t_k)\right]\delta\boldsymbol{\omega}^{\boldsymbol{b}} + \varepsilon(t_k) \end{aligned} \qquad \textit{(Eq. 3.34)}$$

And it is a scalar; hence, each additional two-dimensional point feature adds one more constraint. For the H-matrix this means that for each addition 2D point feature, one row is added:

$$\mathbf{H}_{P_{2D}}(t_k) = \left[-\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_{\boldsymbol{n}}^{\boldsymbol{b}}(t_k) \quad \mathbf{0}_{1\times 3} \quad -\left[\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_{\boldsymbol{n}}^{\boldsymbol{b}}(t_k)\Delta\mathbf{r}\times\right], \quad \left[\rho_i(t_{k-1})\Delta t\mathbf{e}_{i,\perp}^{b,T}(t_k)\mathbf{C}_{\boldsymbol{n}}^{\boldsymbol{b}}(t_k)\mathbf{e}_i^n(t_{k-1})\times\mathbf{C}_{\boldsymbol{b}}^{\boldsymbol{n}}(t_k)\right] \quad \mathbf{0}_{1\times 3}\right]$$

(Eq. 3.35)

3.10. Experimental results: GPS, inertial and laser slam fusion

In this experiment, the concepts described in previous sections will be tested. The proposed sensor setup consists of a GPS receiver, multiple 2D laser scanners, and an Inertial Measurement Unit (IMU) [51]. The proposed solution will combine three different estimators to demonstrate the feasibility of the indoor/outdoor seamless transition.

The left image in Fig. 18 shows the filtered GPS/INS trajectory superimposed on a photo of Stocker Center and surroundings generated using Google Earth™. The trajectory starts outside next to Stocker Center:

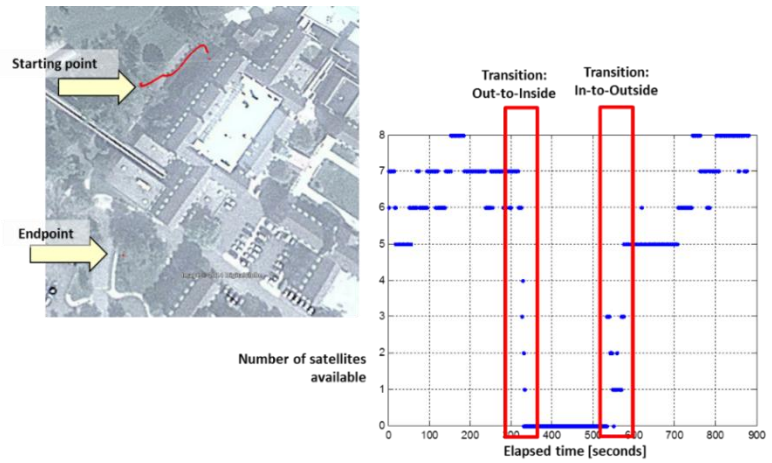


Fig. 18 Filtered GPS solution and visible satellites

The plot on the right side of Fig. 18 shows the visible satellites during the 882 seconds of this UAV flight trajectory. The transitions from the outside to/from the indoor environment can be easily observed. During these transitions the number of satellites rapidly drops when approaching the building, requiring integration with the INS to maintain a valid position as long as possible.

During the indoor operation of the UAV, no GPS satellites are available and the architecture

is solely based on the integration of the IMU with both 2D laser scanners and the baro-altimeter. The functional block diagram of this integration structure is shown in Fig. 19. Basically, the method consists of three main estimators: 3DOF SLAM, the altitude estimator and the 6DOF fusion algorithm [51].

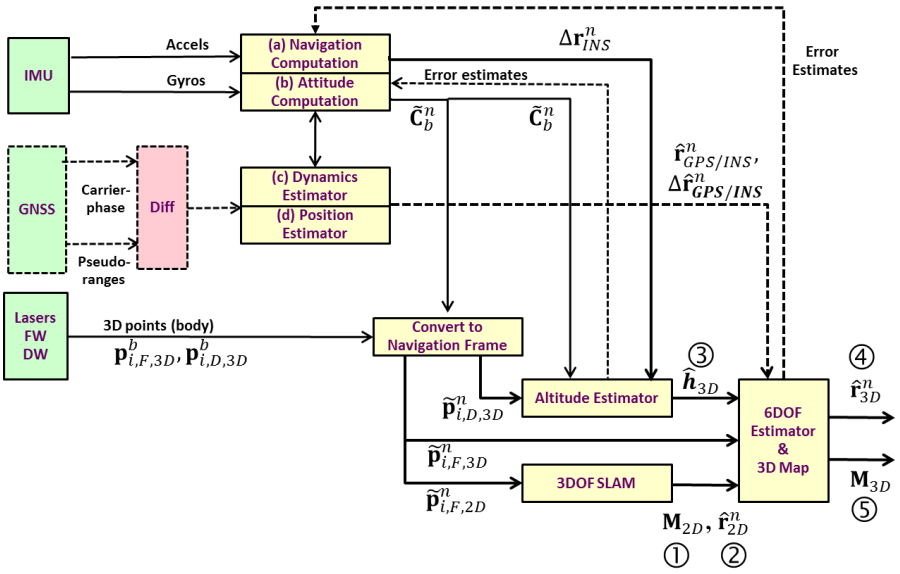


Fig. 19 Integration strategy during indoor (mode 3) and transition (mode 2).

In the method shown in Fig. 19, the complete 6DOF state vector $(x, y, z, \phi, \theta, \psi)$ is separated. The attitude (ϕ, θ) and inertial heading (ψ) of the platform are computed in the attitude computer and used to compute the body-to-navigation transformation matrix, $\tilde{\mathbf{C}}^n_b$. The tilde in $\tilde{\mathbf{C}}^n_b$ indicates that this matrix is effected by errors in the attitude and heading values, or $\tilde{\mathbf{C}}^n_b = (\mathbf{I} - \boldsymbol{\Psi})\mathbf{C}^n_b$. The 3DOF pose estimate $\hat{\mathbf{r}}^n_{2D} = (x, y, \psi)$ and the 2D map, $\mathbf{M}_{2D}$ (defined in only x and y directions) are estimated simultaneously as follows; transformation matrix $\tilde{\mathbf{C}}^n_b$ is used to convert the point cloud from the forward-pointing laser scanners to a frame that is aligned with the navigation frame but offset by the platforms translational motion. This point cloud (laser scan) is then matched with an occupancy grid resulting in a 2D pose estimate $\hat{\mathbf{r}}^n_{2D}$ and updated 2D map $\mathbf{M}_{2D}$. The method used here is similar to the one described in [22]. Fig. 21 shows the 2D occupancy grid after the UAV has traversed the Stocker basement next to an official map of the Stocker basement.

Fig. 20 2D indoor mapping results

Fig. 21 shows the trajectory consisting of all 2D pose estimates within this established 2D map. Note that the 3DOF SLAM was already activated outside since sparse features were available.

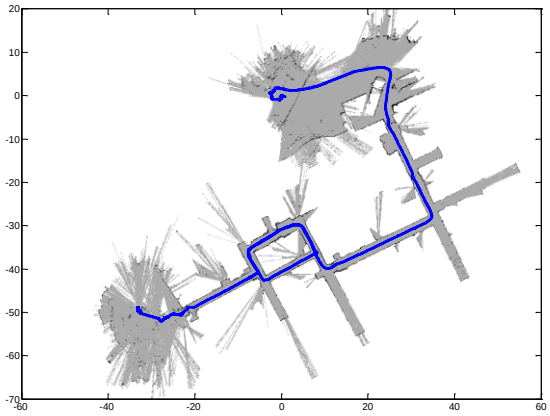


Fig. 21 UAV indoor trajectory

To obtain an estimate of the height, the downward-pointing laser scanner is utilized. Having a downward-pointing laser scanner rather than a mirror that deflects part of the forward-looking laser scanner, not only allows for a better estimate of the relative altitude, but also provides a cross-track vertical scan of the environment resulting in additional information for a 3D map. Furthermore, when assuming a structured environment with flat surfaces, the “slices” of the structured environment can be used to estimate the misorientation, Ψ, present in the attitude. The misorientation estimator is outside the scope of this thesis.

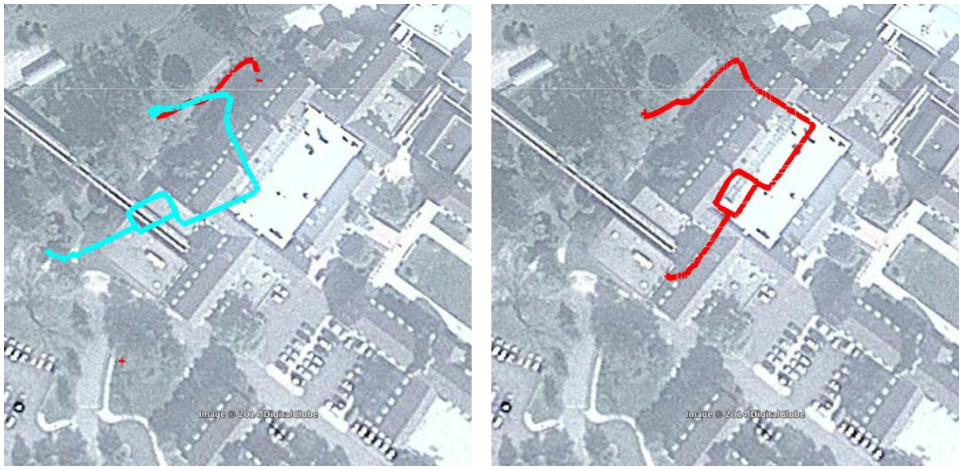


Fig. 22 Results (left) no integration of GPS with SLAM solution; (right) integration of GPS with SLAM solution [51]

The 2D trajectory established by the poses output by 3DOF SLAM method and visualized in Fig. 19, must be referenced to a true navigation fame (North-East-Down). This can be accomplished when the UAV is operating in the vicinity of buildings with sufficient GPS satellite visibility (during Mode 2). In that case the GPS/INS position and velocity estimator can be used and fused with the "3D position estimate" from the combined altitude estimator and 3DOF SLAM in a straight forward manner using a simple KF. Fig. 22 shows the results of this integration. On the left, the GPS/INS trajectory and the 3DOF SLAM trajectory are shown without integration. Clearly the reference frame is offset and rotated with respect to the NED frame. This can be explained by the limited availability of laser-features at the start point of the UAV trajectory and the unknown initial heading when using the 3DOF SLAM method.

3.11. Future works: MAV Copilot

The sensor setup on the MAV Copilot offers several sources for correction of the inertial drift and presents a great configuration to evaluate the algorithm described in this section.

The set of sensors available on the Copilot correspond to the architecture described in Fig. 14. As MAVs are limited in the amount of weight they can carry, the laser ranger has been replaced by a stereo camera that should be able to reliable estimate depth to features in the scenarios of our interest. The following figure summarizes the envisioned operational scenario:

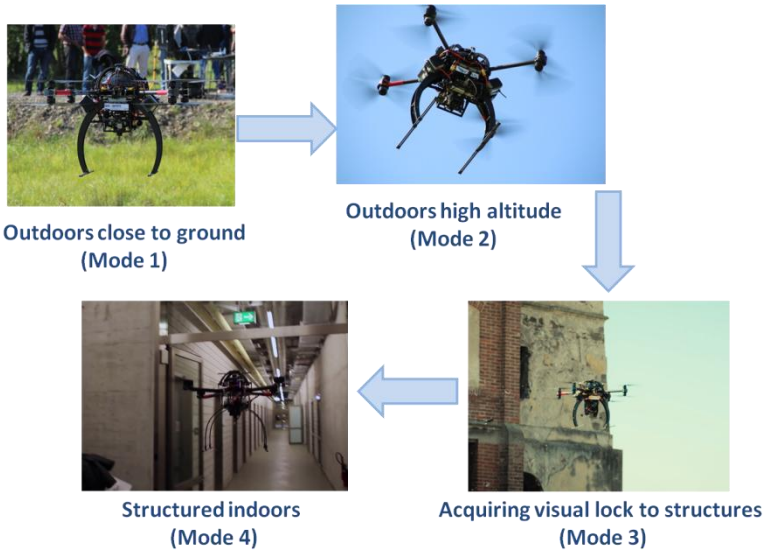


Fig. 23 MAV Copilot operational scenario

While operating at low altitude in the outdoor environment with enough GPS availability the algorithm relies mainly on the tight integration of GPS and IMU with the aid of an altimeter (barometric and/or laser); this operational phase is referred to as Mode 1.

Next, the UAV gain altitude to enter in Mode 2. At high altitude, the stereo pair behaves as a monocular camera since the baseline is very small compare to the current altitude.

Afterwards, the platform gets close to a structure. The stereo pair acquires the building and the system prepares for the transition to the indoor environment (Mode 3).

During this transition the number of available satellite significantly reduces, requiring augmentation of the filter with additional information from the range estimates from the stereo pair.

Finally, when inside the building GPS measurements are completely omitted from the solution, and the filters only rely on the vision, altimeter and IMU data (Mode 4).

4. Fast Aerial Mapping using Structure from Motion

The work described in this section has been published in the Journal of Field Robotics 2016 with the title "*Combined aerial and terrestrial 3D mapping for improving the situational awareness in search and rescue operations",* in collaboration with other partners of the project ICARUS.

The use of small UAVs may improve the response time and coverage for search and rescue allowing search and rescue teams to systematically survey and perform mapping of areas of importance in real time without any physical interaction within dangerous zones [1].

This section describes our own implementation of a fully automated fast mapping software. Our end-users clearly stated that the most important performance requirement for the aerial assessment is speed. The aerial mapping algorithm described here prioritizes speed over the precision or correctness of the resulting map. Our implementation performs the following steps:

I. The images captured, geotagged and stored on the onboard hard disk are later downloaded to the GCS and prepare for the processing.

II. The software searches for matching points by analyzing an optimized pair-selection of the images. We use SiftGPU [58], a GPU implementation of the well known SIFT descriptor and matcher [59].

III. Those matching points are used in a bundle adjustment [30] to reconstruct the exact position and orientation of the camera for every acquired image as well as the 3D coordinates of each matched point. We use the open-source Bundler software [60] [61]. Based on this reconstruction the matching points are verified and their 3D coordinates calculated. The geo-reference system is WGS84, based on GPS measurements from the UAV autopilot during the flight.

These steps are common to most aerial mapping software as for instance Pix4D [29]. The result from these steps is a sparse 3D point cloud. Afterwards, most software packages generate a mesh by triangulation, construct a dense 3D point clouds and project the texture of the original pixels/images onto them. This provides a Digital Elevation Model that can be later computed to produce an ortho-photo. Instead, our pipeline does the following:

IV. Generates an imaginary flat ground and computes the projection onto that ground of the corrected 3D points. Projects each of images to the ground, blending the overlaps

V. And geo-referencing the final map.

The software has been encapsulated in separate modules described in the following sections: Matcher, Bundle adjustment, Mosaicing and Georeferencing. The Fig. 24 illustrate the input and output of each step.

4.1. Data capture, geotagging and automatic download

The drawback of autonomous platforms such as MAVs certainly lies in the relatively low accuracy of their orientation estimates. Since their miniature on-board autopilots cannot deliver extremely precise positioning and orientation of the recorded images, post-processing is key in the generation of georeferenced ortho-images and digital elevation models (DEMs) [29].

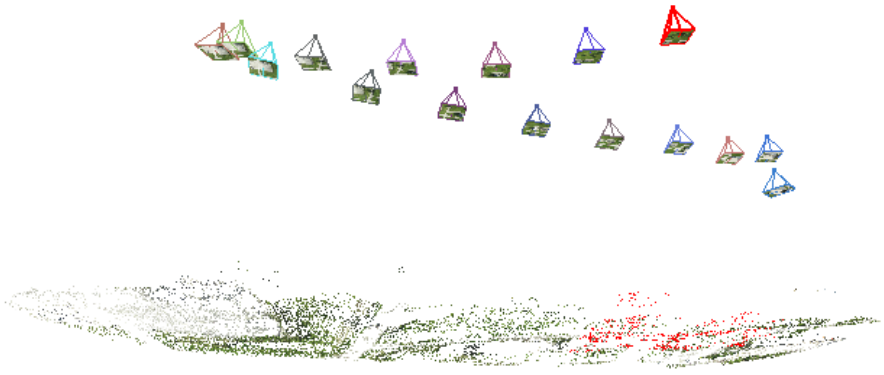
We have used two different payloads:

- A high-resolution camera mounted on a gimbal in NADIR configuration. Two cameras has been used, a Canon IXUS 110 with 12 megapixels, and a Sony α4000 with 24 megapixels. Photo cameras store images on a storage card which has to be manually removed to access them. These images are later geotagged using the flight log via the utilities built in in the MissionPlanner software [62].

- Integrated cameras on the ICARUS payload described in □. These images are accessible on the onboard computer and automatically geotagged using the position estimate of the autopilot and stored on the onboard PC. When the last waypoint of the mission is achieved, the images are downloaded to the ground station (during return-home) and post-processing is started.

(a)

(b) (c)



d)

(f) (g)

Fig. 24 Aerial mapping pipeline
a) raw images,
b)SIFT descriptors,
c) SIFT matches,
d) camera P matrices and sparse point cloud from Bundler,
f) map
g) georeferenced map

4.2. Matcher

The first module called *Matcher* is responsible for pre-pairing and matching the images.

The first step is to convert the images to grayscale, which is required for the later processing, to scale them down if necessary and to extract the *SIFT* feature descriptors. All this information is stored to disk.

A significant part of the time spent in mapping is typically spent on matching features across the images. Matching datasets corresponding to long flights can take a long time, even hours (exponential with the number of images). Images pre-selection speeds up this process by matching only those pairs of images with significant overlap.

Our algorithm selects the images by distance between the viewpoint locations. Only those images closer than a configurable threshold are matched. This is done incrementally; every new image is compared only against the previous ones. The distance is obtained using the *haversine* formula [63] following the equations below:

$$a = \sin^2(\Delta lat/2) + \cos lat_1 \cdot \cos lat_2 \cdot \sin^2(\Delta lon/2) \qquad \textit{(Eq. 4.1)}$$

$$c = 2 \cdot \arctan^2\left(\sqrt{a}, \sqrt{(1-a)}\right) \qquad \textit{(Eq. 4.2)}$$

$$d = R \cdot c \qquad \textit{(Eq. 4.3)}$$

where R is earth's radius (mean radius = 6,371km). This is used to calculate the great-circle distance (d) between two points, that is, the shortest distance over the earth's surface.

To speed up the extraction of the descriptor and the matching, we use a GPU implementation of SIFT [58]. *SiftGPU* processes pixels in parallel to build Gaussian pyramids and detect DoG Keypoints. *SiftGPU* then uses a GPU/CPU mixed method to efficiently build compact keypoint lists. Finally keypoints are processed in parallel to get their orientations and descriptors.

We also use the GPU exhaustive/guided sift matcher *SiftMatchGPU*. It basically multiplies the descriptor matrix on GPU and finds the closest feature matches on GPU. Both GLSL and CUDA implementations are provided.

The output of this step is a set of features matched across multiple images (photos) for which we know the viewpoint coordinates.

4.3. Bundle adjustment

The situation described above is referred as the *projective reconstruction problem* [60]. A set of 3D points X_j (the matching features) is viewed by a set of cameras with matrices P^i (the photos). Denote x_{ij} the coordinates of the *j*-th point as seen by the camera *i*-th, we want to find the camera matrices P^i (describing the position and rotation of the cameras).

If the image measurements are noisy, then the following equation will not be satisfied exactly:

$$x_{ij} = P^i x_j \qquad \textit{(Eq. 4.4)}$$

The estimation involving minimizing the reprojection error is known as *bundle adjustment*. It involves adjusting the bundle of rays between each camera center and the set of 3D points.

Bundler [60] is a structure-from-motion system for unordered image collections (for instance, images from the Internet). It reconstructs the scene incrementally, a few images at a time, using a modified version of the *Sparse Bundle Adjustment* package in [64].

The *Matcher* module generates the input files required by *Bundler*, including a list of the images, *SIFT* descriptor and matches and camera focal length. *Bundler* produces a 3D reconstruction of the camera and (sparse) scene geometry as output described as:

- For each camera (photo):
  - estimated focal length and radial distortion coefficients
  - *R* 3x3 matrix representing the camera rotation
  - *t* 3-vector representing the camera translation

- For each point:
  - *position* 3-vector representing the 3D position of the feature
  - *color*
  - list of views describing the views of this feature on the cameras, described as:
    - camera ID
    - index of the SIFT descriptor
    - *x* and *y* image pixel coordinates

The 3D coordinate system used by Bundler is completely arbitrary. Therefore, any correspondence with the global coordinates needs to be reconstructed. These results are used by the *Mosaicing* module to produce a fast map.

4.4. Mosaicing

The homography transformation is a popular geo-referencing technique used worldwide. It relates any two images of the same planar surface. In this implementation, we will make the assumption that the ground is flat and the map is a perspective view of the ground.
In this implementation, for each of the photos, we will first compute the homography that relates the features on the image with their correspondence on the arbitrary space produced by Bundler.

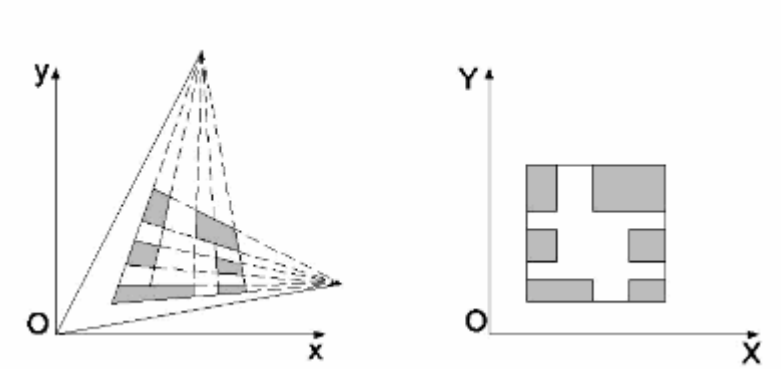


Fig. 25 Homography between the 3D Bundler and the image (Credits: Coormap)

Afterwards, using this homography, we will project the four corners of the image onto the arbitrary coordinate system. Using both the original corners on the image and their correspondence projected on the arbitrary coordinate system, we will compute a perspective transform from the image to the map, and warp the original image onto the map.

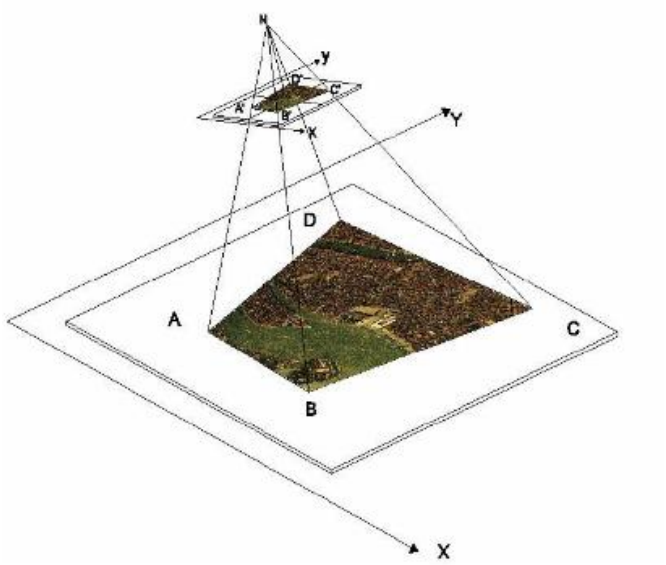


Fig. 26 Perspective transform to project the image into the map (Credits: Coormap)

The warped image may be shifted and rotated on the map. Proper handling of offsets and

increase of size of the temporary map is needed.

The following sequence of images illustrates the process:

Fig. 27 Stitching from Bundler

4.5. Georeferencing

The use of Bundler software shows a single drawback: the arbitrary coordinate system used for the 3D reconstruction is not geo-referenced. If the coordinate system of the input data is not known it may be possible to use a 2D Cartesian transformation. 2D ground control points (GCPs) or common points are then required to determine the relationship between the unknown and the known coordinate system. The transformation may be conformal, affine, polynomial, or of another type, depending on the geometric differences between the two map projections.

Therefore, to georeference the map, we need to estimate the transformation between the world coordinates and the 3D Bundler coordinate systems. This will be done in several steps:

- compute the transformation between world coordinates and Bundler 3D world
- compute the transformation between world coordinates and the pixels on the map
- compute orientation of the map and rotate it to be north-aligned
- compute the world coordinates of the corners of the map

In geodesy, the science of the measurement and mapping of the earth's surface, this problem is usually solved using the *Helmert transformation* [65]. It refers to the transformation involved in changing the coordinates of a point set with respect to one reference surface to make them refer to another reference surface, and involves rotation, scaling and translation. When both sets of points are given, then least squares can be used to solve the inverse problem of determining the parameters. In particular, the parameters of the so-called 7 parameter transformation can be obtained by standard methods.

As a simplification, we assume that the transformation between our two spaces preserves points, straight lines and planes [60]. The algorithm uses an Affine transformation: a linear (or first-order) transformation that relates two 2D Cartesian coordinate systems through a rotation, a scale change in x- and y-direction, followed by a translation.

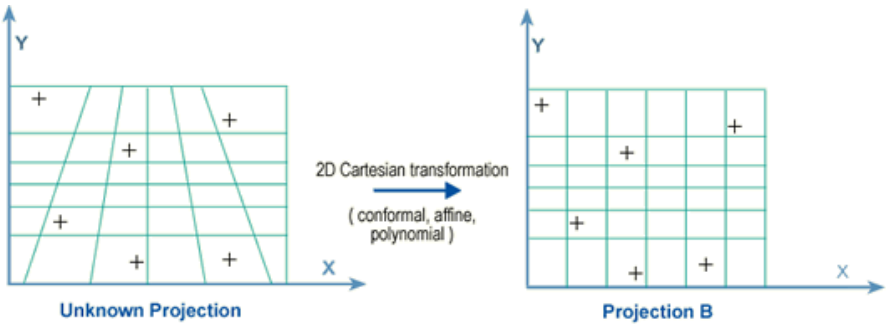


Fig. 28 Coordinate conversions

The transformation function is expressed with 6 parameters: one rotation angle (α), two scale factors, a scale factor in the x-direction (s_x) and a scale factor in the y-direction (s_y), and two origin shifts (x_o, y_o). The equation is:

$$X' = s_x X \cos(\alpha) - s_y Y \sin(\alpha) + x_0$$
$$Y' = s_x X \sin(\alpha) - s_y Y \cos(\alpha) + y_0 \qquad \textit{(Eq. 4.5)}$$

World coordinates to Bundler transformation

To compute the affine transformation that relates the world and 3D Bundler coordinates, we will use the estimated absolute location of the viewpoint for each of the images (measured from GPS) and the reconstructed coordinates of the cameras in 3D produced by Bundler, therefore already adjusted or corrected. An affine transform has 6 degrees of freedom and can be computed from 3 points. We have implemented a highly simplified version of RANSAC [60]:

- iterative sample 3 points that will serve as our *hypothetical inliers,*
- compute the affine transform from them
- test this transform against all the other points on the set
- if sufficient points are within a threshold, the transform is accepted

World coordinates to map transformation

At this point, we can estimate the latitude, longitude and altitude coordinates of any point in the 3D Bundler coordinates. However, our goal is to georeference the pixels on the map.

We will use control points from the images for it. A control point is a one of the points in the Bundler sparse cloud. For each of these points, we know the 3D coordinates in Bundler and the pixel coordinate in those images where the point was viewed. Furthermore, after the previous step we can also estimate its world coordinates.

We will use a configurable number of control points per image. During the mosaicing step, the four corners of each image are projected onto the map. We now project also these control points. Therefore, we also know the pixel coordinate of our control points on the map.

Similarly to the case above, following the same iterative approach, we will compute the affine transformation that relates the world coordinates and the pixel on the map. This is effectively georeferencing the map.

Map export

To facilitate the visualization map some more handling is needed. Most GIS systems and for instance the KML overlay data type, require the map to be north aligned. The orientation of the current map is computed from the most extreme pixels and used for rotating it.

The resulting map is exported as a *png* file and a raster file in *tiff* format. KML files are also generated for easy visualization through Google Earth.

4.6. Collaborative mapping

In the context of the ICARUS project integration trials, ground and air robots carry out mapping tasks over the same scenario [66]. Collaborative mapping consists on the fusion of data coming from the different platforms. It can be done at several levels. A tightly-integrated fusion strategy implies that raw data is fused, for instance, images from the air and ground robots are matched to reconstruct the scenario. A loosely- integrated fusion strategy assumes that the results of the independent mapping tasks is fused.

This work was done in collaboration with the Institute of Mathematic Machines in Warsaw (Poland), the Royal Military Academy of Belgium (RMA) and the Autonomous Systems Lab at ETH Zurich. A loosely-integrated fusion has been done for a UAV and a UGV. Both platforms generate a dense 3D reconstruction of the area and share it as point clouds in local coordinates, for which we keep the corresponding world coordinate. The point clouds are then merged, initially by hand, but later by means of an semantic Iterative Closest Point (ICP) where points are only matched with points on the same class (essentially surfaces) .

The mapping results of this project were used in these experiments and has been published on Journal of Field Robotics2016 [66]. Some of these results are shown in section 6.

4.7. Maps classification

Remote sensing has been a very popular and well-established research topic for several decades. Images captured by satellite and manned aircraft were traditionally used but the recent progress in UAV-based aerial mapping (see section 4) has introduced a new paradigm. Remote sensing focus was traditionally strategic: it was used to provide high level periodic information to support coordination, sectorization and prioritization. The fast availability of UAV-based maps allows now a tactical approach: the information can be used immediately by the actors on the field to plan how the next approach the next step and who is to do it. This poses new challenges: remote sensing techniques needs to aim at immediate results.

The classification of aerial images into several regions (classes) enhances the overall situational awareness at the On-Site Operations Coordination Center (OSOCC). This information is very relevant and will help the coordinator to make better decisions. For instance, these are some examples of relevant information:

- Monitoring the evolution of the flooding in a tsunami,
- Assess the traversability of a route for the ground troops; sometimes they travel several hours at an average speed of 5Km/h through to end up on a blocked bridge,

or collapsed roads.

- Estimate size of urban areas and probability density of survivors.

There is a vast literature on research efforts, but due to high variability in aerial imagery, automatic classification and semantic description still pose an open research problem [66]. Most people has developed sophisticated methods that fuse several sources of information such as edge responses, height information, normalized ratio between image planes, etc. Among the typical classifiers, the following can be found maximum likelihood, neural network, decision tree and support vector machine (SVM) [67] [68].

However this research effort stands in contrast to the practical reality in the field, where these tools have great difficulty finding their way to the end-users [4]. An attempt to provide an easy-to-configure tool is explored here. The objective is the fast classification of aerial images in Search and rescue operations. Similarly to the approach to the victim detection algorithm described in section 5, the aim is to provide the rescue rs with some tools to quickly and easily adapt the parameters to the particular characteristics of the scenario.

The map segmentation and classification algorithm described here was initially developed in the Computer Vision subject in the first semester of the Master. This section briefly summarizes the key concepts. The algorithm has been tested during the large scale demonstration of the project and results are described in section 6.

The implemented method consists of a simple pixel-level classification based on a minimum Mahalanobis distance [69] classifier in the HSV (Hue, Saturation and Value) space, using only HSV values, followed by some morphological operations to enhance the segmentation. The following regions have been defined:

1. Vegetation, wood, grassland, fields
2. Concrete: manmade structures like streets, gray-roof buildings, etc
3. Houses, colored roofs, etc
4. Land, soil, bare earth.
5. Water

4.7.1. HSV and RGB spaces

RGB is based on Cartesian coordinates and there is a high correlation between the three components which makes it very inconvenient for image segmentation. HSV uses a cylindrical coordinate. Hue is measured as an angle and value and saturation could be understood as percentage. In HSV, the three components are relatively independent and it is a continuous space. This leads to a natural, gradual and continuous distribution of tonalities and intensities which is very powerful for color segmentation and classification.

The following image illustrates the differences between both representations:

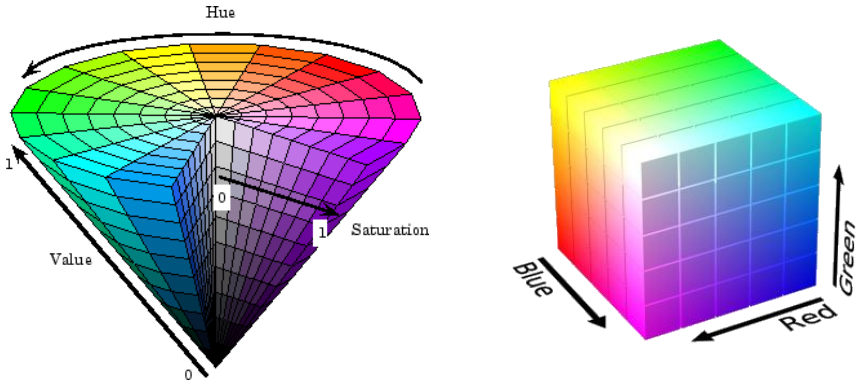


Fig. 29 HSV versus RGB color representation

4.7.2. HSV-based image classification

The algorithm implements a minimum distance classifier. The different classes are characterized according to the HSV values of their pixels. Each pixel in the sample map is assigned to the class at the minimum distance. A sequence of morphological operators further smoothens the classification.

The distance is computed as the Mahalanobis distance. Given a vector X=(x1, x2,..., xn), and a class with mean values M=(m1, m2,..., mn) with the covariance matrix C, the Mahalanobis distance is defined as:

$$d(X,M)=\sqrt{(X-M)^T C^{-1}(X-M)} \qquad \textit{(Eq. 4.6)}$$

If the covariance matrix is the identity, this is nothing but the Euclidean distance. If the covariance matrix is diagonal then the distance becomes the normalized Euclidean and can also be expressed as follows. Given two vectors X and the class described by the mean values M and standard deviations σi:

$$d(X,M)=\sqrt{\sum_{i=1}^{n}\frac{(x_i-y_i)^2}{\sigma_i^2}} \qquad \textit{(Eq. 4.7)}$$

The benefits of using the normalized Euclidean distance is to include the standard deviation in the computation, which in this classifier represents the variability of a given region and provides more flexibility to the algorithm.

To apply the Mahalanobis distance classifier, a method to calculate the HSV means and standard deviations for each class (region) was needed. Given a sample map, the user can define a region and assigned it to the desired class. After processing the pixels of the region, and estimating the means and standard deviation of the three components, this type of region distribution in the three-dimensional HSV space results, where the ellipsoids are center on the mean value and their axes are proportional to their standard deviations :

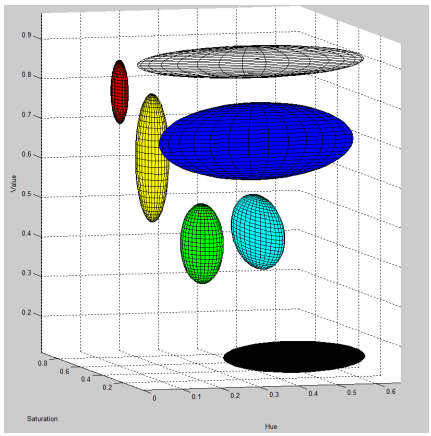


Fig. 30 Training classes in the HSV space. Red: roofs and bricks, yellow: land and paths, blue: water, green: vegetation, cyan: roads and paths

Two artificial classes has been added to handle the singularities of the HSV space (white and black areas). This pixels are assigned to the most popular surrounding region.

4.7.3. Morphological operations

A sequence of morphological operators were applied to each of the regions to improve the processing. The processing for each region is different, taking into account the different spatial characteristics of each regions. For instance, we know that vegetation and houses are spatially different in an image and we try to benefit from this knowledge. For example:

- in texture-rich regions such as vegetation, land and concrete a close operation is performed to remove small artifacts in the region.
- in more uniform regions such as water, a small close operation helps to refine the edges and a fill operator removes bigger miss-classification. These are usually due to high-reflection in water. The sea acts as a mirror as it waves, providing a wide range of saturation and values around the neutral hue and this processing showed decent results.
- Houses are difficult to pick and a more aggressive closes bring them up in the image making them more visible.
- In general in most of the regions we remove small areas as they are considered not relevant for the project.

4.7.4. Class stitching

Each of the classes forms a binary image. For visualization purposes, a color is applied to each of them. One of the advantages of this method is that each pixel is assigned to one and only one class and therefore there is no conflict to address here. The following diagram visualizes this step:

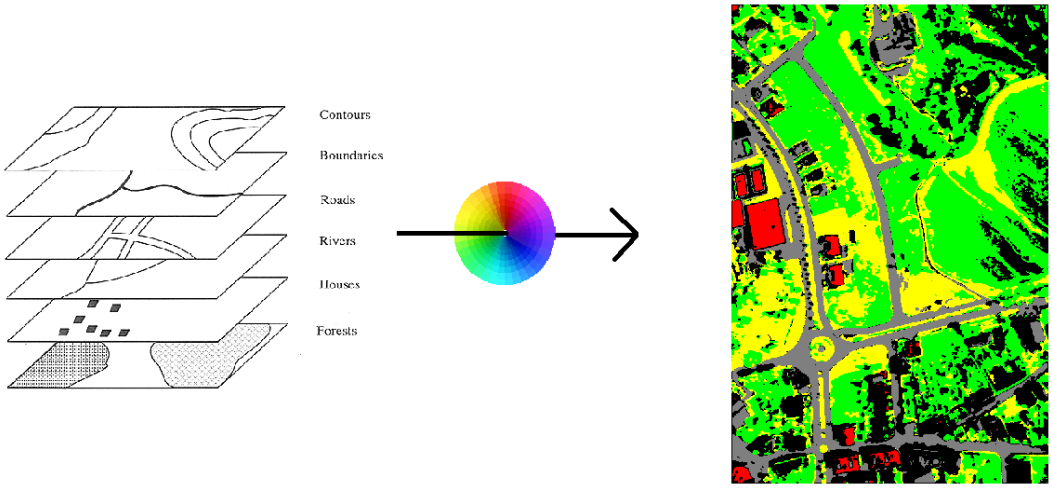


Fig. 31 Colouring the layers to form a unified classified map

This simple algorithm has been successfully tested during large-scale realistic scenarios.

4.7.5. Conclusions

The accuracy of the map highly depends on the ground resolution of the images and therefore the flight altitude, the sensor, etc. Let's review some parameters:

- Given the sensor size (mm) and the camera format (resolution in pixels), the pixel size (mm) can be computed as *sensor size/number of pixels*.
- The resolution of the images and maps on the ground or ground sample distance (GSD) is defined as *pixel size x flight height (AGL) / focal length*.
- An important parameter is the image footprint defined as the area on the ground captured within the image and can be computed by *camera resolution x GSD*.
- MAV are very agile platforms. The camera shutter speed must be set to 1/1000 at least to avoid blurred images.

For each of the images, enough matches with the neighbor images need to be found. This is ensure by forcing an overlap on the image captured of around 70% longitudinal and 50% lateral in the flight plan. Most of the MAV Ground Control Systems (GCS) provide support to generate an optimal flight plan for aerial mapping using the characteristics of your sensor.

The algorithm has been shown to work nicely as long as the quality of the dataset (illumination, overlap, precision, etc) meets the expected standards and so is the case for commercial software. An optimize flight planning, mapping oriented is critical to obtain good results.

The comparison in terms of performance of this algorithm against commercial software is probably not meaningful. Aerial mapping software such as Pix4D, Agisoft Photoscan and Microsoft ICE are really powerful tools with strong development and support teams progressing recurrently beyond state of the art really fast. The outcome of the algorithm presented here will hardly be to overcome their performance. The advantage of the developed pipeline is to be based on open-source components. The inputs and outputs of each step are generated and available to exploit with other existing tools (i.e.PVMS, etc).

Further developments of this algorithm should aim at real-time mapping. Bundler supports incremental bundler adjustment. Small modifications on the pipeline will allow to periodically invoke the bundle adjustment and mosaicing methods, for instance every time a flight leg is finished.

The system is ready and thought to be integrated onboard a MAV if required. The most important hardware requirement was the need for a GPU. This algorithm has been developed to run on CPU only and to exploit multiple cores if available. Therefore, any MAV embedded computer should be capable to run it. However, as long as there is reliable download link and not need to exploit the map for onboard planning, this is not a strong requirement. It still seems more efficient to provide a live digital stream of images to the ground control station and produce the map on the ground, with the support of more computational resources.

This implementation shows however a limitation, common to most of open-source mapping pipelines. It does not support off-nadir configurations. Images tilted more than 5 to 10 degrees will not be correctly stitched. Most commercial software already support this configuration.

The experiments on collaborative mapping have been published on Journal of Field Robotics2016 [66].

5. Victim Search

Aerial people detection and tracking using visible-light images have been subject to extensive research for many years already. The recent availability of affordable thermal cameras in civil applications has enabled a transition to the thermal detection or the combination of thermal and visible [28].

This section describes a new algorithm to detect victims on thermal images. The victim search described here is conceived to involve the operator as a supervisor. Simplicity in the controls and configurations and a high intuitive visualization are therefore a priority. It implements a rather common blob detector over a thermal image that has been pre-process to force human bodies (and other hot surfaces) to stand out. Similarly to the background subtraction methods, it assumes that humans have a different temperature than the surroundings (rubble, water, grass, etc). The algorithm performs a histogram normalization to improve object visualization, and a contrast enhancement, to increase the differences of temperature. The image is then binarized (thresholding) to extract the hot-spots. A sequence of morphological operators then enhances the performance of the detector. Since we have a stream of images at high update rate, we do tracking on top of pure detection to provide better performance [51][36]. All these steps are easily configured in real time by a reduced set of parameters highly intuitive for the operator.

This approach was inspired by the end-users feedback to provide an easy-to-configure algorithm as a practical alternative to more sophisticated detection and classification methods that usually require similar training data from previous operation, which is not always the case.

This implementation performs the following steps:

I. Thermal image normalization.

II. Victim detection

III. Victim tracking

5.1.1. Thermal imager

The victim search algorithm runs over the images captured from the FLIR Tau2 Camera. They are 14 bits, 640x512 intensity images. They are published as *sensor_msgs::Image* messages in ROS with *mono16* encoding. See [46] for details about the sensor.

The FLIR Tau2 implements several built-in image processing features to improve the quality

of the image. In particular, it provides a flat field correction (FFC). FFC is a process whereby offset terms are updated to improve image quality. During FFC, the camera shutter presents a uniform temperature source to each detector element in the array. While imaging the flat-field source, the camera updates the offset correction coefficients, resulting in a more uniform image after the process is complete. A faint 'click' may be heard when the shutter moves in front of the sensor. The imager provides three triggering modes: automatic, manual or external. The result of the FFC is called a correction map and can be stored in the non-volatile memory of the camera. [46]

For mobile robot applications where the searching area is large and can include different type of scenarios, a periodic FFC is required. For simplicity the automatic mode has been used.

5.1.2. Image normalization

The raw image provided by the camera driver represents each pixel value as 14 bits only. This is not supported by ROS. The first step in the algorithm is to scale the pixels values to use the full gray level scale:

$$dst(x,y) = \alpha * src(x,y) + \beta \qquad \textit{(Eq. 5.1)}$$

This operation is said histogram normalization, and it is typically used to improve the visualization of the objects of an image. *src* and *dst* represents the input and output image matrix respectively, α is the scale factor and β is the delta added to the scaled values.

Fixed Normalization

In this mode, the user selects the thresholds for maximum and minimum thermal intensities and α and β are obtained as follows:

$$\alpha = \frac{255}{highest - lowest} \qquad \textit{(Eq. 5.2)}$$

$$\beta = -lowest * \alpha \qquad \textit{(Eq. 5.3)}$$

Pixels with values higher or lower than the limits are saturated. This allows the user to quickly discard those regions that are either too cold or hot to be human bodies. There are certain scenarios with multi-modal images, for instance containing both land and sea, where the user may want to use this fixed normalization to focus on a given intensity range. The following image shows an example.

Fig. 32 Example of the fixed normalization method

Even though the roofs are burnt (saturated), the rest of the image shows a nice and clear level of detail. This gives the operator the flexibility to process the image for his own purposes. However, the drawback is also shown in this image.

The following figure shows how the corresponding histogram looks like. Since this is a fixed approach, the histogram will change with the changes of illumination and the operator will have to be in constant tuning to keep the interesting regions in range. For instance, the red line shows the average intensity of the pixels belonging to the human bodies. It is obvious that this normalization was too extreme and the human bodies have also been saturated.

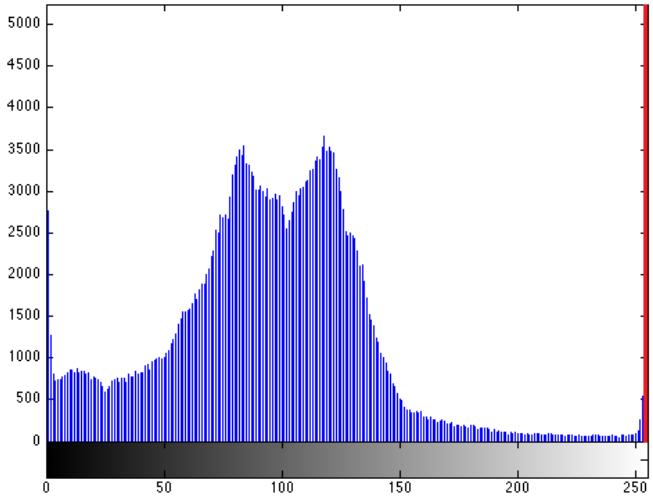


Fig. 33 Histogram of the fixed normalization method

Adaptive Normalization

In this mode, the image is analyzed to obtain the maximum, minimum and mean values. Assuming that the mean value corresponds to the average intensity of the background, we first compute the shortest distance from the mean to the maximum and minimum intensities values and then compute α and β values will be computed as follows:

$$\Delta = \min\left(\left|highest - mean\right|, \left|lowest - mean\right|\right) * \text{contrast\_booster} \qquad \textit{(Eq. 5.4)}$$

$$\alpha = \frac{255}{2 \times \Delta} \quad (5.1)$$

$$\beta = -\left(mean - range\right) * \alpha \qquad \textit{(Eq. 5.5)}$$

This adaptive normalization will focus on a bandwidth between the background intensity and the nearest value between the maximum and minimum intensities. This method centres the background intensity on the gray scale, and increase the contrast of the highlights and shadows of the image which will help the detector.

Fig. 34 Example of the adaptive normalization

The details on the roofs are now visible. Even though, in general, the image does not look as clear as before, this result is usually better for detection. The following figure shows the histogram. Its shape will remain pretty much constant through a entire flight, since this method is adaptive and will tend to concentrate on the center of the histogram the background values. This result will be used to detect the victim with the help of some morphological characteristics on the image.

Likewise, the red line shows the intensities of the human bodies:

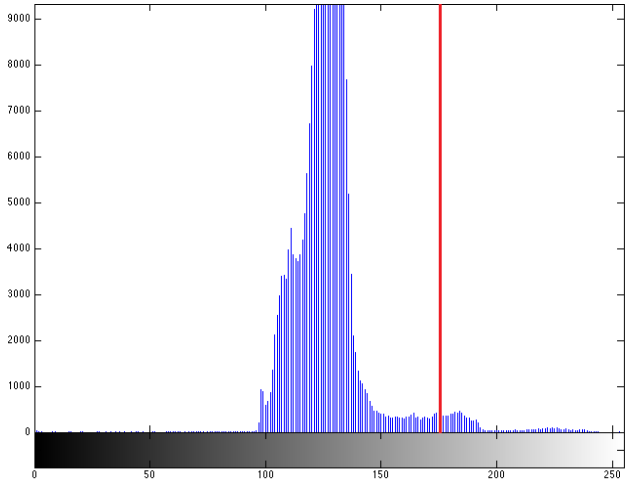


Fig. 35 Histogram of the adaptive normalization

In the adaptive mode, there is a contrast booster that allows to exaggerate the normalization around the mean to improve visualization. This approach will provide a more clear and detailed image, but, in some cases it might saturate most of the regions. It is up to the operator to use it with care. The following two figures show the image with contrast booster. The roofs are again visible. Even though the victims remain in the detectable zone, there are now more areas towards their right (in the histogram). The method improves visualization but does perhaps difficult the detection.

Fig. 36 Example of the adaptive normalization to improve visualization

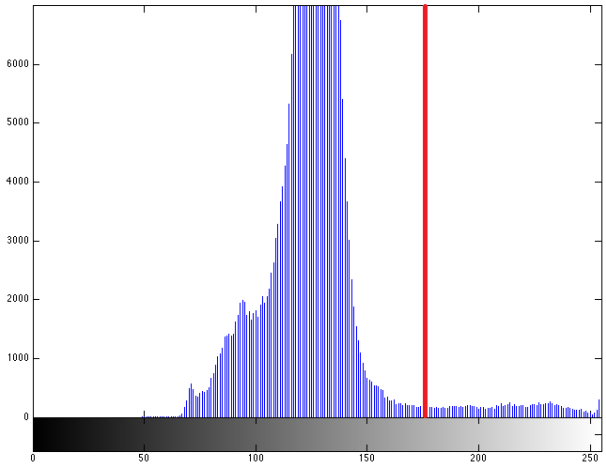


Fig. 37 Histogram corresponding to the previous figure

False Colouring

False colouring, as opposed to true-color images, is a method to render in colors data captured in the non-visible parts of the spectrum.

The thermal images are stored as monochrome and therefore visualized as grayscale images. Human perception theory states that our vision system isn't built for observing fine changes in grayscale images. Human eyes are more sensitive to observing changes between colors, so often colour maps are applied to grayscale images to improve the visualization.

This has been experienced in the project, but not always. The FFC mechanism described estimates correction offsets in flight that sometimes lead to sudden changes in the colour distribution of the image. This can be confusing to the operator and sometimes working on grayscale is more convenient. Both images are made available to the user and it is up to his/her preference to chose the mode. The following example illustrates the process:

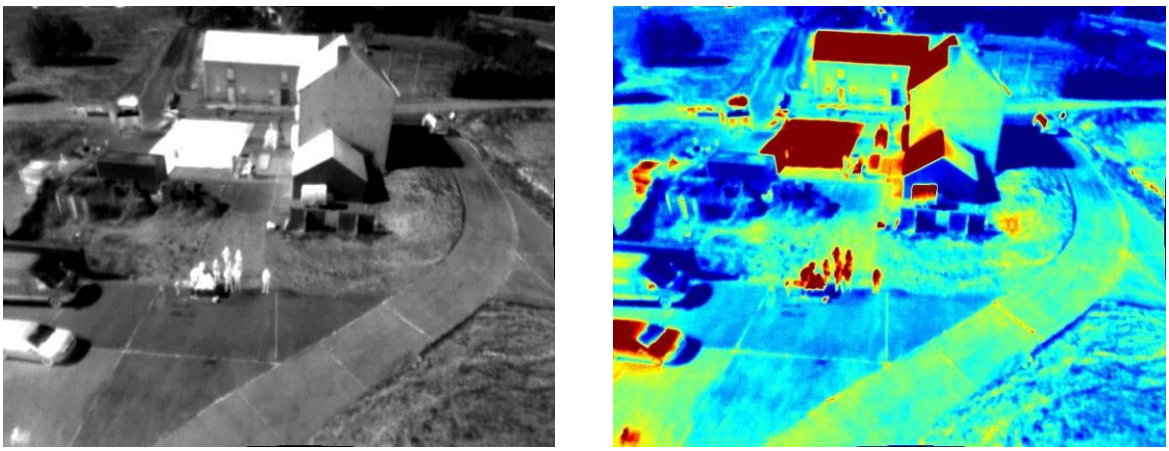


Fig. 38 False colouring of thermal images

Thermal_proc node

This functionality is deployed as a ROS node called *thermal_proc*. It subscribes to the raw thermal images and publishes the normalized image, both as grayscale and false coloured. The parameters are exposed to the user through the *dynamic_reconfigure* server, easily accessible with ROS *rqt*.

5.1.3. Victim detector

The victim detector receives as input the normalized images. After a binarization and some enhancements with morphological operators, it applies the well known blob detector to extract the labelled regions on the binary image. The detector also includes a minimum altitude threshold to disable it when the MAV is close to the ground.

Smoothing: Thermal images are corrupted by noise. A simple method to remove noise is a linear low-pass filter. In this case, we have used a linear convolution. The structuring element is a square box kernel. The aperture or size of the kernel defines the aggressiveness of the filtering and can be configured through the parameter *SmoothingFactor*.

Binarization by thresholding: This is the most critical step in this pipeline. Warm bodies come brighter in the image than the surroundings. The binarization is required to find contours and regions on the image. This operation will transform each pixel to black (zero intensity) or white (full intensity). There are several threshold methods. The easiest and fastest method is absolute threshold. In this method the resulting value depends on current pixel intensity and some threshold value. If pixel intensity is greater than the threshold value, the result will be white (255); otherwise it will be black (0).

The most critical parameter in this pipeline is this intensity threshold. This value is configured through the parameter *IntensityThreshold*. The following image illustrate the binarization transformation, where m indicates our threshold:

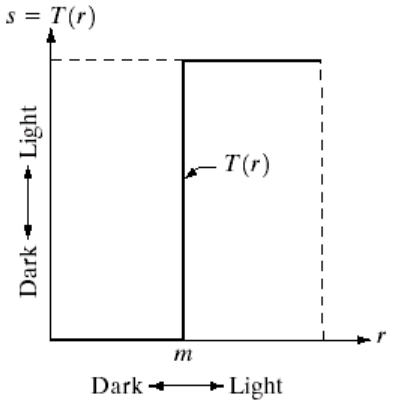


Fig. 39 Thresholding method

Dilation: Dilation is a basic morphological operator that enlarges the boundaries of the regions of foreground pixels (white pixels). Our implementation uses a dilation with structuring elements. The method takes a binary image B, places the origin of structuring element S over each 1-pixel, and ORs the structuring element S into the output image at the corresponding position.

In our case, we use it for two purposes. On one hand, it is used to exaggerate features in an image that would otherwise be missed. In particular, to recover weak detections at higher altitudes or partially occluded. In the maritime scenario, the contrast between the human and the sea is considerable, at least within a reasonable time frame, however, when flying above 40 meters above sea level, the size of the detection on the image becomes really small and this operator brings it back into the detection pipeline by increasing its size.

On the other hand, the result from the binarization is, in general, very cluttered. A dilation will fill gaps and glue together big areas, decreasing the amount of cluttering. The drawback is that in certain cases, a strong dilation can force a real victim to join a big area when it is in the proximity of structures. This will lead to a miss detection. The aggressiveness of the operation is configured through the parameter *DilatingFactor* and should be use with caution.

Labeling: This step is equivalent to a region-growing algorithm: labels the connected regions on a binary image. We rely on the implementation by Chang, Chen and Lu in [70]. The main step of this algorithm is to use a contour tracing technique to detect the external contour and possible internal contours of each component, and also to identify and label the interior area of each component.

Filter by size: This step discards regions larger than a certain threshold configured by the parameter *MaxObjectSize*. The size of an object on the image depends on the altitude at which the MAV is flying. Given the foreseen flight profile the characteristics of the camera and the approximate known size of the human, the operator can set the appropriate threshold. Depending on the orientation of the platform and the scene, areas at the horizon (where we do not want to detect yet) appear on the image with intensities similar to the human bodies. This big areas are discarded here.

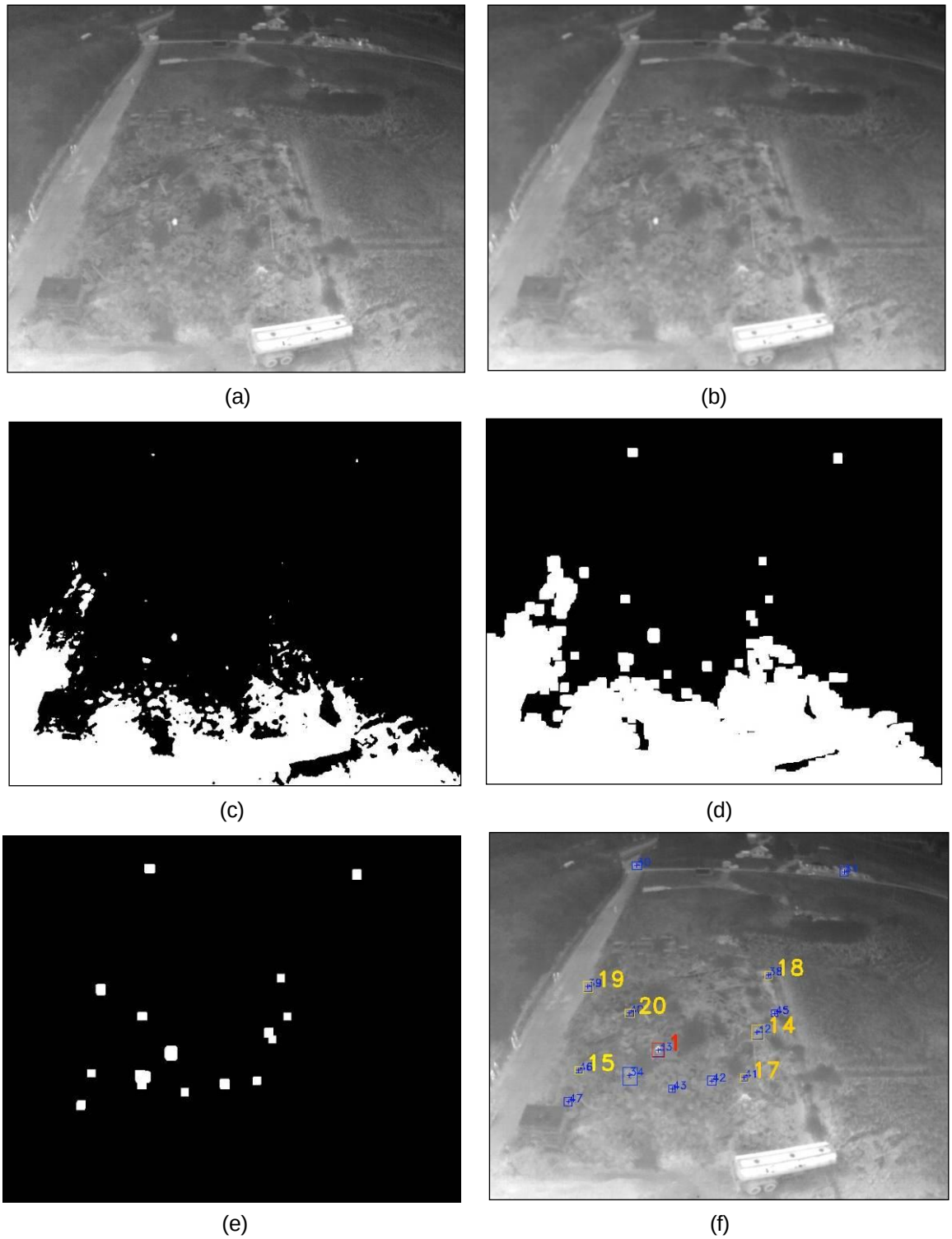


Fig. 40 Victim detection pipeline:
a) normalize image,
b) smoothed image,
c) binarization,
d) dilated,
e) small areas,
f) detections overlay

5.1.4. Victim tracker

Since we have a stream of images at high update rate, we can do tracking on top of pure detection to provide better performance [51] [36]. This tracking is done in two spaces: image and world coordinates.

Since we are operating on binary images and the shape of the region may change with the flight, there is no possibility to do feature matching by appearance or other techniques. Both trackers are based on distance matching.

Image-space tracker: Blob detections are tracked over multiple images. A blob is considered a track if the current pixel coordinates are within a given distance from the previous ones. The value is configured through the parameter *MinimumObjectLifetime* and it is also influence by the velocity of the platform. A blob that has been tracked over the minimum number of frames is considered a detection.

Direct geo-referencing of detections: To estimate the coordinates of the victim, the 2D pixel coordinate is projected to 3D. This projection is up-to-scale, meaning that the projection is a ray, and by using only one detection, we cannot estimate the distance to the victim. Multiple detections could have been used to estimate the point of intersection of several rays. Since we work with uncertainty, this intersection is unlikely to occur, therefore, robust estimation methods could be applicable.

In our case, an assumption made to reduce the computational workload of the algorithm is to assume that the ground is flat and use the current relative altitude of the MAV (altitude above the take-off location) to get rid of the scale ambiguity.

These coordinates are still in robot frame and they need to be translated to local first, and global coordinates later. The following scheme shows the coordinates transformations required to compute the location of the victim:

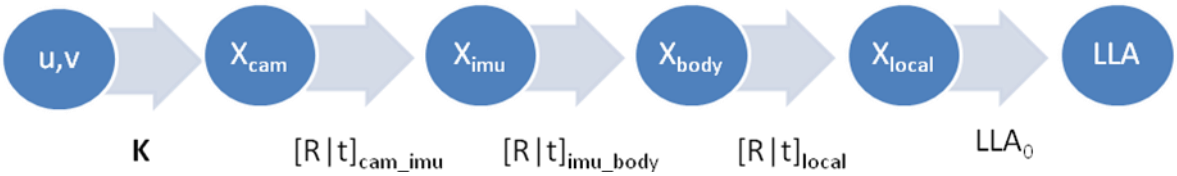


Fig. 41 Concatenation of coordinates transformation to georeference the victim

Position-space tracker: A second tracker implements the same principles but this time over the calculated absolute coordinates of the victim. This allows multiple detections of the same victim at different moments in time to be associated. The estimated position is uncertain, subject to noise and the accumulated errors in the platform state estimation, detection errors,

latency, etc. A standard linear Kalman filter estimates the location and velocity for each victim:

The state vector is composed by the position and velocity of the system:

$$x = \begin{bmatrix} x \\ y \\ vx \\ vy \end{bmatrix} \quad \textit{(Eq. 5.6)}$$

The measurement vector is composed by the estimated coordinates of the victim:

$$z = \begin{bmatrix} x \\ y \end{bmatrix} \quad \textit{(Eq. 5.7)}$$

The prediction and update equations are as follows:

$$x'(k) = A * x(k-1) + B * u(k) \quad \textit{(Eq. 5.8)}$$

$$x(k) = x'(k) + K(k) * (z(k) - H * x'(k)) \quad \textit{(Eq. 5.9)}$$

Where the state transition matrix is the one of a first order system:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textit{(Eq. 5.10)}$$

The control matrix (B) is 0 since this system is not actuated. The measurement matrix (H) is the identity (I). The process noise covariance matrix (Q) and the measurement noise covariance matrix are initialized to the values σ_p = 0.1 and σ_n = 0.0001 respectively, what makes the filter trust more the measurements at the beginning. The rest of the equations as for a standard Kalman filter are as follows.

The priori error estimate covariance matrix:

$$P'(k) = A * P(k-1) * At + Q \quad \textit{(Eq. 5.11)}$$

The posteriori error estimate covariance matrix:

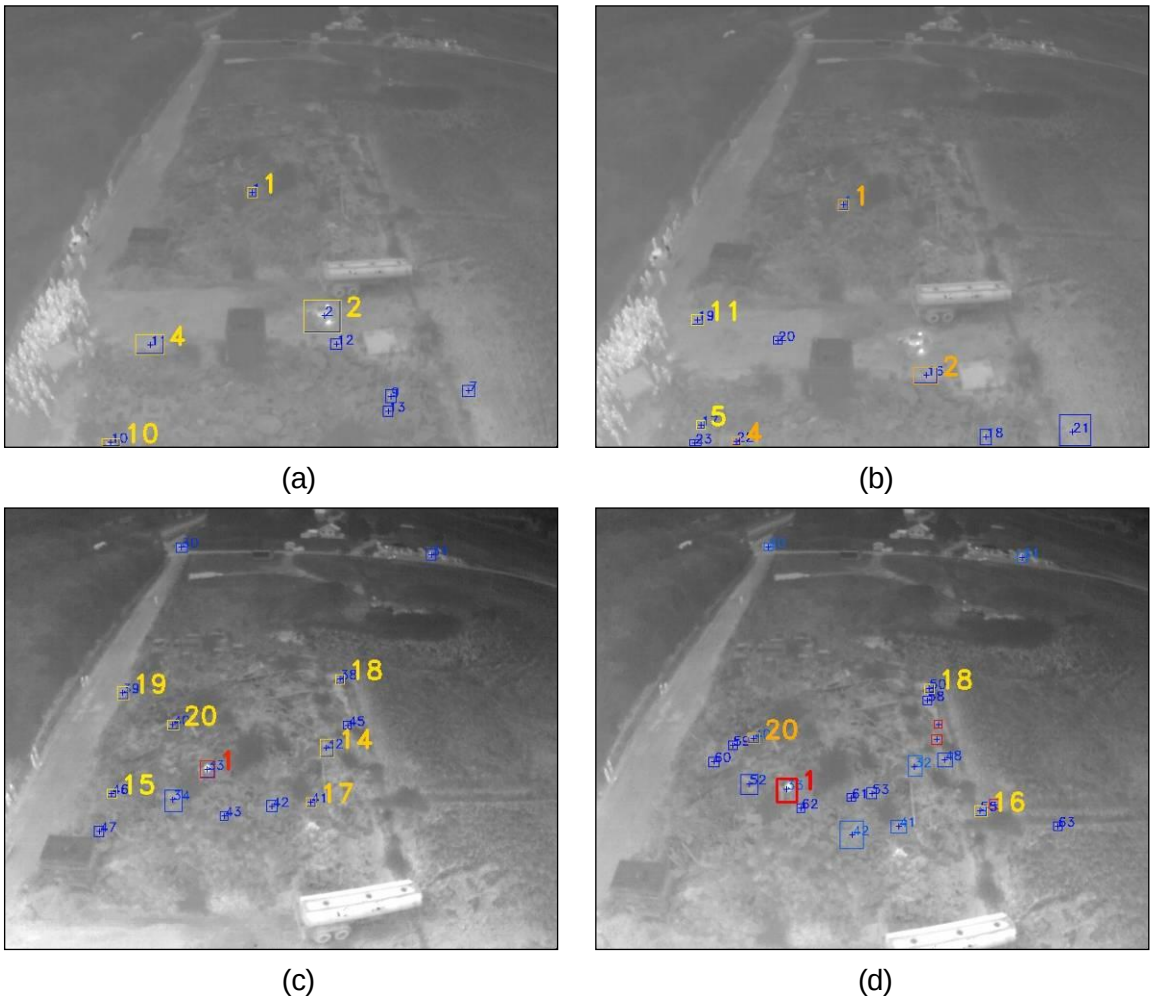
$$P(k) = (I - K(k) * H) * P'(k) \quad \textit{(Eq. 5.12)}$$

The Kalman gain is therefore:

$$K(k) = P'(k) * Ht * inv(H * P'(k) * Ht + R) \quad \textit{(Eq. 5.13)}$$

A detection is matched and therefore tracked if the estimated global coordinates fall within a given threshold. We used the Swift Nav library to compute the distance [71]. The distance threshold value is configured through the parameter *MaximumRelativeDistance*.

Therefore, a probabilistic estimator such as a Kalman filter is required to minimize the estimation errors and ensure the feasibility of a position-based tracker. Furthermore, the filter is able to estimate the location and velocity of the victim which has multiple applications in this project. If a victim is moving or running and it is temporarily occluded by trees, for instance, the algorithm will still be able to track it once it is back on the image. In the maritime scenario, for instance, there are strong currents that can cause the victim to drift. This estimator will enable the tracking in such scenario.



(a) (b)

(c) (d)

Fig. 42 Victim tracking pipeline
a) multiple detections (yellow),
b) some are tracked (orange),
c) only one is tracked over the threshold and the detection is confirmed,
d) continue tracking

5.1.5. Flight profile

The performance of the victim detection depends on the altitude profile of the mission. This also varies from one scenario to another. In general terms, the maximum altitude at which we could detect victims was 45 meters. This is mainly due to the resolution of the sensor. At this altitude, a victim is represented by few pixels and looks almost as noise. At lower altitudes, the detection is much more reliable, but the scanning of the given area, ensuring a minimum overlap to avoid missing victims, is much slower. In general terms, an altitude of 20 to 25 meters has been a good compromise between detection performance and speed.

5.1.6. Conclusions

The algorithm described here is a rather simple hot-spots detector that turned out to be very powerful in the field. It relies on a particular approach to the normalization of the image optimized for the purpose of the detection. In the scenarios where we tested, humans were in general warmer than the surrounding. Even in challenging scenarios, for instance flying over concrete at noon, the victims were still detectable.

The advantage of the algorithm is the simplicity of use and the reasonable performance showed on the field trials. This algorithm has proven to be easy to use by the operators and powerful. It allows very simple tuning during field operations. Furthermore, using the adaptive mode, this tuning is not even needed in most cases.

It does obviously not differentiate whether the blob is a person and may show some false positives. In line with the ICARUS concept of operation, it serves as an aid to the operator who is responsible to visually confirm the detection. More sophisticated algorithms overcome this problem by implementing classifiers such as decision-trees or combining visual and thermal images. These algorithms are typically supervised, require therefore training data sets, which is not always feasible for Search and rescue in unknown scenarios and are more complex to set up and tune (or train). Of course, if the number of false positives becomes too high, it would create a considerable work overload on the operator and this should be avoided. Even thought this has not been experience during the experimental tests, it is subject to future works.

6. Experimental results

The developments of this this Master's Thesis have gone from theory to practical implementation and realistic validation. A high-performance MAV has been developed that has been validated in realistic search and rescue scenarios. The content of this section is the result of over forty days of field work in three different locations: the Portuguese Navy School in Alfeite (Lisbon, Portugal), Camp Albert Roi in Marche-en-Famenne (Belgium) and the euRathlon competition in Piombino (Italy).

6.1. Maritime trials

These experiments were carried out during the final demonstration of the maritime scenario of the ICARUS project, from the 26th June to 11th July 2015. A shipwreck, was simulated at Alfeite naval base, near Lisbon, Portugal, similarly to the Costa Concordia disaster. An ICARUS team of unmanned surface vehicles and unmanned aerial vehicles helped the crisis managers in locating and providing immediate support to human survivors. This illustration depicts the large-scale demonstration scenario:

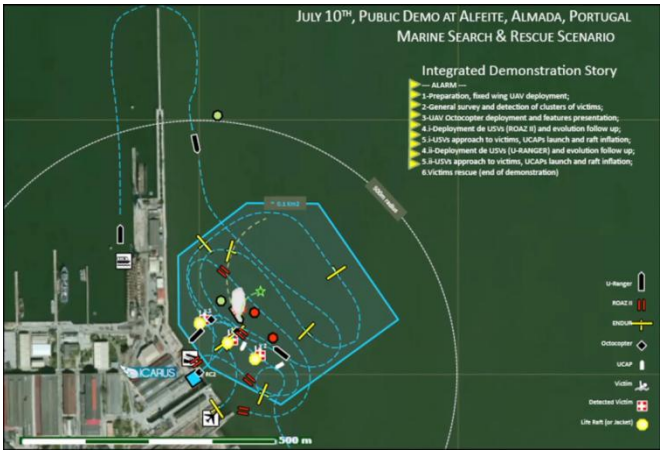


Fig. 43 Maritime Search and rescue scenario

Up to 9 victims were placed in water in unknown locations and grouped in several clusters of unknown size. The only information available at the beginning of the mission was the approximate location of the boat.

Fig. 44 Dummy victims going into water

An objective for these experiments was to explore the cooperation between the different type of robots. In a multi-stage approach, a fixed-wing and our multirotor cooperate in the victim search tasks.

Fig. 45 Collaborative aerial disaster assessment

The first aerial assessment of the area was provided by a solar fixed-wing developed by the ASL at ETH Zurich [73][74]. This first flight confirms the location of the boat and approximate location of the survivors but the flight altitude is too high to get accurate estimation. Therefore, our multirotor is asked to assess the area, search for victims, and provide live feed to the command and control station.

Flight operation

This operation was a combination of different autonomy modes:

- Given an initial approximate location and time of the disaster, the estimated area to cover is guessed. The first intervention was a scanning mission around the boat. This mission was executed completely autonomously. The data link was constant and the operator would just look at the screen to ensure safety.

- After the different clusters of victims were identified, the intervention of the marine robots could be planned. Since there was an isolated victim, the operator requested a waypoint (semi-autonomous) and the platform flew towards the victim.

- The deployment of the lifejacket is a delicate operation since you want to descent to avoid displacement caused by wind, etc. The operator took manual control and teleoperated looking at the live image feed.

Mapping

As a general consensus, mapping on water is not feasible. There is not enough texture to match the images. Therefore, mapping was not tested on water during these experiments.

Victim Search

The following image shows the trajectory executed to scan the area around the boat (bottom) and both live feed in thermal and grayscale:

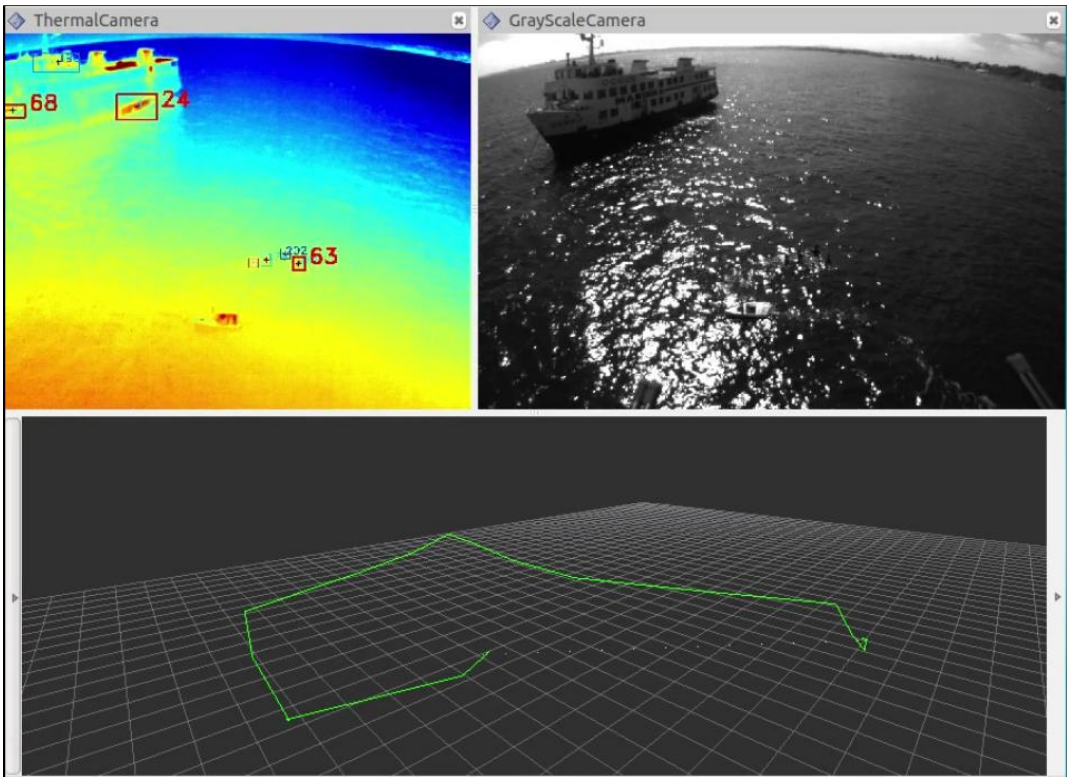


Fig. 46 MAV Copilot User Interface

This figure shows the importance of proper image normalization. In the grayscale image, the unmanned capsule and the victims are hardly visible. On the contrary, the thermal camera can nicely show it. On the thermal, the victim detection algorithm has picked up a victim and reported automatically its location to the GCS:

Fig. 47 ICARUS C2I showing the trajectory for the quadrotor and the different victim location detected by all robots

Support to rescue

During the execution of all this mission, the quadrotor remains airborne to provide a real-time tracking of the victims and live image feed. The following image shows a moment where ROAZII is rapidly approaching the victims while the quadrotor hovers above them:

Fig. 48 Victim tracking for collaborative rescue in the water

Delivery Kit

As described above, the following images show the delivery of the lifejacket:

Fig. 49 MAV equipped with a lifejacket

Fig. 50 Lifejacket drop to support a victim in the water

6.2. Land trials

The final field trials took place in Marche-en-Famenne (Camp Roi Albert) in Belgium. The tests were carried out from the 24th of August to the 4th of September 2015. For these final field trials, an earthquake was simulated to show how an ICARUS team of unmanned ground and aerial vehicles can assist the B-FAST Search and rescue (SAR) teams.

The demonstration included three different operations located in the following two main scenarios:

- A rubble field simulating a collapsed building.

Fig. 51 Rubble field on the ICARUS land demo

- A group of unstable buildings simulating a chemical warehouse and a demolished

school.

Fig. 52 School area on the ICARUS land demo

The demonstration simulates an earthquake in an unknown area. The collaborative aerial assessment starts by launched the solar fixed-wing and obtain an initial overview of the disaster. From this data, two different sectors are identified: a demolished block of apartments and a school building.

Fig. 53 Operations area

Mapping

The quadrotor is appointed to inspect the rubble field area. An unknown number of victims are located in the rubble.

Fig. 54 Aerial view of the rubble field

Mapping and victim search are driven by two different performance parameter (speed and accuracy) that lead to two different flight profiles. Therefore, in a realistic scenario, the first mission is typically mapping since more information is going to be acquired in less time, flying at a higher altitude. An initial mapping mission is first planned at an altitude of 30 to 60 meters. The victim search is also attempted over this flight, but the results are not as reliable as flying lower. The area to cover is around 5 hectare. The following flight plan shows a mapping mission for this scenario and takes less than 5 minutes:

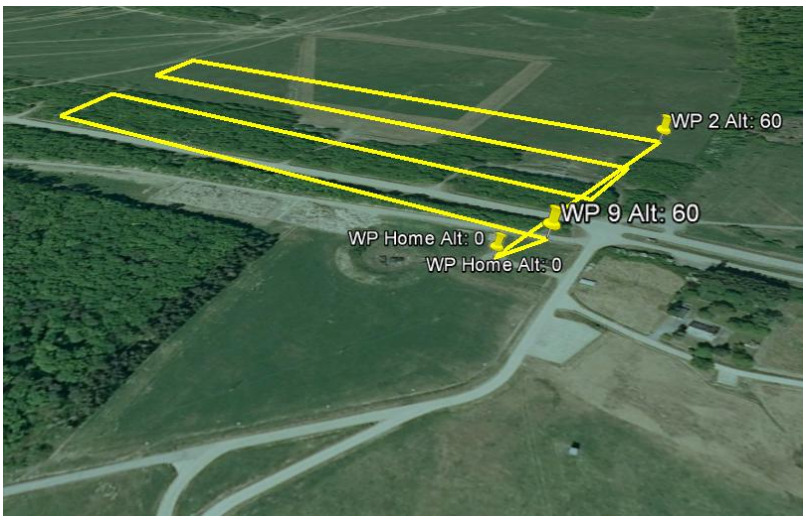


Fig. 55 Aerial mapping flight plan

The map resulting from this flight is the following:

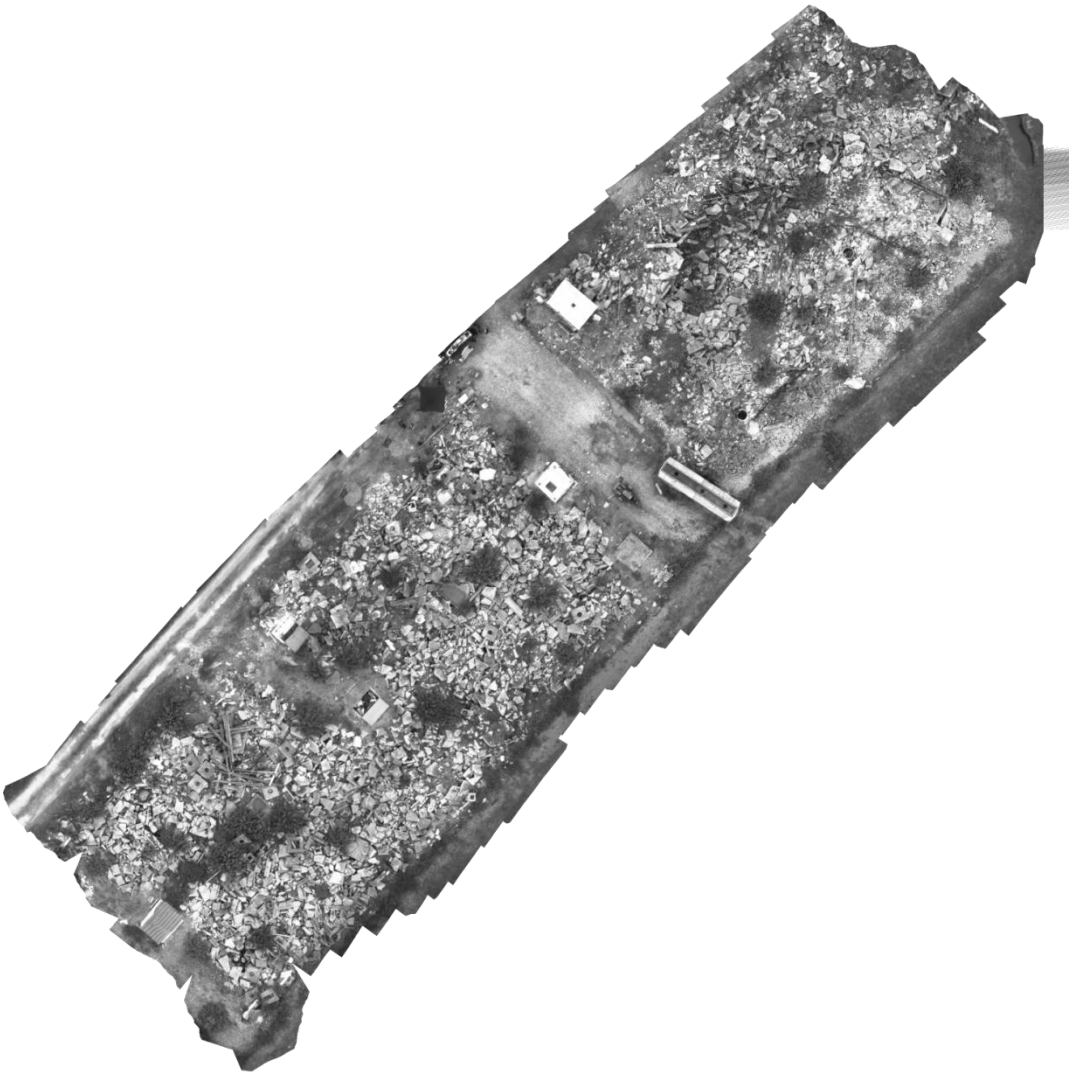


Fig. 56 Aerial map of the rubble field

A second flight for victim search at a lower altitude was planned. The plan was execute in complete realistic conditions. Only one victim was detected. Afterwards, the operator was informed that there were three victims on the rubble field and this victim was the only one that could be seen from the air (the other victims were detected by the canine team and ground robots).

The following images show how the victim looks like in gray scale (hardly visible) and thermal:

Fig. 57 Victim (center of the image) hardly visible on the gray

Fig. 58 Victim clearly visible on the normalized thermal mage

Furthermore, the victim is also visible on the georeference map:

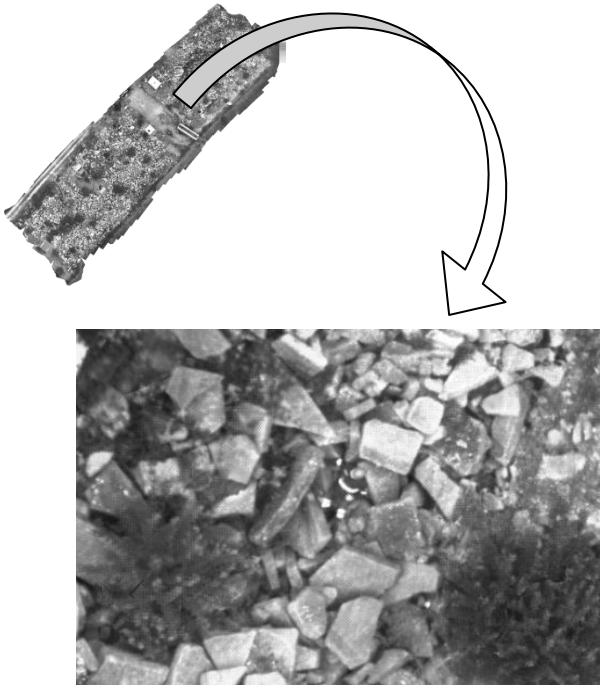


Fig. 59 Zoom in the map on the victim location

Large-scale highly-realistic mapping

As a side experiment, an attempt to do large-scale mapping in the three bands: color, grayscale and thermal was carried out. The area of operation was divided in four different sectors. Four flights with high-resolution cameras and the Copilot were perform. The following 2D and 3D maps are generated with commercial software to evaluate the performance of aerial mapping with multirotors UAVS:

Fig. 60 Results of post-processing aerial images (Software used: Agisoft Photoscan)

Fig. 61 Large-scale map (Software used: Agisoft Photoscan)

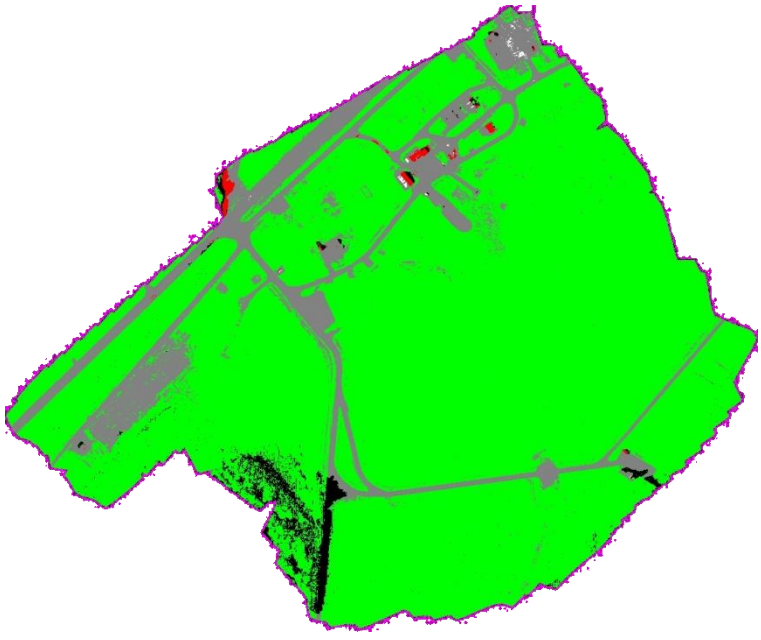


Fig. 62 Map Classification

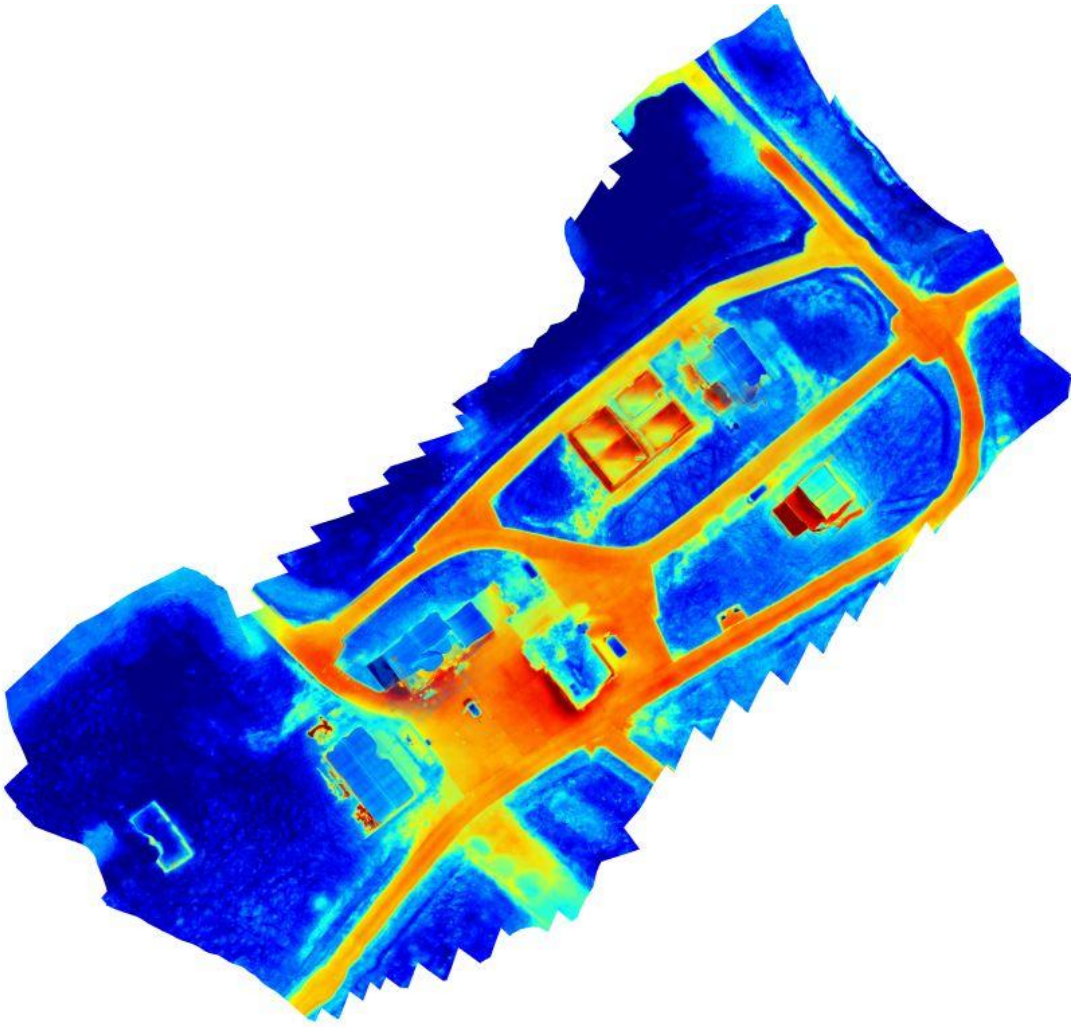


Fig. 63 Thermal Maps (false coloured) of the school area at night (30 meters altitude)

6.3. euRathlon 2015

EuRathlon is the first multi-domain outdoor robotics competition where teams test and demonstrate the intelligence and autonomy of their robots in realistic emergency-response scenarios. Inspired by the 2011 Fukushima accident, the euRathlon competition requires land, underwater and flying robots to work together, survey the scenario, collect environmental data and identify critical hazards.

The author was the team leader of the ICARUS party competing. Both for the author and the rest of the project, this was an opportunity to test developments in a scenario that was completely unknown. The competition was held in Piombino (Italy) from the 17th to 25th September 2015.

The objectives of these challenges were the following:

- Inspect the inside and outside of a building with the UGV(s) and UAV(s) to:
  - Find entrances and blocked paths to the building.
  - Find a safe path to a machine room inside the building for the land robot.
  - Search a missing worker in the area of operation

- Build a map of the different areas (outside the building, inside the building and underwater area) with references to the detected OPIs (Objects of potential interest).
- If possible, transmit live position and imagery to the control station.

A particularity of the challenge was that the control stations were completely locked, without line of sight to the platform. Therefore, the concept of MAV Copilot was a key advantage. A relaxation compare to the previous experiments was that the teams were given two hours for post-processing and were evaluated by precision, rather than time. Therefore, the same platform, sensors and copilot were used. Data capturing and georeferencing was the same. But for the processing of data, based on the knowledge generate by the own implementation of the mapping algorithm, an optimized automated mapping pipeline using commercial software was used.

The following images show the mapping results from the trials:

Fig. 64 Aerial 3D map of the disaster area (color)

Fig. 65 Aerial 3D map of the disaster area (gray)

As described in section 4.6, collaborative mapping between UGV and UAVs was demonstrating in conjunction with the IMM and RMA partners:

Fig. 66 Ground (top) and Aerial (bottom) 3D maps

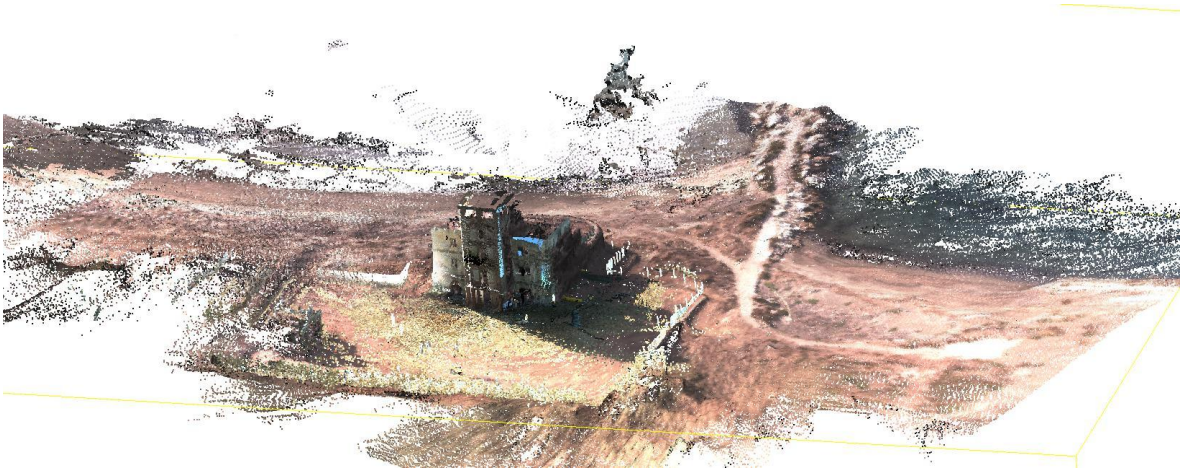


Fig. 67 UAV + UGV fused map (gray)

The performance of the UAV platform enabled the team to get the Texas Instrument award for Best Innovation on the UAV Domain. As a multi-domain team, the IEEE Robotics and Automation Society (RAS) Best Autonomy Award and IEEE Robotics and Automation Society (RAS) Best Multi-Robot Coordination Award were received.

7. Impact

The following achievements are directly related to the developments of this project:

Publications:

- D. Serrano, M. Uijt de Haag, E. Dill, S. Vilardaga and Pengfei Duan, "*Seamless Indoor-Outdoor Navigation for Unmanned Multi-Sensor Aerial Platforms*". The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-3/W1, 2014

- H. Balta, J, Bedkowski , S. Govindaraj , K. Majek , P. Musialik, D. Serrano, K. Alexis, R. Siegwart, G. De Cubber. "*Combined aerial and terrestrial 3D mapping for improving the situational awareness in search and rescue operations*"., Journal of Field Robotics (JFR) 2016.

Conference Presentations:

- D. Serrano. " *Final Project Demonstrations*" Purdue University. IEEE International Symposium on Safety, Security, and Rescue Robotics - SSRR 2015

Awards:

- The author was Team Leader of the ICARUS team competing in the Grand Challenge in euRathlon 2015 in Piombino (Italy):
    - Eurecat team received the winner award of the Texas Instrument UAV Innovation Prize for their developments in the UAV domain.
    - ICARUS team received the Best euRathlon IEEE RAS Security Safety and Rescue Robotics Autonomy (SSRR) Award and the Best euRathlon IEEE RAS Multi-Robot Coordination Award.

Conclusions

A multi-sensor estimation strategy for seamless indoor/outdoor navigation has been developed and demonstrated on an experimental test. The system relies on inertial navigation as the core sensor and sets up the required framework for the integration of secondary sensors. Equations for a GNSS receiver, ranger scanners and stereo and monocular cameras have been described. This work has been published in IARPS'14.

An algorithm for fast aerial mapping using an Structure from Motion approach has been developed and validated. The system emulates a standard terrain modelling pipeline but implements some optimizations and assumptions to speed up the process. After pair-selection, image stitching and bundler adjustment, the system assumes a flat ground and computes the required perspective transformation to generate a georeferenced map. A contribution from this work is the cooperative mapping of MAVs and UGVs. This work has been published in JFR'16.

Real time detection and tracking of persons based on thermal images has been developed and demonstrated. The system has been designed to be easy to use by the operators and therefore does rely on the interaction of the user.

The project has demonstrated an integral development of the prototype of a MAV and the Copilot payload concept. The system is functional and ready. It has served as development platform for the targeted algorithms and has been validated in multiple field trials. This component should be a key enabler for future research and developments.

An added value of the project has been the extended field validations. Three large-scale demonstrators have been carried out during Summer 2015. A contribution of particular interest has been the exploration of the potential multi-robot collaboration. During these field trials several collaborative scenarios involving the MAV and the Copilot were tested: multi-stage aerial assessment (fixed-wing + multirotor UAVs), maritime victim tracking and supportive rescue (MAV and USV), collaborative 3D mapping (MAV and UGV)

Some interesting future developments would be:

- Indoor/outdoor MAV navigation exploiting the sensors on the MAV copilot and the our multi-sensor framework.
- Autonomous target tracking and take-off and landing on fix or mobile platforms in remote areas exploiting the real-time detection capabilities explored in this project.
- Autonomous inspection of infrastructures
- Explore possibilities for better human-MAV interaction with augmented visualization and immersive technologies exploiting the capacities of the MAV copilot.

Annex I: Glossary

Acronyms & Definitions	
COTS	*Commercial of-the-shelf*
C2I	*Command and Control Interface*
FFC	*Flat Field Correction*
FOV	*Field of View*
GCS	*Ground Control Station (operator interface)*
GNSS	*Global Navigation Satellite System*
GPS	*Global Positioning System*
IMU	*Inertial Measurement Unit*
LWIR	*Long Wavelength InfraRed*
MAV	*Mini Aerial Vehicle (<25Kg)*
RTF	Ready To Fly
ROS	*Robot Operating System (see www.ros.org)*
SaR	*Search and rescue*
SotA	*State of the Art*
UAV	*Unmanned Aerial Vehicle*
UGV	*Unmanned Ground Vehicle*
USAR	*Urban Search and rescue*
USV	*Unmanned Sea Vehicle*
w.r.t	*With respect to*

Annex II: Bibliography

[1] R. Murphy. "Disaster Robotics" [Book]. - 2015

[2] Erdos, D., Erdos, A., and Watkins, S. E. "*An experimental uav system for search and rescue challenge*". IEEE A&E Systems Magazine. 2013

[3] De Cubber, G., Doroftei, D., Serrano, D., Chintamani, K., Sabino, R., and Ourevitch, S. "*The EU ICARUS project: Developing assistive robotic tools for search and rescue operations*". In Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on, pages 1-4.

[4] G.De Cubber, D. Doroftei, D. Serrano, K. Chintamani and R. Sabino. "*ICARUS: Providing Unmanned Search and rescue Tools*". 6th IARP Workshop on Risky Interventions and Environmental Surveillance (RISE). 2012 .

[5] Kobayashi, A. , and K. Nakamura. "*Rescue robot for fire hazards*". In Proceedings of the 1983 International Conference on Advanced Robotics, pp. 91-98. Tokyo, Japan.

[6] NIFTi FP7 website: http://www.nifti.eu/

[7] ICARUS FP7 website: http://www.fp7-icarus.eu/

[8] Sherpa FP7 website: http://www.sherpa-project.eu/

[9] TRADR FP7 website: http://www.tradr-project.eu/

[10] euRathlon FP7 website: http://www.eurathlon.eu/

[11] G.-J. Kruijff, V. Tretyakov, T. Linder, F. Pirri, M. Gianni, P. Papadakis, M. Pizzoli, A. Sinha, E. Pianese, S. Corrao, F. Priori, S. Febrini, and S. Angeletti, "*Rescue robots at earthquake-hit mirandola, italy: a field report*," in IEEE International Symposium on Safety, Security and Rescue Robotics, pp. 1–8.

[12] Farrell, J. L., "*GNSS Aided Navigation & Tracking – Inertially Augmented or Autonomous, American Literary Press*", 2007.

[13] M. M. Miller, J. Raquet, M. Uijt de Haag, "*Navigating in Difficult Environments: Alternatives to GPS*," Proceedings of the NATO RTO Lecture Series on "Low Cost Navigation Sensors and Integration Technology," SET-116, November 2008.

[14] Pfister, S. T., "*Algorithms for Mobile Robot Localization and Mapping Incorporating Detailed Noise Modeling and Multi-scale Feature Extraction*," Ph.D. Dissertation, California Institute of Technology, 2006.

[15] Soloviev, A., D. Bates, and F. van Graas, "*Tight Coupling of Laser Scanner and Inertial Measurements for a Fully Autonomous Relative Navigation Solution*," NAVIGATION, Journal of the Institute of Navigation, Vol. 54, No. 3, Fall 2007, pp. 189-205.

[16] Grisetti, G., C. Stachniss, W. Burgard, "*Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling*," Proc. Of the IEEE Intl. Conf. on Robot. and Autom., Barcelona, Spain, April 2005, pp. 2432-2437.

[17] Rusinkiewicz, S. and M. Levoy, "*Efficient variants of the ICP Algorithm,*" Proc. Of Intl. Conf. on 3-D Digital Imaging and Modeling, Quebec City, Quebec, May 2001, pp. 145-152.

[18] Horn, J. P., "*Bahnführung eines mobilen Roboters mittels absoluter Lagebestimmung durch Fusion von Entfernungsbild- und Koppelnavigations-daten*," Ph.D. Dissertation, Technical University of Munich, 1997.

[19] Uijt de Haag, M., D. Venable, and M. Smearcheck, "*Use of 3D laser radar for navigation of unmanned aerial and ground vehicles in urban and indoor environments*," Proceedings of the SPIE - Volume 6550, SPIE Defense and Security Symposium, Orlando, FL, April 9- 13, 2007.

[20] Nüchter, A., "*3D Robotic Mapping – The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*", Springer, 2010.

[21] Soloviev, A. and M. Uijt de Haag, "*Three-Dimensional Navigation of Autonomous Vehicles Using Scanning Laser Radars: Concept and Initial Verification*," IEEE Transactions on Aerospace and Electronic Systems, Vol. 46, Issue 1, 2010.

[22] S. Kohlbrecher, O. von Stryk, J. Meyer, U. Klingauf, "*A Flexible and Scalable SLAM System with Full 3D Motion Estimates,*" Proc. Of IEEE Conference on Safety, Security, and Rescue Robotics, 2011.

[23] S. Shen, N. Michael, V. Kumar, "*Autonomous Multi-Floor Indoor Navigation with a Computationally Constrained MAV,*" Proc. Of IEEE Conference on Robotics and Automation, 2011.

[24] S. Grzonka, G. Grisetti, W. Burgard, "*A Fully Autonomous Indoor Quadrotor*," IEEE Transactions on Robotics, Vol. 28, Issue 1, 2012.

[25] S. Lynen, M. Achtelik, S. Weiss, M. Chli and R. Siegwart. "*A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation*". IROS 2013

[26] S. Weiss, M. Achtelik, M. Chli, R. Siegwart. "*Versatile Distributed Pose Estimation and Sensor Self-Calibration for an Autonomous MAV*". ICRA 2012.

[27] Ethzasl-msf: "*A EKF based modular-sensor fusion framework*". Website: https://github.com/ethz-asl/ethzasl_msf

[28] F. Remondino, L. Barazzetti, F. Nex, M. Scaioni, and D. Sarazzi, "*UAV Photogrammetry for mapping and 3D modeling - Current status and future perspectives*". International Conference on Unmanned Aerial Vehicle in Geomatics (UAV-g) (Volume XXXVIII-1/C22). 2011

[29] Christoph Strecha, Olivier Küng and Pascal Fua. "*Automatic Mapping from Ultra-Light UAV Imagery*". EuroCOW 2012, Barcelona, Spain, February 8-10, 2012.

[30] R. Hartley and A. Zisserman. "*Multiple View Geometry in Computer Vision*" [Book]. - 2000

[31] Bernhard Draeyer and Christoph Strecha. White paper: "*How accurate are UAV surveying methods?*". February 2014

[32] Lanny Lin, Michael Roscheck, Michael A. Goodrich, Bryan S. Morse. "*Supporting Wilderness Search and Rescue with Integrated Intelligence: Autonomy and Information at the Right Time and the Right Place".* Twenty-Fourth AAAI Conference on Artificial Intelligence. 2010.

[33] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "*Pedestrian detection: An evaluation of the state of the art*," in IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2011

[34] P. Viola and M. Jones, "*Rapid object detection using a boosted cascade of simple features,*" in Proceedings of the IEEE Conferenceon Computer Vision and Pattern Recognition (CVPR), 2001.

[35] Anna Gąszczak, Toby P.Breckon and Jiwan Han. *"Real-time people and vehicle detection from UAV imagery".* Proceeding of SPIE : Intelligent Robots and Computer Vision XXVIII : Algorithms and Techniques, 24-25 January 2011, San Francisco, California, US. Pages 78780B-1-13

[36] M. Andriluka, S. Roth, and B. Schiele, "*People-tracking-by-detection and people-detection-by-tracking,*" in Proceedings of the IEEE Conference on Computer Vision and Pattern

Recognition (CVPR), 2008

[37] K. Jüngling and M. Arens, "*Local feature based person detection and tracking beyond the visible spectrum,*" in Machine Vision Beyond Visible Spectrum. Springer Berlin Heidelberg, 2011, vol. 1, pp. 3–32, ISBN: 978-3-642-11567-7.

[38] Jan Portmann, Simon Lynen, Margarita Chli and Roland Siegwart, "*People Detection and Tracking from Aerial Thermal Views, Thermal imager*". IEEE International Conference on Robotics and Automation (ICRA). 2014.

[39] UAV Development Board and MatrixPilot firmware website: https://github.com/MatrixPilot/MatrixPilot/wiki

[40] APM Autopilot Board: http://copter.ardupilot.com/wiki/common-apm25-and-26-overview/

[41] Pixhawk Autopilot website: https://pixhawk.org/modules/pixhawk

[42] ArduCopter firmware website: http://copter.ardupilot.com/

[43] MAVLink website: http://qgroundcontrol.org/mavlink/start

[44] SeeTrack CoPilot brochure: http://www.seebyte.com/wp-content/themes/seebyte/pdfs/CoPilot.pdf

[45] Nikolic, J., Rehder, J., Burri, M., Gohl, P., Leutenegger, S., Furgale, P. T., and Siegwart, R. Y.. "*A Synchronized Visual-Inertial Sensor System with FPGA Pre-Processing for Accurate Real-Time SLAM*". In IEEE International Conference on Robotics and Automation (ICRA). 2014

[46] "*FLIR Tau2 Product Specification*", Rev 141

[47] Kalibr toolbox wiki: https://github.com/ethz-asl/kalibr/wiki

[48] Paul Furgale, Joern Rehder, Roland Siegwart . "*Unified Temporal and Spatial Calibration for Multi-Sensor Systems".* In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan. 2013.

[49] B. Siciliano and O. Kathib, "*Handbook of Robotics*", Springer-Verlag Berlin Heidelberg (Book) 2008

[50] ROS website: http://www.ros.org/

[51] D. Serrano, M. Uijt de Haag, E. Dill, S. Vilardaga and P. Duan, "*Seamless Indoor-Outdoor Navigation for Unmanned Multi-Sensor Aerial Platforms*". The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-3/W1, 2014

[52] E. Dill, M. Uijt de Haag, P. Duan, D. Serrano and S. Vilardaga, "*Seamless Indoor-Outdoor Navigation for Unmanned Multi-Sensor Aerial Platforms*". Position, Location and Navigation Symposium - PLANS 2014, 2014 IEEE/ION.

[53] Uijt de Haag, M, D.Serrano, J. Batlle, E. Dill, "*Integration of GNSS, Electro-optical Sensors and Inertial for Navigation using a Plug-and-play Architecture Based on the Robotic Operating System (ROS)*". Proceedings of the EuroCow 2012.

[54] Uijt de Haag, M., 2009. "*Inertial Navigation Course Notes",* Ohio University, Athens, Ohio.

[55] E. Dill, "*Integration of 3D and 2D Imaging Data for Assured Navigation in Unknown Environments*," M.S.E.E. Thesis, March 2011.

[56] J. Andrade-Cetto, A. Sanfeliu, "The Kalman Filter". Chapter Environment Learning for Indoor Mobile Robots Volume 23 of the series Springer Tracts in Advanced Robotics pp 119-125.

[57] Uijt de Haag, M., D. Venable, and M. Smearcheck "*Integration of an Inertial Measurement Unit and 3D Imaging Sensor for Urban and Indoor Navigation of Unmanned Vehicles*," in Proceedings of the Institute of Navigation National Technical Meeting 2007, Jan. 2007, pp. 829-840.

[58] SiftGPU by Changchang Wu: http://cs.unc.edu/~ccwu/siftgpu/

[59] D. G. Lowe. "*Distinctive image features from scale-invariant keypoints*" . International Journal of Computer Vision, November 2004.

[60] Bundler project by Noah Snavely: http://www.cs.cornell.edu/~snavely/bundler/

[61] Noah Snavely, Steven M. Seitz, Richard Szeliski. "*Photo Tourism: Exploring image collections in 3D. ACM Transactions on Graphics"* (Proceedings of SIGGRAPH 2006), 2006.

[62] Geotagging Images with Mission Planner http://copter.ardupilot.com/wiki/common-geotagging-images-with-mission-planner (Wiki)

[63] Frank Ivis. "*Calculating Geographic Distance: Concepts and Methods*" . NESUG 2006.

[64] M.I.A. Lourakis and A.A. Argyros. "*The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm".* Tech. Rep. 340, Inst. of Computer Science-FORTH, Heraklion, Crete, Greece. Available from http://www.ics.forth.gr/~lourakis/sba.

[65] G.A. Watson. "*Computing Helmert transformations*". Journal of Computational and Applied Mathematics Volume 197, Issue 2, 15 December 2006, Pages 387–394

[66] H. Balta, J, Bedkowski , S. Govindaraj , K. Majek , P. Musialik, D. Serrano, K. Alexis, R. Siegwart, G. De Cubber. "*Combined aerial and terrestrial 3D mapping for improving the situational awareness in search and rescue operations*", Journal of Field Robotics (JFR) 2016.

[67] S Kluckner, et al. "*Semantic Classification in Aerial Imagery by Integrating Appearance and Height Information*". Computer Vision – ACCV 2009 Volume 5995 of the series Lecture Notes in Computer Science pp 477-488

[68] C. Zhang and P. Wang, "*A New Method of Color Image Segmentation Based on Intensity and Hue Clustering*". International Conference on Pattern Recognition, ICPR 2000.

[69] Mahalanobis, Prasanta Chandra. "*On the generalised distance in statistics*". Proceedings of the National Institute of Sciences of India 2 (1): 49–55. (1936)

[70] Fu Chang, Chun-Jen Chen and Chi-Jen Lu. "*A linear-time component-labeling algorithm using contour tracing technique*". Computer Vision and Image Understanding Volume 93, Issue 2, February 2004, Pages 206–220.

[71] Swift Nav library website: https://github.com/swift-nav/libswiftnav

[72] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "*Collaborative mapping of an earthquake-damaged building via ground and aerial robots*," Journal of Field Robotics, vol. 29, no. 5, pp. 832–841, 2012.

[73] Atlantik Solar endurance UAV developed by the Autonomous Systems Lab: http://www.atlantiksolar.ethz.ch/

[74] P. Oettershagen et al. "*A Solar-Powered Hand-Launchable UAV for Low-Altitude Multi-Day Continuous Flight*". ICRA 2015.