# Human detection and recognition in visual data from a swarm of unmanned aerial and ground vehicles through dynamic navigation

**Marc Volger**

February 2015

**Master's Thesis**
Artificial Intelligence
University of Groningen, The Netherlands

Executed at the Machine Intelligence Laboratory,
University of Florida, United States of America

**Internal supervisor:**
Dr. M.A. Wiering (Artificial Intelligence, University of Groningen)

**External supervisor:**
Dr. E.M. Schwartz (Electrical & Computer Engineering, University of Florida)

university of groningen / faculty of mathematics and natural sciences / artificial intelligence

**Abstract**

One of the hot topics in current Artificial Intelligence research and in society are outdoor unmanned systems and their recent applications. Development in sensor output processing and computer vision is one of the main reasons for the rapid growth in the abilities of such systems to operate autonomously. Detecting and recognizing objects and humans has been a prominent subject in research since computer vision originated. Combining the field of outdoor unmanned systems with computer vision yields interesting new research topics. Reactive vehicle behaviors and possible human recognition opposed to solely detections from such systems is a fairly unexplored side of the scientific field.

The current research focuses on autonomous human detection and recognition in real-time sensory data from unmanned ground vehicles (UGV) and unmanned aerial vehicles (UAV) through dynamic navigation. Additional information and heightened perception can be gained by creating intelligent navigational behaviors combined with well performing object classifiers. More specifically, the autonomous vehicles in the architecture search for test subjects in a field and react upon those detections. If a person is detected in the camera imagery, a vehicle will dynamically stray off its initial search pattern to gain more information on the subject. The dynamic navigation is used to approach the subject and to attempt facial recognition using a data set of the test subjects. Through the deployment of a heterogeneous swarm of multiple UGVs and UAVs individual search spaces can be decreased and detection rates increased.

The research was built upon a software architecture called CongreGators that controls a swarm of autonomous vehicles. A complete system for the autonomous detection and recognition of human subjects through dynamic navigation with a heterogeneous swarm of autonomous agents was implemented and tested. Dynamic navigation patterns were created and optimized to increase the perception and information gain of the robotic systems at hand. The CongreGators architecture was created at the University of Florida's Machine Intelligence Laboratory, where the current research was conducted as well.

## Acknowledgments

This thesis and the current research has been made possible by the resources of the Machine Intelligence Laboratory of the Mechanical Aerospace Engineering department of the University of Florida. I would like to thank Dr. E.M. Schwartz and Dr. M.A. Wiering for their support, useful comments, and engagement throughout the entire period of the research. Dr. Schwartz his guidance and trust in me to take on the CongreGators project and become the head of one of the lab's main projects, has been a true inspiration and motivation.

Furthermore, I would like to thank A. Gray, A. Baylis, K. Tran, M. Langford, and K.M. Frey for their aid and support in the execution of the experiments. Also, I like to thank the participants in my experiments, who have willingly shared their time during the process of testing the modules.

Finally, I would like to thank my father, Dr. E. Otten, for being the greatest intellectual example, my mother, J.M. Volger, for showing everlasting support, and my sisters and brother for their motivation to keep pushing myself and their confidence in me.

# Contents

# Chapter One

# Introduction

Unmanned autonomous robotic systems are a very hot topic in today's society and these systems are becoming increasingly important and popular in the economy. The unmanned autonomous control of robotic systems has many benefits over full human control or task execution by humans, i.e. robotic systems have a more structured way of behavior, handle situations without putting humans at risk, and free up time and resources for human beings. Unmanned vehicles also often have the ability to reach locations which are unreachable or hazardous for human beings. These benefits are the reason why unmanned vehicles have gained such a great economical feasibility these days.

The autonomous detection and recognition of humans from sensory data of unmanned robotic systems can improve the use and performance of these systems significantly. Although the detection of humans in an outdoor environment by autonomous agents has been investigated, intelligent responding agent behaviors and actual human recognition (opposed to solely detection) is a much more unexplored side of the scientific field. The detection of a (human) object is defined as the classification of a newly seen object to be part of a predetermined class, while recognition is defined as classifying a new object to be a specific individual of that class that was seen before. The goal of the current research was to create a complete system for the autonomous detection, investigation and finally recognition of human subjects in an outdoor environment through dynamic navigation by a heterogeneous swarm of autonomous agents. The main research questions that drove the current research were which human classifying algorithm would perform best on raw video imagery from the available vehicles, could UGVs or UAVs use dynamic search patterns to autonomously approach detected subjects, and would they be able to recognize those subjects in a robust and reliable fashion from a created database. In order to create a behavior with the mentioned capabilities, three dependent modules were implemented in an existing architecture that handled the coordination of the swarm. These three modules were the human detection module, dynamic navigation module, and the facial recognition module, which all combined created a behavior achieving the previously mentioned goal.

The types of agents functioning in the swarm were "Unmanned Ground Vehicles"

(UGVs) and "Unmanned Aerial Vehicles" (UAVs). All hardware and resources were provided by the Machine Intelligence Laboratory of the University of Florida, where the current research was executed. To provide the agents with a setup to provide live video imagery, two cameras were tested in a benchmark test and the optimal camera setup was mounted on the vehicles. Using the existing 'CongreGators' architecture (Weaver, 2014) the agents could make practical use of the swarm property to increase effectiveness of the entire robotic system by deploying multiple vehicles at the same time, which leads to the agents splitting up search areas for faster completion of their search. The CongreGators architecture and all of the software for the current research was developed within the Robotic Operating System (ROS) (Quigley et al., 2009).

Towards the creation of the human detection module multiple algorithms were implemented, applied, and compared. The collection of classifiers for human detection consisted of four pedestrian detection classifiers using Haar-like features (Viola & Jones, 2001), one custom created top view Haar classifier, and one Histogram of Oriented Gradients pedestrian classifier (Dalal & Triggs, 2005). The custom created top view classifier was created for the detection of humans from a top view, i.e. the view from a UAV, since no such classifier was known to exist yet. All classifiers were compared in benchmark tests on four separate datasets on their performance.

To have the agents respond in an intelligent investigating fashion to human detections a dynamic navigation module was created. The dynamic navigation combined information from the vehicle and the detection into the localization of the subject and a dynamic approach pattern towards the subject. This dynamic approach pattern was executed by agents through diverting from their original static search paths to a dynamic approach of the subject. This approach would subsequently lead to checking whether the subject's face could be detected and recognized. To segregate false-positives from the detections and to increase 'confidence' in the agent's detections and subject localizations, a clustering algorithm was incorporated in the module alongside subject representation using prototypes.

To provide the agents with facial recognition capabilities, a facial recognition module was added to the system. This module was only activated after dynamic driving patterns were completed and was therefore heavily dependent on the previous detection and approach stages of the behavior. The facial recognition module is based on the work by Baggio et al. (2012), and modified to function within the current research. This module makes use of a combination of a Haar face detector, conversion to Eigenfaces (Turk & Pentland, 1991), and a support vector machine (Cortes & Vapnik, 1995; Joachims, 1998). The output of the facial recognition module consisted of a confidence value which represents the certainty of the module that a certain face was in view and the label accompanying that face. The model of the recognition classifier was created from a dataset of the test subject's faces.

To determine the performance of the modules towards the goals of the current research, two types of experiments were set up. The type of experiments differed in human subject search methods by the agents. One experiment type was set up to search for multiple human subjects in a search area to determine the overall performance of the behaviors, while not having a certainty that the subjects would appear in the camera's

view. The second type of experiment was set up to increase the certainty of a subject being in view of the camera by having an agent search along a straight path for one subject positioned on that path. The latter experiment type was created to test individual agent behavior performance, while including a higher certainty of the activation of the facial recognition module.

The scientific relevance for the field of Artificial Intelligence lies in the fully autonomous intelligent behaviors that are created for the agents to detect, dynamically approach, and even recognize human subjects in a noisy outdoor environment. Possible applications of the created system are to search, find and recognize people in search and rescue missions of victims in large or non-traversable areas for humans. Areas that are too hazardous for humans to enter could also be investigated by this system for human occurrences, and even specific human occurrences through the facial recognition capabilities. Recognizing and logging all the human subjects in an area for security purposes would be another application of the created system.

This thesis consists of the following chapters. The theoretical framework in which the current research is embedded is described in chapter 2. The description of the CongreGators architecture and its capabilities on which the current research builds is discussed in chapter 3. Chapter 4 includes the description of all the methods used for the three main modules created in the current research, namely human detection, dynamic navigation, and facial recognition. In chapter 5 the results from the classifier benchmark tests and experiments are discussed. And concluding, in chapter 6 all the conclusions drawn from the current research will be discussed along with proposals for future work.

# Chapter Two

# Theoretical Background

## 2.1 Swarm robotics

Reactive groups of robots were first studied at the end of the 1940's by Walter (1950) through observing the behaviors of light and touch sensor equipped turtle-like robots. The swarm robotics approach for the coordination of a group of robots is inspired by observations of group dynamics of social insects, e.g. ants or locusts. Swarm intelligence produces in many cases behaviors and solves problems, which an individual would not be able to perform by itself. Both centralized and decentralized types of swarm robotic systems have been implemented nowadays with different advantages per type (Beni, 2005). The main advantages of swarm robotics are robustness, flexibility, and scalability (Şahin, 2005), all even more advantageous if a heterogeneous swarm of agents is used.

Swarm intelligence in biological organisms can also be connected to object detection, recognition, and gaining additional information through dynamic navigation by such organisms. Biological organisms often use movement to collect additional information if the currently obtained information is not sufficient for their goal, e.g. locusts use body movement to obtain depth information for their jump (Sobel, 1990). Since the scientific field of robotics has often made use of mimicking biological organisms to its advantage in the past, this technique also seems to have potential for a performance increase in autonomous robotic systems.

The current research builds upon an architecture that is created to coordinate a swarm of heterogeneous autonomous ground and aerial vehicles (Weaver, 2014). Agents deployed in the architecture could search for objects along a path or within a search area, dividing such an area among the enrolled agents automatically.

## 2.2 Human detection

Since the 1990's the field of computer vision has gained an increasing interest in the detection of human beings as an object (Gavrila, 1999). The process of object detection often involves the extraction of image features from datasets and matching those features with the ones of new images. One of the most famous and widely used features for

detection were proposed by Viola and Jones and are called Haar-like features (Viola & Jones, 2001), which owe their name to their intuitive similarity with Haar wavelets. Human detection is basically similar to the detection of any other object in most cases since meaning is unimportant to an algorithm. For example, detection techniques that train on a database of positive (and negative) examples for object detection work the same for every object. Thus, such a technique can also be used for human detection. However, one of the foremost problems with human detection is that the appearance of humans in general changes very often, both in shape and in color. Therefore classifiers have been tailored to perform human detection like specified Histogram of Oriented Gradients (HOG) classifiers (Dalal & Triggs, 2005).

Developing computerized person detection in the past was primarily powered by pedestrian detection and avoidance in on-board systems of automobiles (Breckon, Han, & Richardson, 2012; Papageorgiou & Poggio, 1999), human detection in sensor output from unmanned aerial vehicles (UAVs) (Rudol & Doherty, 2008), and person detection in surveillance cameras for security purposes (Viola, Jones, & Snow, 2003). On a side note, other objects related to human presence like cars or windows with people behind them, can be detected by using various similar object detection techniques as well (Breckon, Gaszczak, Han, Eichner, & Barnes, 2013; Gaszczak, Breckon, & Han, 2011). In some cases general detection methods like the Scale-invariant feature transform (SIFT) algorithm (Lowe, 1999) are used to detect objects or humans, which was subsequently used to dynamically navigate through the surroundings of the recognized object (Mondragon, Campoy, Correa, & Mejias, 2007; Campoy et al., 2009).

### 2.2.1 Human detection with unmanned aerial vehicles

Human detection is sometimes applied on sensor output like video, infrared, or thermal imagery from UAVs. Rudol and Doherty booked some great results in geo-localizing victims with regular and thermal imagery from an UAV, using a "classifier which is in fact a cascade of boosted classifiers working with Haar-like features" (Rudol & Doherty, 2008). Leira has written a Master's thesis on the comparison between a Boosted Cascade Haar-like classifier and a Histogram of Oriented Gradients Support Vector Machine (HOG-SVM) classifier for object detection and tracking in UAV infrared imagery (Leira, 2013). Flynn and Cameron have fused visible and infrared imagery and shown that by "tracking detections over time, the false positive rate is reduced to a minimum" (Flynn & Cameron, 2013). This study shows that multiple models can be used to detect one object with higher performance, opposed to Breckon et al. who use multiple models to detect several different objects in visual imagery (Breckon et al., 2013; Gaszczak et al., 2011). For human detection in UAV imagery Andriluka et al. compared 'monolithic models', like HOG detectors, with 'part-based models', like poselet based detection algorithms, discriminatingly trained part based models, and pictorial structures with discriminant part detectors (Andriluka et al., 2010).

## 2.3 Facial recognition

Facial recognition within the computer vision research field has gained a lot of popularity since 1990. Techniques based on Karhunen-Loeve expansion, neural networks, and feature matching have been widely investigated since then (Chellappa, Wilson, & Sirohey, 1995). Human face detection and the recognition of such a face in particular are valuable assets to a robotic system in any environment involving human beings. It is very important that such a facial detection and recognition module is robust and performs adequately. Facial recognition has widespread applications in commercial use and law enforcement and is powered by about 25 years of research in the scientific field of robotics and processing techniques for sensory data. Zhao et al. stated "Even though current machine recognition systems have reached a certain level of maturity, their success is limited by the conditions imposed by many real applications. For example, recognition of face images acquired in an outdoor environment with changes in illumination and/or pose remains a largely unsolved problem. In other words, current systems are still far away from the capability of the human perception system." (Zhao, Chellappa, Phillips, & Rosenfeld, 2003).

Nowadays facial detection (and sometime recognition) is often available on digital devices like cameras (Ray & Nicponski, 2005) and smartphones (Chun & Maniatis, 2009). Many parts of the human body, with facial features in particular, have been targeted for detection by computer vision. Studies on the detection of elements of the face have been performed like eye-pair detection from visual imagery (Karaaba, Schomaker, & Wiering, 2014; Jee, Lee, & Pan, 2004) and ear detection in images of faces (Chen & Bhanu, 2004; Islam, Bennamoun, & Davies, 2008). A ROS based facial recognition module was implemented by Baggio et al., which will be used and build further upon in the current research (Baggio et al., 2012). This module combines a Haar classifier from the OpenCV libraries for the detection of faces, training data transformation to Eigenfaces using Principal Component Analysis (Turk & Pentland, 1991), and a support vector machine (Cortes & Vapnik, 1995; Joachims, 1998) for the classification of new faces.

# Chapter Three

# The CongreGators Architecture

Weaver et al (2014) have created an architecture for a swarm system of heterogeneous vehicles to cooperate in search areas in an outdoor environment. The current research is implemented upon that architecture and is aimed at performing human detection and recognition through computer vision. The architecture handles the autonomous navigation of the vehicles from either a decentralized base station or on a centralized vehicle. Only the decentralized variation was used for the current research due to the need for precise agent observation and control by a base station. The architecture entails components like mission control, agent control, goal planners, and a Graphical User Interface (GUI). Although the architecture was created for autonomous vehicles to follow drive patterns and search marked areas, the actual object searching methods were limited to color thresholding for pink objects and fiducial marker tracking. The current research gives the autonomous vehicles the capability to search for humans, which is further discussed in chapter 4. All of the software was developed within the ROS environment (Quigley et al., 2009) which is a set of software libraries and tools for the development of robotic applications. ROS uses a combination of the programming languages C++ and Python, in which all of the modules for the architecture and the current research are written as well. The used functional hardware in the architecture and the architectural software are discussed below.

## 3.1 Hardware overview

Currently there are two types of vehicles functional in the architecture, namely Unmanned Ground Vehicles (UGV) and Unmanned Aerial Vehicles (UAV). Although capabilities vary per vehicle type for transportation, the other hardware components are kept as similar as possible. This provides features like modularity, software generality, and design robustness. The main processing boards in the vehicle designs were either the ODROID-U2 quad-core 1.7 GHz Exynos ARM single board computer[1] with aluminum full metal body with heat sink, or the ODROID-U3, which is a smaller equivalent board without a housing. All on-board higher level processing like path planning and role call

[1]Specifications of the ODROIDs can be found at `http://www.hardkernel.com/main/main.php`

were performed on these ODROIDs. In this research 'on-board' processing refers to computations performed on the vehicle on the ODROID, while 'off-board' processing refers to computations performed on the base station. This station used throughout all of the development and testing was a ASUS K73S Intel Core i5 laptop with an NVIDIA Geforce GT520M graphics card. The agent control and navigational execution was performed by an ArduPilot-Mega (APM) 2.6[2], which is a complete open source autopilot system embedded in each agent. The APM uses an external GPS module with an on-board compass produced by the same company as the APM for full autonomy. Each vehicle and the base station are equipped with an XBee 900 HP DigiMesh enabled RF module[3] with antenna for communication purposes. Due to the relatively small bandwidth of these modules they are only used to transmit low level data feedback and commands, i.e. for mission control, agent status, and location information. An externally powered 4 port USB Hub is added to the designs for sufficient USB access between components. The OrangeRx R620 DSM2 compatible full Range 6-channel 2.4 Ghz receiver[4] was used for manual control input and failsafe handling. Manual control was executed with a linked Spektrum DX7s 7-Channel DSMX radio system transmitter[5]. The named transmitter can handle both UAV and UGV control through several pre-programmed profiles. One of the channels is programmed as a failsafe switch between autonomous and manual control. All autonomous behavior immediately ceases if the manual control mode is activated and vice versa. A Turnigy nano-tech 5000 mAh 3 Cell Lipo battery[6] pack was used as the power supply for the UGVs. The UAVs used the 4 cell version of the same brand of battery packs. Both designs made use of a 3DRobotics power module with XT60 connectors and 6-position connector cable for correct power distribution to several vehicle components. Each vehicle carried a camera gimbal and was tested with different types of cameras, to find the optimal camera for the purpose. This optimization process is further discussed in section 3.1.3.

### 3.1.1 Unmanned ground vehicle design

The UGV design in the CongreGators architecture was based on the XTM Rail design by XTM Racing[7] as a modified radio controlled vehicle with a custom made carbon fiber housing. This housing functioned both as a roll cage for protection against mechanical damage and dirt, a feature the original XTM Rail roll cage lacked. The XTM Rail design includes features like a heavy-duty brushless Electronic Speed Controller (ESC)

[2]Specifications of the ArduPilot-Mega 2.6 can be found at
`http://store.3drobotics.com/products/apm-2-6-kit-1`

[3]Specifications of the Xbee 900 HP module can be found at `http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-pro-900hp`

[4]Specifications of the OrangeRx R620 receiver can be found at `http://orangerx.com/2013/01/22/orangerx-r620-spektrumjr-dsm2-compatible-full-range-6ch-2-4ghz-receiver-wfailsafe/`

[5]Specifications of the Spektrum DX7s transmitter module can be found at
`http://www.spektrumrc.com/products/default.aspx?prodid=spm7800`

[6]Specifications of the Turnigy battery pack module can be found at
`http://www.hobbyking.com/hobbyking/store/__11956__Turnigy_nano_tech_5000mah_3S_45_90C_Lipo_Pack.html`

[7]Specifications of the XTM Rail design can be found at `http://www.rccaraction.com/rail`

and high-torque motors, 3 motor cooling fans, a 4WD drivetrain with gear differentials, and threaded aluminum oil-filled shock absorbers with heavy duty shock shafts. The vehicles lacked active brakes but decreased their speed by friction with the ground and internal differential/motor friction. An On/Off switch was added between the battery pack and the system for easy start-up and shutdown control. Ardupilot's pre-existing Rover Firmware was loaded on the APM for correct navigation and status handling. A labeled external view of a UGV is shown in figure 3.1. In the UGV design the APM, USB Hub, and ODROID U-3 are scaffolded on top of each other (in that order) for efficient space usage. A labeled internal top view and side view of a UGV are shown in figure 3.2.

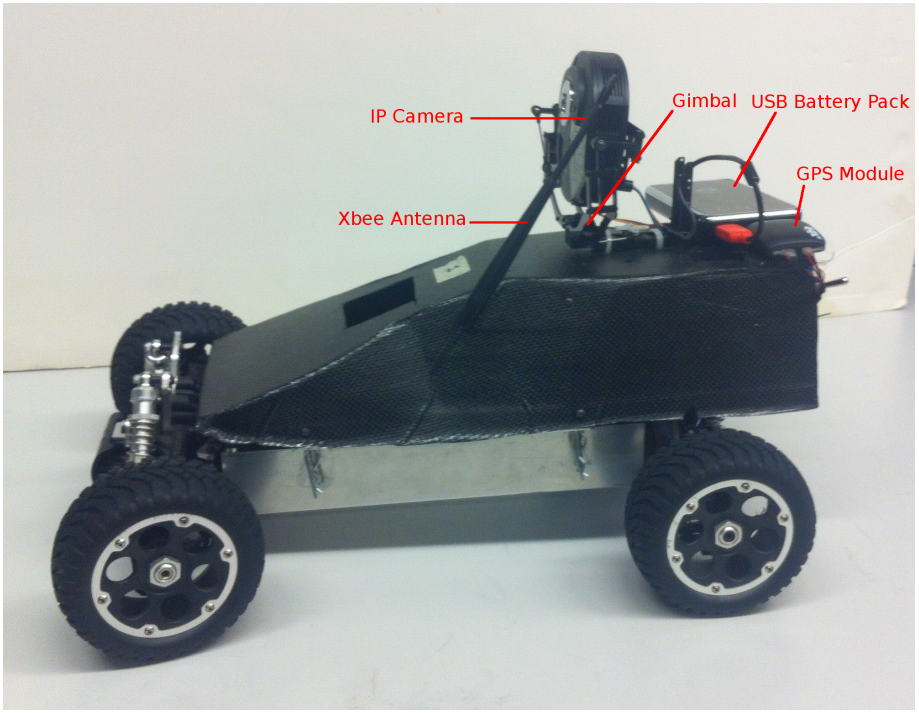
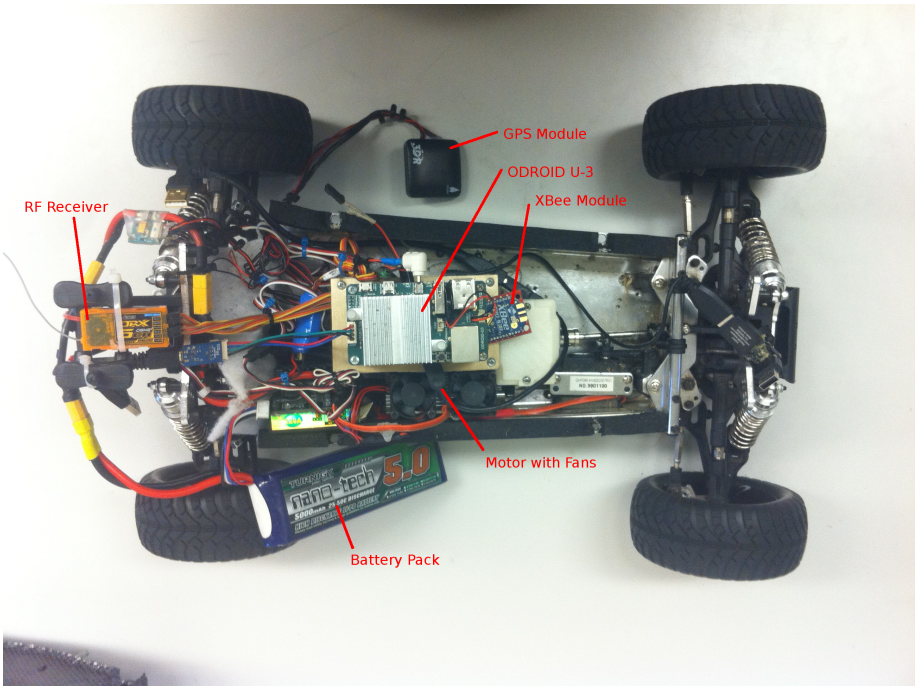



Figure 3.1: External view of a UGV

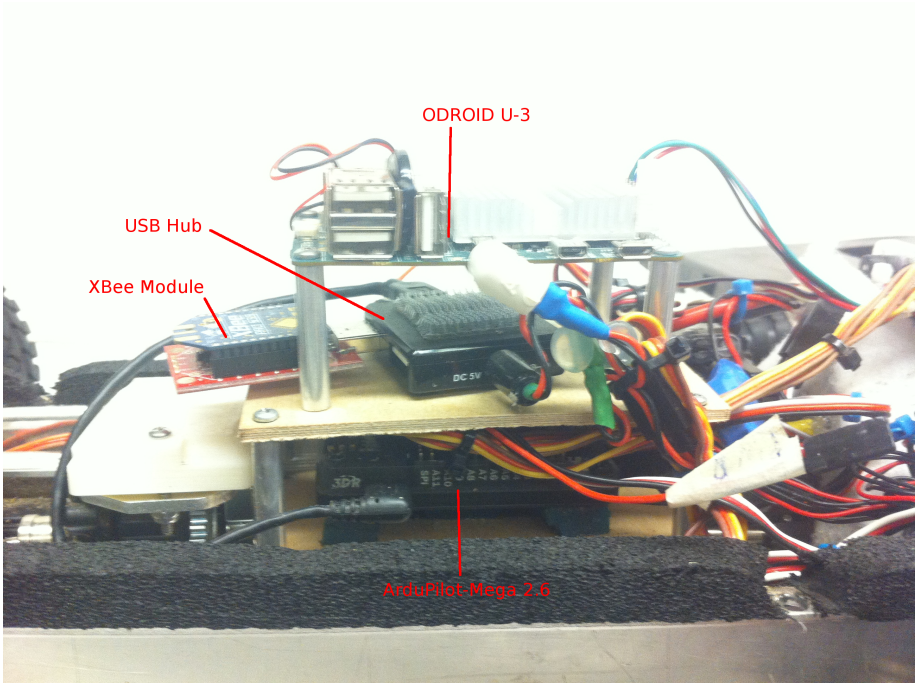
### 3.1.2 General unmanned aerial vehicle design

Within the CongreGators architecture multiple UAVs with different configurations were created among which a Flamewheel[8] quadro-copter (model F450), a Flamewheel hexa-copter (model F550), a custom made hexa-copter, and a custom made Octorotor X-8 copter (from now on referred to as the X-8). All custom made UAVs were designed and created at the Machine Intelligence Laboratory of the University of Florida. This

[8]Specifications of the Flamewheel designs can be found at
`http://www.dji.com/product/flame-wheel-arf/feature`




(a) Internal top view




(b) Internal side view

Figure 3.2: Internal view of a UGV

section will describe the general UAV setup with the X-8 as a reference example. Besides the standard components named in section 3.1, the UAVs were equipped with brushless motors, a number of ESCs according to the number of motors, and varied sizes of carbon-composite propellers, all dependent on the UAV configuration. A landing gear was used for safe landings as well as a protective structure for the vehicle's gimbal and camera. Due to the absence of a roll cage on the UAVs, the ODROID U-2 with protective housing was used on the X-8 to ensure the ODROID's safety in the case of a vehicle crash. A labeled vehicle overview of the X-8 is shown in figure 3.3.

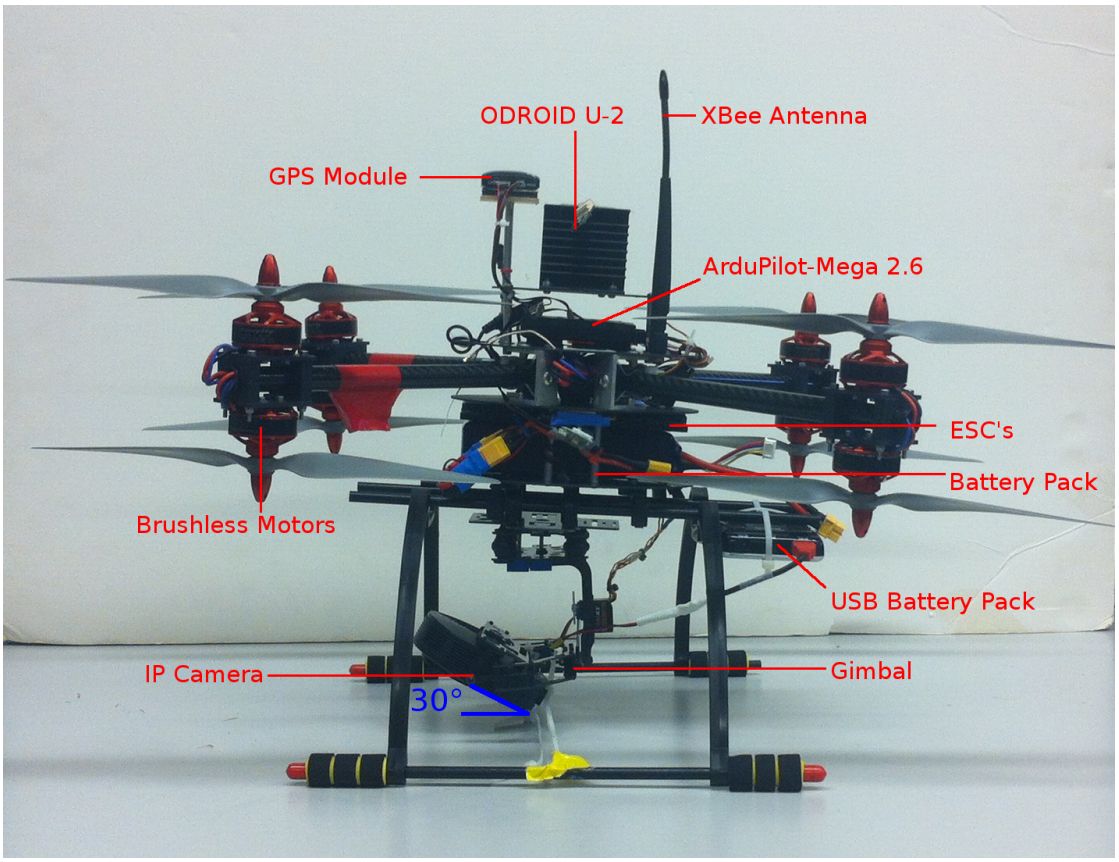



Figure 3.3: Overview of a UAV, namely the X-8

### 3.1.3 Sensor selection

For the purpose of human detection and recognition the vehicles had to be equipped with cameras on a gimbal. Multiple cameras and processing methods were tested before the final selection was made. The options for the camera selection were the Logitech

HD Pro Webcam C920[9] and the Linksys WVC80N Wi-Fi Wireless-N IP camera[10]. The latter camera works in combination with a D-Link 802.11n compliant Xtreme N Gigabit Router[11]. Some of the differences between the cameras are resolution, size and weight, and output method. The specifications of the cameras are shown in table 3.1. Note that if the battery pack is included with the Linksys camera, which the Logitech camera does not require, 260 grams are added and the total weight of the package will become 420 grams. From the specifications the Logitech webcam has the most favorable specifications in terms of resolution, field of view, dimensions, and weight.

| Model | **Logitech C920** | **Linksys WVC80N** |
|---|---|---|
| Resolution | 1280x720 | 640x480 |
| Connection type | USB | Wi-Fi |
| Size | 29x24x24 mm | 90x120x37 mm |
| Weight | 65 gr | 160 gr (420 gr) |
| Microphone | yes | yes |
| Horizontal Field of View | 78° | 61.2° |

Table 3.1: Table of camera specifications

For processing images for human detection and recognition we also have multiple options, namely on-board processing on the vehicle versus off-board processing on the base station and real-time processing versus post-processing. Since the system has to use the detections immediately to make navigational choices, it has to use real-time processing and the option of post-processing is discarded. For the decision between on-board or off-board processing camera benchmark tests were performed for 4 different camera-system configurations. These camera benchmarks consisted of running the human detection algorithm with the Histogram of Oriented Gradients classifier discussed in chapter 4 on a live camera feed for 60 seconds. The performance measure in the tests is the processed frame-rate in Frames Per Second (fps). Camera resolutions were kept at 640x480 and streaming frame rate at 10 fps on both cameras. The camera benchmark setup consisted of the Logitech webcam which was connected through USB to the Asus Intel Core i5 laptop and processing was done off-board. This camera benchmark was created to measure the maximum throughput without the wireless connectivity. Three camera-system configurations were measured against the camera benchmark:

1. The Logitech webcam is connected through USB to the on-board ODROID U-3 of the vehicle which processes the video feed.

[9]Specifications of the Logitech HD Pro Webcam C920 can be found at
`http://www.logitech.com/en-hk/product/hd-pro-webcam-c920`

[10]Specifications of the Linksys WVC80N IP camera can be found at
`http://store.linksys.com/cameras/linksys-WVC80N_stcVVproductId84737621VVcatId554678VVviewprod.htm`

[11]Specifications of the D-Link router can be found at
`http://us.dlink.com/products/connect/wireless-n-gigabit-router/`

2. The Logitech webcam is connected through USB to the on-board ODROID U-3 which sends its information through a Wi-Fi dongle to the Asus Intel Core i5 laptop, which processes the video feed. The raw camera image was sent over Wi-Fi as a ROS message on an external node and processed on the laptop.

3. The Linksys WVC80N IP camera is mounted on the vehicle and streams its video feed straight to the D-link router which is connected to the Asus Intel Core i5 laptop. For image processing the ODROID is completely omitted and video processing is done off-board.

The results from these benchmark tests are shown in table 3.2 and refer to the enumeration discussed above. Again, note that these results are frame rates after the human detection algorithm has processed the images and not raw frame rates from the cameras themselves. The results from the benchmark test in table 3.2 show that the Linksys WVC80N in this configuration has the highest throughput out of the 3 options. Although configuration 3 is Wi-Fi dependent, this camera-system setup was chosen for the execution of this research. When faster on-board processors become available, the human detection processing should be moved back to the vehicle's processor.

| Configuration | **Used Camera** | **Frame Rate (fps)** |
|---|---|---|
| Benchmark | Logitech webcam | 10 |
| 1. | Logitech webcam | 0.56 |
| 2. | Logitech webcam | 0.69 |
| 3. | Linksys WVC80N | 2.82 |

Table 3.2: Benchmark tests for different camera-system configurations

### 3.1.4 Gimbals

For stabilization and camera directionality purposes the cameras were mounted on gimbals with 2 degrees of freedom (DOF). On the UGVs the gimbals were mounted on top of the carbon fiber housing of the vehicles and could rotate the camera around the pitch and yaw axes. The gimbals on the UAVs were mounted underneath the vehicles and could rotate around the pitch and roll axes. The gimbal was connected to the APM of the vehicle which moves the gimbal depending on the vehicle's spatial orientation. User settings were applied to enable stabilization in both DOF for the UAVs and only in the pitch DOF for the UGVs. While the foremost functionality of the gimbals on the UAVs was to stabilize the camera during flight maneuvers, the UGVs mostly used the gimbals to actively change the camera's viewing area in different behavior modes (see section 4.2 for more details).

## 3.2 Architecture overview

As mentioned previously, the architecture is set up to work both centralized and decentralized which entails that the entire architecture, including the agent control and mission control, has to function on each vehicle of the swarm. The 'base controller' is in charge of the mission commands and this role is taken up by either the base station or in the absence of a base station one of the agents serves functions as such. Although the current research only uses the decentralized option, the initial architectural setup was kept intact throughout the research. The architecture consists of a number of ROS main modules, which will be elaborated on below. In addition to the main modules some sensor modules existed like a vision module, an AR tag tracking module, and an obstacle detection module (Weaver, 2014). The latter described modules will not be discussed here further. For the current research the GUI was altered and four modules were added, namely human detection, facial recognition, dynamic navigation, and a user tracker. The main modules, added modules, and GUI changes will be discussed below.

### 3.2.1 Main architecture modules

One of the main ROS modules is the swarm core which consisted of multiple functions to assist in agent control and mission and path planning. Weaver et al stated "Swarm Core is made to be customizable, allowing a diverse selection of mission types, planners, or vehicle control applications to be implemented." (Weaver, 2014). These planners make use of the standard ROS packages sbpl and sbpl_lattice_planner that implement a generic set of motion planners using search based planning (Cohen, Chitta, & Likhachev, 2010).

Roscopter is a ROS package implemented in the CongreGators architecture for the autonomous control of the unmanned vehicles. It handles the communication between an autopilot like the currently used APM and a processing board running Ubuntu using the mavlink protocol (Meier, Tanskanen, Fraundorfer, & Pollefeys, 2011). The previously mentioned apm status publisher aids in this communication by providing a frequent feedback loop from the APM to the processing board.

The role call module initiates a digital handshake between the base controller and each enrolling vehicle using agent role call service messages and role acknowledge messages. If the handshake is successful, the base controller and all vehicles present in the swarm will be updated of the enrollment.

Heartbeat is a straightforward function that provides the base station with a publisher that sends out a boolean message at a frequency of 1 hertz. This message invokes a request to the agents to acknowledge themselves. Agents will return such an acknowledgement by sending an Agent Status message at half the heartbeat rate consisting of the variables: intended receivers, agent ID, mission status, latitude, longitude, heading, battery status, and waypoint distance.

Finally, the xbee bridge handles all the actual communication among agents and base station through the XBee 900 HP DigiMesh enabled RF module discussed in section 3.1. It handles both outgoing and incoming messages. Note that the base station

only sends out a heartbeat and no basic status messages. Command messages consist of a mission related message including role acknowledgements, waypoints, mission settings, and start/stop commands, among others.

### 3.2.2 Added modules

#### 3.2.2.1 The human detection classification module

The human detection classification module is roughly based on the work by A. Leigh[12] and was created to implement the detection of human subjects in video imagery from the agents. The module makes use of OpenCV libraries and algorithms which include a HOG classifier and multiple Haar classifiers for testing. In the final experiments only the HOG classifier remained in the module based on comparing classifiers. Details on this comparison and the functionality of the classifiers is discussed further in section 4.1. When a detection occurs, a bounding box is drawn on the output window the module provides, which is published as a ROS message. The bounding box consists of 4 values, namely the x- and y-position, width, and height, all measured in pixels. All output images, with the drawn bounding box included, are saved for result analyses. If multiple agents are used at once, a ROS launchfile can be used to start multiple instances of the module. To prevent lag and buffer overflow in the image processing the algorithms are threaded in an image callback function and an image processing function. The image callback handles retrieving the frames from the camera at a maximum frequency of 10 Hz. The image processing function handles the application of the classifier to the frames, publishes the bounding box, manages the output storage, after which output can be viewed on the base station.

#### 3.2.2.2 The dynamic navigation module

The dynamic navigation module was created from scratch to combine several inputs from other modules for the dynamic navigation of agents. The dynamic navigation module receives input from the human detection module, on the status of the agents and the mission, as well as from the user tracker discussed below. From this input it calculates detected person positions, clusters person positions into prototypes, calculates dynamic driving patterns to approach said prototypes, and sends out vehicle commands accordingly. The results are all sent to the GUI for display to the user. When a vehicle has completed the approach maneuver the module requests a camera-flip action and initializes the facial recognition module, which is discussed below. In the case of a dynamic maneuver the original static path is stored and continued when the facial recognition is completed. Note that in this research a static path is defined as the search path that is set at the start of a mission, either set by the user as a waypoint path or calculated by the agent from a given search area. A dynamic path is defined as the evolving waypoints that are calculated by the dynamic navigation module as a result from human

[12]More information and code by A. Leigh can be found at `https://github.com/angusleigh`

detections. Details on the methods and functionality implemented in the module are discussed further in section 4.2.

#### 3.2.2.3 The facial recognition module

The facial recognition module is based on the work by Baggio et al (2012), which was then adapted for the performance within the current research. The facial recognition module makes use of a combination of a Haar classifier for the detection of faces, a transformation to Eigenfaces in a process that is called Principal Component Analysis (Turk & Pentland, 1991), and a support vector machine (Cortes & Vapnik, 1995; Joachims, 1998) for the classification of new faces in a video feed. Details on these methods and functionality of the module is discussed further in section 4.3. The facial recognition is only activated when the UGVs are in a holding position in the stage of the behavior where the vehicle is close to a detected test person's location and the camera is flipped up.

#### 3.2.2.4 The user tracker module

For the implementation of additional user functionality in issuing commands to the vehicles, the user tracker module was added to the architecture. The module makes use of the OpenNI libraries from ROS to enable a Kinect sensor (Zhang, 2012) to segment humans from the sensor data. The user tracker will recognize a person consisting of up to 23 segments of the body through segmentation in the point cloud, e.g. the head, torso, feet, hands, and so on. In the current research the module was used to detect arm gestures to issue mission start and pause commands for different vehicles by raising a different arm. This added functionality gives the user the ability to control the swarm with gestures, thus without physically touching any buttons. This functionality makes the user more embedded in the physical world while commanding the agents. An example of the view from the camera of the Kinect versus the 3D model of the human (which is giving a gesture) that is created in ROS though the depth information from the infrared sensor, is shown in figure 3.4. In the figure the different colors of the 3D model indicate a range of proximities to the distance sensor, red being closer to the sensor and blue being further away. Besides the described information the different body segments and their relative positions from the sensor are represented as lines from sensor position to the separate segments in figure 3.4. The 3D visualization tool Rviz[13] from ROS was used to visualize the 3D models. Note that due to the use of an infrared sensor by the Kinect sensor the performance heavily decreases if sunlight shines directly onto the sensor. This event would render the user tracker module useless, unless the sensor and the user are covered in shade, e.g. with the use of a tent.

[13]More information about Rviz from ROS can be found at `http://wiki.ros.org/rviz`

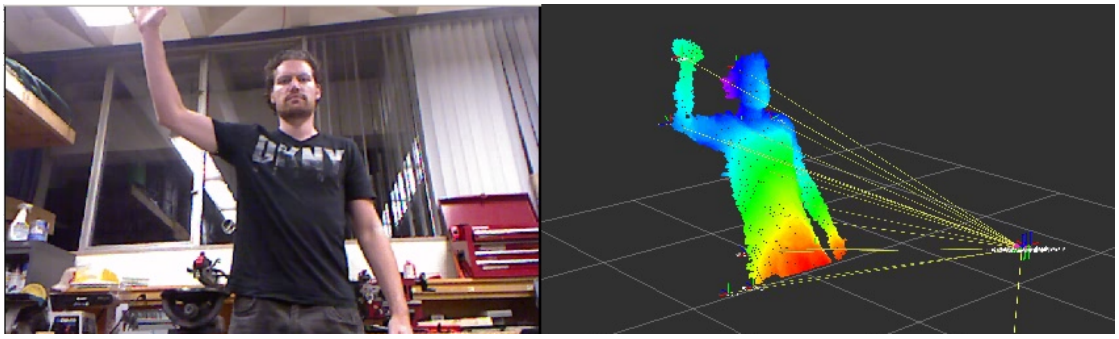


Figure 3.4: Example of the view from the camera of the Kinect versus the created 3D model of the human in ROS

### 3.2.3 Graphical user interface

The GUI for the CongreGators architecture consists of 3 main components within the 'rqt' package, which is a Qt-based framework[14] for GUI development in ROS. These components are the Agent Status, Command Plug-in, and Waypoint Plug-in. For the initial CongreGators architecture architecture those components suffice, but for this research some functionality had to be added. The initial GUI and the modified version will be discussed next.

#### 3.2.3.1 Initial GUI

The Agent Status component of the GUI keeps track of agents that are enrolled in the architecture. It creates a new tab for every agent which displays 8 fields of information. UAVs are marked by 'AA' (Agent Air) followed by the agent's label, while UGVs are marked by 'AG' (Agent Ground) followed by the label. The initial 8 fields of the Agent Status tabs were the agent's role, Altitude (m), Latitude (degrees), Longitude (degrees), the number of waypoints the agent has past, distance to the next waypoint, Battery voltage (mV), and travel state.

The Command Plug-in is used to set the mission settings, to send mission commands to the agents, and to display communication feedback from the agents. The mission settings that can be selected are Mission Name, Mission Type (which corresponds to the agent's role in the Agent Status section), Altitude Ceiling (m), Coverage Rate (%), Overlap Label (m), and Mission Timeout (s). The commands that can be sent are Send Mission, Start Mission, Pause Mission, Abort Mission, and Return To Base. Mission settings and mission commands can be set per agent individually or for all agents at once. The communication feedback function informs the user with corresponding feedback from the agents, e.g. the message "Sending Command Accepted" or "Mission Complete".

The Waypoint Plug-in handles the creation of waypoint patterns and paths from user input. This component consists of an interactive Google Maps section[15], a waypoint

[14]Information on Qt 4.8 and downloads can be found at `http://qt-project.org/doc/qt-4.8/`

[15]When the GUI is started an internet connection must be at least briefly available for Google Maps to load. A tethered smart phone with 3G internet connection would also suffice.

settings menu, and 5 buttons to aid in the waypoint creation. These 5 buttons have the labels Add, Delete, Modify, New, and Clear, which respectively function to toggle the Add mode, Delete mode, Modify mode, create a new waypoint from scratch, and clears all the waypoints. In using one of the modes, for example the Add mode, the user can click the Add button and then click on the location on the map to create a waypoint there, signified by a dark blue marker. The Delete and Modify mode work in analogous ways, where the latter is used to change default settings of the waypoint like the altitude or the position accuracy. Like with the Command Plug-in all input can be given per agent individually or for all agents at once. All waypoint markers can be dragged and dropped on the map for easy adjustments. The Plug-in also creates one green base station location marker that has the same features as the waypoint markers, but signifies the location of the base station. The waypoint settings menu provides useful functions like the saving and loading of waypoint patterns. While a waypoint pattern is created dark blue lines are drawn between the waypoints, creating a visual path in case of the 'Path' mission type and a visual search area in case of the 'Search' mission type. If a search mission is accepted by an agent it will send back a planned lawnmower pattern within the search area. A lawnmower pattern is defined as paths from side to side of the search area parallel to the edges of the area with turns of 180° at the borders, including a small shift to ensure no path section is repeated. Examples of such lawnmower patterns can be reviewed in figure 3.5 in which the (dotted) light blue lines are the agents (calculated) driving paths. Search areas are autonomously divided among all enrolled agents in which they will individually generate lawnmower paths to cover their own part of the assigned search area. The GUI represents this patch with dotted light blue lines. Agents that are enrolled and are communicating their current GPS location are shown at that location on the map with a light blue marker showing the agent's label. From the time of enrollment to the time of agent log out or program termination a light blue line marks the path that the agent has traveled.

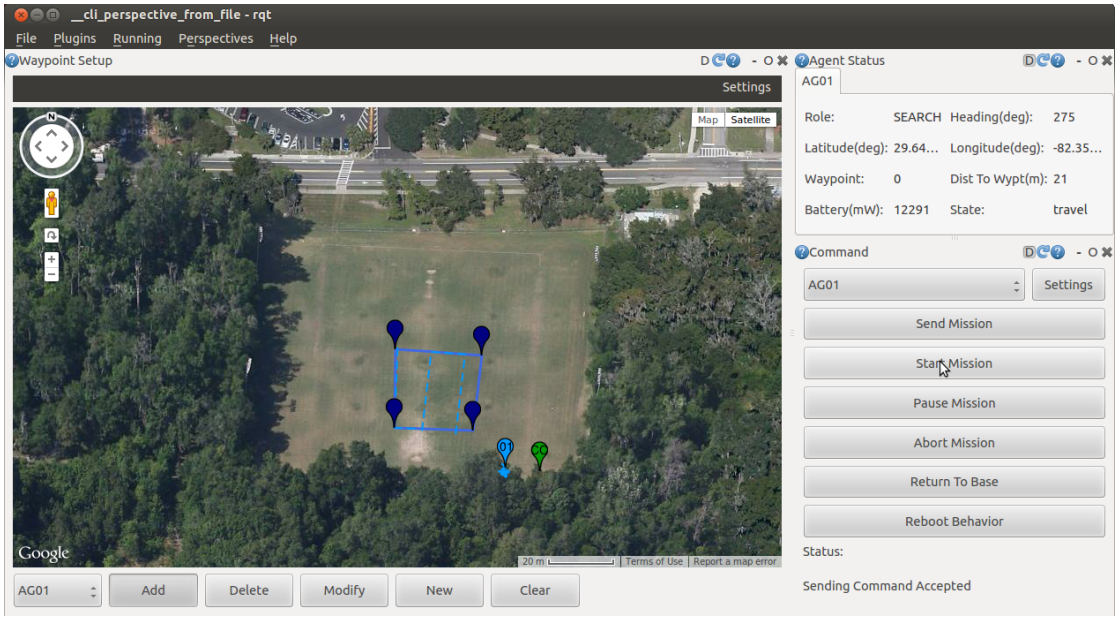
#### 3.2.3.2 Modified GUI

To represent the information provided by the current research some extra functionality is added to the GUI. The additions are detected person locations, prototype locations, dynamic waypoint locations, actual person locations, actual person input through buttons, saving and loading actual person locations, and a change in the Agent Status from altitude to heading. For information and methods on how locations are calculated and provided, see section 4.2.

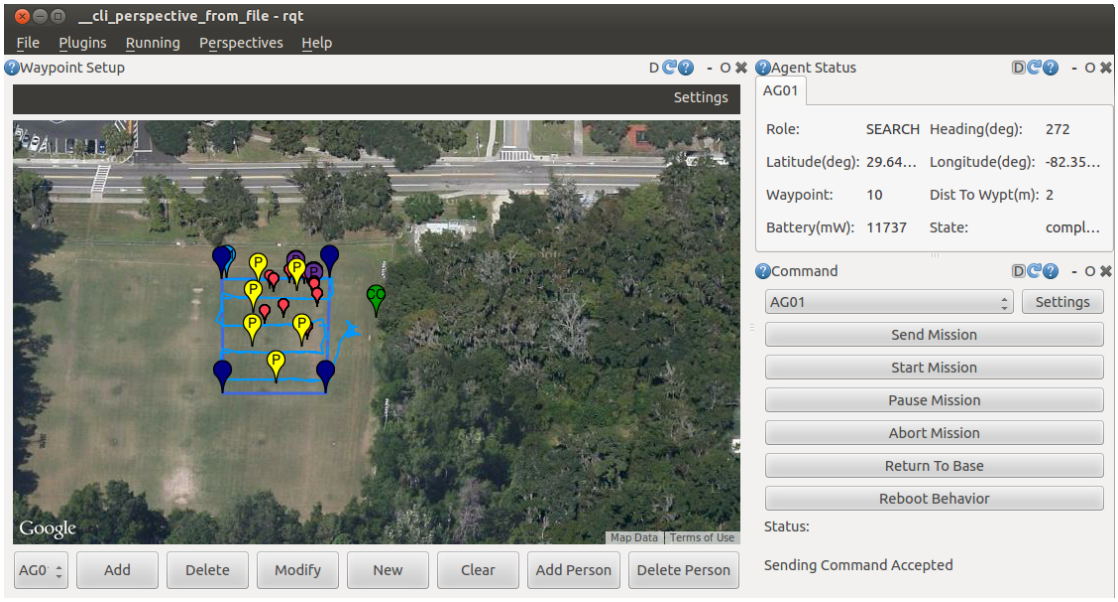
When information about a detected person location becomes available, that position is marked on Google Maps with a small red marker. In a similar fashion a purple marker signifies the location and label of a prototype and a light blue marker containing an 'N' signifies a new dynamic waypoint that is created. Actual person locations can only be entered by the user after a GPS location of a test subject is determined and are shown by a yellow marker. The user can either input that information through Google Maps using the toggle buttons Add Person and Delete Person and clicking and dragging on the map, or the user can input the GPS data through coordinates. Actual person locations can be

saved and loaded just like one can do with waypoints. All markers in the GUI were made to prompt a pop-up window with GPS location of the marker and label information (if available) when clicked.

A change was made in the Agent Status section of the GUI to display the agent's heading instead of the altitude. For obvious reasons the heading of a ground vehicle is a more important piece of information than its altitude. Examples of the initial GUI and the modified GUI are shown in figure 3.5.




(a) Initial GUI




(b) Modified GUI

Figure 3.5: Two versions of the GUI

# Chapter Four

# Human Detection and Recognition through Dynamic Navigation

## 4.1 Human detection

For the detection of humans in a video feed, multiple classifiers were considered, namely a Histogram of Oriented Gradients (HOG) classifier and five cascades for a Haar classifier. The input of the classifiers consists merely of raw video without any other information on possible human locations. Benchmarks were created to test the classifiers on their performance on 4 separate datasets in a post-processing experiment. The separate classifiers and the benchmarks on the datasets are discussed below.

### 4.1.1 Histogram of oriented gradients classifier

The pedestrian 'HOGDescriptor'[1] from the OpenCV libraries was tested on its performance on the benchmarks. The reason that this classifier was chosen is because "Locally normalized Histogram of Oriented Gradient (HOG) descriptors provide excellent performance relative to other existing feature sets" (Dalal & Triggs, 2005). The HOG pedestrian detection algorithm makes use of an overlapping grid of HOG descriptors of which the results are combined into a feature vector for a conventional Support Vector Machine (SVM) based window classifier. Before the HOG algorithm is applied to the input frames the image is pre-processed by converting it to the grayscale color space, without specific color filtering, and equalizing the histogram of the grayscale image. Sequentially the HOGDescriptor is applied on the image through a multi-scale sliding window technique. If multiple detections are made, an overlap threshold is applied to (partially) prevent the algorithm from outputting multiple detections of the same object. Multiple detections are compared through overlapping pixel areas of their corresponding

[1] The OpenCV HOGDescriptor class description can be found at
`http://docs.opencv.org/java/org/opencv/objdetect/HOGDescriptor.html`

bounding boxes, which are discussed in section 3.2.2.1. The overlap threshold is set to 50% which signifies that if two bounding boxes within one frame cover more than 50% of the same pixel area, the bounding box with the smallest total area is discarded.

### 4.1.2 Haar classifiers

For the development of Haar classifiers, Viola and Jones developed Haar-like features (Viola & Jones, 2001) which are adapted on the idea of Haar wavelets. In a Haar-like feature the pixel intensities from adjacent rectangular subsections of an image are summed up and differences between those sums are calculated. These differences are then matched and categorized against a classifier cascade defining the Haar-like features. The training of a Haar classifier is the creation of such a cascade through multiple stages in which false-positive rates and detection rates are optimized.

Five Haar cascades for human detection from the pedestrian view were used in the benchmark tests, from which four were taken from the OpenCV libraries and one was created specially for the current research. The four OpenCV human detection cascades are for full body, lower body, upper body, and an adapted upper body cascade named 'Haar mcs upperbody' (Castrillón-Santana, Déniz-Suárez, Antón-Canalís, & Lorenzo-Navarro, 2008). Since there was no cascade on-hand for the detection of humans from a top view, one was created from scratch and named 'Haar top-view'.

#### 4.1.2.1 Creating the Haar top-view classifier

For the creation of the Haar top-view classifier a cascade had to be created. The training phase used input images, recorded using both UAV imagery and static video imagery from an experimental setup. All UAV imagery was recorded through manual flight on dates before June 18, 2014 and in accordance to the regulations applied before the publication of the "Interpretation of the Special Rule for Model Aircraft"[2] by the Federal Aviation Administration. The UAV recorded multiple individuals from different altitudes passing underneath the vehicle. In the experimental setup individuals were recorded from a static altitude passing underneath the camera from the same angle at which the camera on the UAVs was mounted. Through the recordings of different individuals a spread in appearance was ensured to create a diverse training set. The camera view angle was kept at 30° raised from a downwards perpendicular view, both on the vehicle, which is shown in figure 3.3 of the X-8 by the blue markings, as in the experimental setup. From these experiments 1000 positive images and 2000 negative images were taken for training and 100 positive and 100 negative images for testing (the later discussed UAV field data set was created from these images). For these purposes the same camera was used.

The 1000 positive images in the training set were cropped such that only the subjects were seen in the result. These 1000 cropped images were then processed into 7000 positive examples with the use of OpenCV's 'opencv_createsamples' function which applies

[2]The "Interpretation of the Special Rule for Model Aircraft" was published on June 18, 2014 and can be found at `http://www.faa.gov/uas/publications/media/model_aircraft_spec_rule.pdf`

perspective transformations on the original images. To be more precise, the method creates a large set of positive examples from the given object input images by randomly rotating, changing the image intensity as well as placing the images on arbitrary backgrounds taken from the negative image set. The cascade was created through training on these 7000 positive examples and 2000 negative images in a 30 stage training process using the OpenCV 'opencv_traincascade' algorithm. The parameter for minimal desired hit rate for each stage of the classifier was set to 0.99 and the parameter for maximal desired false alarm rate for each stage of the classifier was set to the default 0.5. Using an Apple Mac Pro "Eight-Core" 2.8 GHz Xeon desktop computer for the training procedure the overall training time was 6 days. After finishing the top view Haar cascade could now be used for field and benchmark testing.

### 4.1.3 Benchmark tests and datasets

The six human detection classifiers, namely HOG, Haar fullbody, Haar upperbody, Haar mcs upperbody, Haar lowerbody, and Haar top-view, were tested with benchmarks on four datasets. A benchmark consists of running the detection classifiers on 100 positive images, i.e. images with a person fully shown in the image, and 100 negative images, i.e. images from the same environment as the positive images but without a person present. The four data sets were taken from (1) video footage from a UGV in the field where final experiments were performed, (2) UAV top view video footage in the same scenario, (3) video footage from a UGV in a busy campus area, and (4) from the INRIA unoccluded person dataset[3] (Dalal & Triggs, 2005). These datasets were named UGV field, UAV field, UGV campus, and INRIA, respectively. The INRIA dataset is a well-established collection of random photos of humans, therefore the negative images are also a collection of random scenes without humans in them. The reason for the choice of the first two datasets was to test the classifiers on data from the same environments the final experiments would be conducted in. Dataset 3 and 4 were included to test classifiers on their general performance. In the different datasets the angles from which the human subjects are viewed differ significantly. The view from the UGVs will be defined here as Pedestrian view and the UAV's view will be defined as Top view. From this definition we can now determine that dataset 1, 3 and 4 are Pedestrian view datasets, while dataset 2 is a Top view dataset.

Table 4.1 shows properties of the datasets, namely their number of unoccluded human subjects in the positive images, resolution of the camera used for recording, and the type of view. All datasets consist of 100 positive images, but due to occurrences of multiple unoccluded subjects in single positive images in the UAV field set and INRIA set, their number of test subjects exceeds 100.

The benchmarks were scored on 3 measures, namely correct detections, false-positives, and processing rate in frames per second (fps) of the 200 frames per dataset. False-positives are the detection of a human subject when there is no subject in the output bounding box, which was checked by the user. In the checking process the following rules

[3]The INRIA dataset can be downloaded from `http://pascal.inrialpes.fr/data/human/`

| Dataset | **number of subjects** | **Resolution** | **View type** |
|---|---|---|---|
| UGV field | 100 | 640x480 | Pedestrian view |
| UAV field | 128 | 1280x720 | Top view |
| UGV campus | 100 | 1280x720 | Pedestrian view |
| INRIA | 158 | Diverse | Pedestrian view |

Table 4.1: Dataset specifications

applied. For the HOG, Haar fullbody, and Haar top-view classifiers, if a bounding box covered more than 50% of the subject's body the detection was considered correct. The Haar upperbody, Haar mcs upperbody, and Haar lowerbody classifiers were composed of training on several parts of the human subject (Kruppa, 2004). Therefore, if parts of the subject associated with the classifier were detected (with a relatively correct size) the detection was considered correct, e.g. the top of a head for the Haar upperbody classifier or a leg for the Haar lowerbody classifier. For all classifiers, if a human shadow 'attached' to a subject was detected, the detection was also considered correct. Besides, if multiple detections were made of a subject only 1 correct detection was counted and the other detections dismissed. Note that a dismissal is subtracted from the 'correct' benchmark measure while not being added to the false-positive count. The same action applied if illustrations or other representations like statues were detected in the background. Only the INRIA set included some of those objects in the background. All other detections from classifiers that did not apply to the discussed exceptions, were marked as false-positives.

## 4.2 Dynamic navigation

A dynamic navigation module was created to combine all the input from other modules and agents into a new investigation behavior. When human detections are made through the HOG classifier, that information is combined with the current Agent Status information from the agent that made the detection. This results in a detected person location consisting of a GPS coordinate. After 3 or more detections are made, every occurrence of a human detection initiates a clustering algorithm to see which detections should be labeled as a cluster of detections, thus belonging to one test subject. A prototype is created in the middle of the cluster as representation of the found person and a new drive pattern is created to investigate the person, i.e. try to recognize the person's face. If the agent is in an investigating position, the camera could be 'flipped up' with the gimbal under an angle of 30° to get the subject's face in view. This angle is the maximum gimbal pitch angle and is in line with the investigation distance and the average human height. Finally a sequence of commands is issued by the module to complete the behavior. The previously described process will be discussed in detail below.

### 4.2.1 Detection processing

Each detection is processed into a detected person location with a label and possibly accompanied by a prototype. All the produced information is displayed on the GUI. Clustering will re-occur after every human detection when 3 or more detections are made, but will only generate a prototype if more than 3 detections are made within a range of 3 meters of each other. The clustering algorithm is incorporated to exclude outlying false-positives and to build a certain 'confidence' about detected person existence and location.

#### 4.2.1.1 Person localization

At the moment an agent detects a human through the HOG classifier, the person localization is started. First the values of the detection bounding box are used to calculate distance and angle to the subject. The angle to the subject relative to the vehicle is calculated through equation 4.1, where $frameWidth$ is 640 and $FoV$ (Field of View) is 61.2°, when the IP camera is used.

$$angle = \left( \frac{\Big(Boundingbox.x + (Boundingbox.width/2)\Big)}{frameWidth} - 0.5 \right) * FoV \tag{4.1}$$

Note that angles of detections range from -30.6° to 30.6° minus half of the width of the detection. Distance from the agent to the subject is calculated according to the height of the bounding box. A distance calibration was performed to determine the relation between the height of a bounding box and the distance to the subject. A subject of 180 cm was placed in front of the vehicle in a range from 2 to 15 meters with 1 meter intervals. Detections were made by the HOG classifier at distances from 3 to 14 meters. Averaged bounding box heights were plotted against distances and regression analysis was performed up to the fourth degree polynomial. Results from the regression analysis are shown in figure 4.1. The analysis shows that $3^{rd}$ degree polynomial regression (also know as cubic regression) and $4^{th}$ degree polynomial regression are very similar and describe the data better than $2^{nd}$ degree polynomial regression (quadratic regression). Following Occam's Razor the function is chosen that describes the data well while keeping the complexity of the function as low as possible, which in this case is the cubic regression function. The general cubic function of distance versus bounding box height is shown in equation 4.2, with the coefficients $p_1 = -1.0123 * 10^{-6}$, $p_2 = 0.00094457$, $p_3 = -0.30537$, $p_4 = 38.996$.

$$\begin{aligned} distance = p_1 * Boundingbox.height^3 + p_2 * Boundingbox.height^2 + \\ p_3 * Boundingbox.height + p_4 \end{aligned} \tag{4.2}$$

When the distance and angle to the subject are known, the current latitude, longitude, and heading of the agent in question are taken from the current Agent Status. These 5 values can be combined to determine the latitude and longitude of the subject with the pair of equations 4.3 and 4.4, where $lat_{sub}$ and $lon_{sub}$ signify the subject's

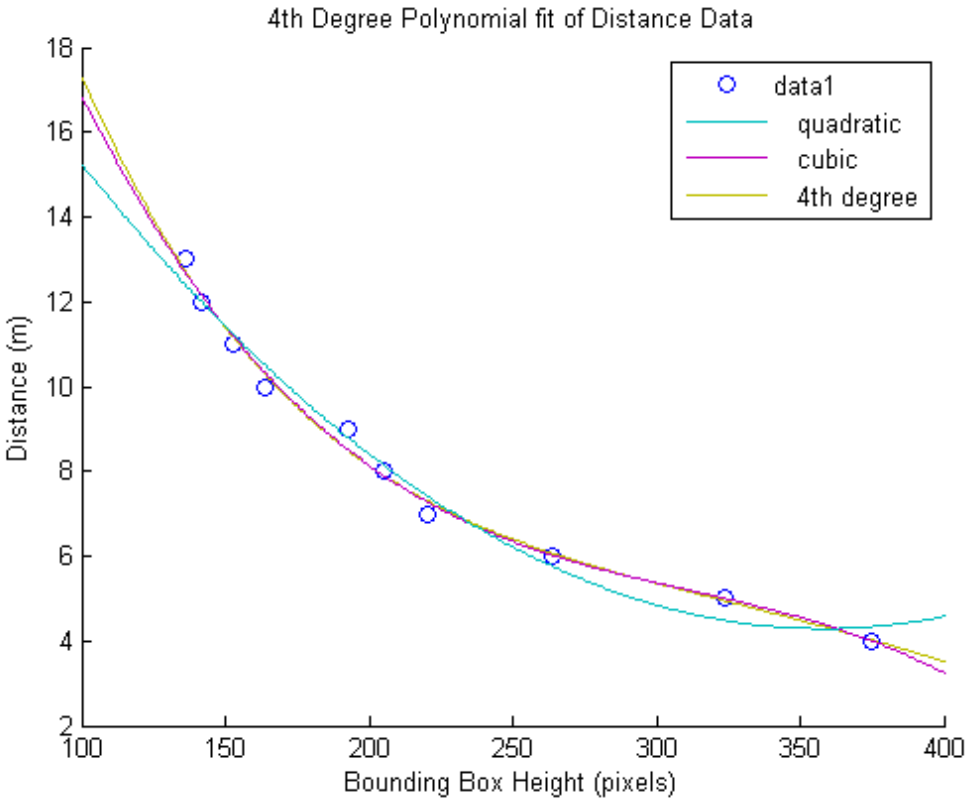



Figure 4.1: Regression analysis on distance calibration data

latitude and longitude respectively, $lat_{veh}$ and $lon_{veh}$ signify the agent's latitude and longitude respectively, *dis* being the distance between agent and subject, *Earth* being the Earth's radius (km), *heading* being the agent's bearing (clockwise from magnetic north), and *angle* being the relative angle from agent heading to subject.

$$lat_{sub} = asin\Bigg( sin(lat_{veh}) * cos\Big(\frac{dis}{Earth}\Big) + cos(lat_{veh}) * sin\Big(\frac{dis}{Earth}\Big) * cos(heading + angle)\Bigg) \tag{4.3}$$

$$lon_{sub} = lon_{veh} + atan2\Bigg( sin(heading + angle) * sin\Big(\frac{dis}{Earth}\Big) * cos(lat_{veh}), cos\Big(\frac{dis}{Earth}\Big) - sin(lat_{veh}) * sin(lat_{sub})\Bigg) \tag{4.4}$$

Note that all angles should be expressed in radians. End results are calculated as radians and can be recalculated to degrees by multiplying them with the factor $(180/\pi)$. The

detected person location can now be published on ROS through a customized message for the GUI to be shown.

#### 4.2.1.2 Clustering and prototype creation

The clustering algorithm is initiated after 3 human detections occur and clusters are re-calculated after every new detection occurrence. In the clustering algorithm two parameters are most important, namely *detectionsInCluster* representing the minimum number of detections to form a cluster and *clusterRadius* which represents the radius a cluster area has. Default values of the parameters are $detectionsInCluster = 3$ and $clusterRadius = 3$ (meter). Detections are marked by numerical labels which represent cluster identifiers. If a new detection is added it initially receives the label 0, which represents that the detection has not been assigned to a cluster yet. The first time the clusters are generated, which occurs when exactly 3 detections are made, a detection is chosen at random and assigned the label 1. Note that at the start of this process all detections are labeled as 0. Subsequently another detection is taken and the euclidean distance between the two detections is calculated. This process first converts the detection location data from Geographic (latitude and longitude) coordinates to Universal Transverse Mercator (UTM) coordinates (Northing and Easting). Conversions between these coordinate types was performed in the software with the LLtoUTM and UTMtoLL functions of the gps_common package[4] from ROS. UTM coordinates include a Zone value which is assumed to remain the same in this research, since no large distances are traveled by the agents and no zone border is close to the experiment locations[5]. Differences between Northing and Easting of the two locations are calculated and provide input to the Pythagoras Theorem to calculate distance. Equation 4.5 shows the function that produces the euclidean distance between two UTM coordinates, while equation 4.6 shows the function that produces the angle between two locations. The latter function (needed later in the dynamic navigation behavior) is expressed in radians but can be converted to degrees by multiplying the result with $180/\pi$. Both equations include the components $N_i$ and $E_i$ being the Northing and Easting with location identifier i, and $i_1$ and $i_2$ being the specific identities of the detected locations.

$$distance_{eucl} = \sqrt{(N_{i_1} - N_{i_2})^2 + (E_{i_1} - E_{i_1})^2} \tag{4.5}$$

$$angle(rad) = atan2(E_{i_1} - E_{i_2}, N_{i_1} - N_{i_2}) \tag{4.6}$$

If the calculated distance is smaller than *clusterRadius* the new detection is assigned the same label as the labeled detection, namely label 1 in this first iteration. If the distance exceeds *clusterRadius* a new label will be assigned which is an increment of the highest existing label number. This process is repeated for the third detection. If the number of person detections with the same label exceeds *detectionsInCluster*, a prototype is generated to represent the cluster. The location of this prototype is

[4]Package information can be found at
`http://docs.ros.org/hydro/api/gps_common/html/namespacegps__common.html`.

[5]All experiments were performed in Gainesville, Florida, U.S.A.

calculated by averaging the UTM location data of the cluster's members, shown in equations 4.7 where the equation variables denote the same variables as in equation 4.5.

$$\begin{aligned} N_{proto} &= \frac{\sum_{i=1}^{n} N_i}{n} \\ E_{proto} &= \frac{\sum_{i=1}^{n} E_i}{n} \end{aligned} \tag{4.7}$$

When the previously described process of labeling has already taken place at least once, only new detections will be included to the prototype if they are within the cluster radius. If that is not the case, but they are within the range of other detections, they are added to the cluster of those detections. This is an opposite approach to re-checking and renaming every location from scratch. The reason for this choice is to keep a certain consistency in cluster naming. After every addition of a person detection and label, all prototypes are recalculated to update their position. After that the creation of prototypes dynamic drive patterns can be initialized.

### 4.2.2 Dynamic drive patterns

Since prototypes are representations of detected person locations, these locations of interest are investigated by the agents. Three options of drive patterns have been implemented to approach prototypes, namely a single waypoint, a star pattern, and a diamond pattern. In every pattern the closest waypoint(s) to the subject are 'investigation' points where the camera is flipped up 30° in the pitch direction by the gimbal to get the subject's face in view and start facial recognition, which is discussed in section 4.3. At this waypoint the 'holding' state is prolonged to give the facial recognition module enough time to perform correctly. All dynamic driving patterns are discussed below and shown in figure 4.2, where the dotted lines were added for clear representation of the path and red crosses mark the investigation points. When approaching a location of interest to investigate a person's face, two factors are of great importance, namely maintaining a certain distance from the subject and making sure the subject is in view of the camera. These factors were regulated by a behavior parameter *approachDistToPerson* that indicates the approach distance of the agent to the subject and the varying drive patterns. The default setting of *approachDistToPerson* was 3 meters in all dynamic navigation experiments. Note that the efficiency of every dynamic driving pattern is highly dependent on the accuracy of the human detection and person localization algorithms. After a prototype was created the agents would immediately respond by approaching the created prototype with a dynamic driving pattern.

#### 4.2.2.1 Single waypoint

The most simple way to approach a location of interest is to move along a straight line to that location up to a distance of the given *approachDistToPerson* parameter. To compute the coordinates of this single new waypoint one can combine the previously discussed steps. These include calculating the distance between the locations with equation 4.5, subtracting *approachDistToPerson* from that result, calculating the angle to

the location with equation 4.6, and inputting those variables with agent location information into the pair of equations 4.3 and 4.4. This process computes the stopping location of the vehicle along the shortest path. For a visual representation of the single waypoint approach see figure 4.2a.

#### 4.2.2.2 Star pattern

The driving pattern in the form of a star was created to approach a subject from 4 angles while trying to keep the heading of the vehicle towards the subject. The angle of approaches towards the subject are 0°, 90°, 180°, and 270° respectively. The star pattern consists of 7 waypoints, of which 4 are used for actual recognition and 3 are used for correct agent heading purposes. The first waypoint is a waypoint created with the same method for a single waypoint, i.e. the point in between agent and subject on the given approach distance. Subsequently three pairs of waypoints are reached in a counter-clockwise fashion. In a waypoint pair, first the agent drives to a waypoint on twice the $approachDistToPerson$ distance under the correct angle to the subject. Then the agent drives inwards towards the subject to stop at the given approach distance. This would theoretically result in having the detected subject in view of the camera since the agent's heading lines up with its location. For a visual representation of the star pattern see figure 4.2b.

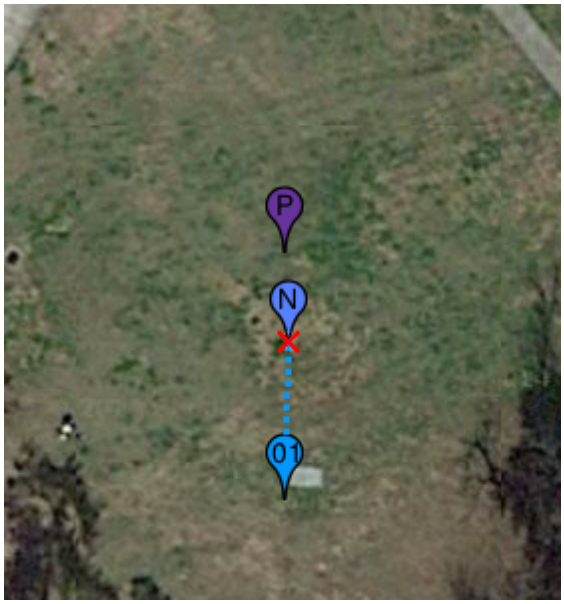
#### 4.2.2.3 Diamond pattern

Another option to get the subject's face into camera views is to mount the camera on the vehicle in a different fashion and drive around the subject. The gimbal on the vehicle is then rotated 90° counter-clockwise in the yaw direction and fixed manually in that position before the mission. The camera could still be flipped up in a pitch rotation for the facial recognition and would stay in this configuration during the entire pattern. Driving around the subject in a diamond shape while holding still in the middle of the paths would theoretically result in having the face of the subject in view at 4 different angles. For a visual representation of the diamond pattern see figure 4.2c.

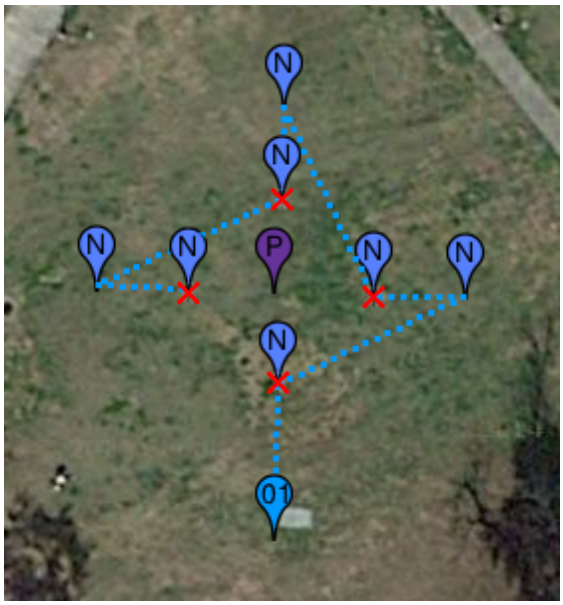
Mounting the camera sideways like described above proposes multiple swarm behaviors because of the heterogeneity it introduces among the UGVs. To give but a few examples, if one agent would have its camera mounted in a sideways fashion and other agents would have their cameras mounted 'normally', the regular agents could search for human subjects but let the investigation tasks be done by the 'special' agents. Another option would be to adjust the software so human detection and localizing could also be done with the sideways oriented camera, thus rendering the special agents with the entire set of capabilities that the normal agents possess.

#### 4.2.2.4 Dynamic drive pattern choice for dynamic navigation experiments

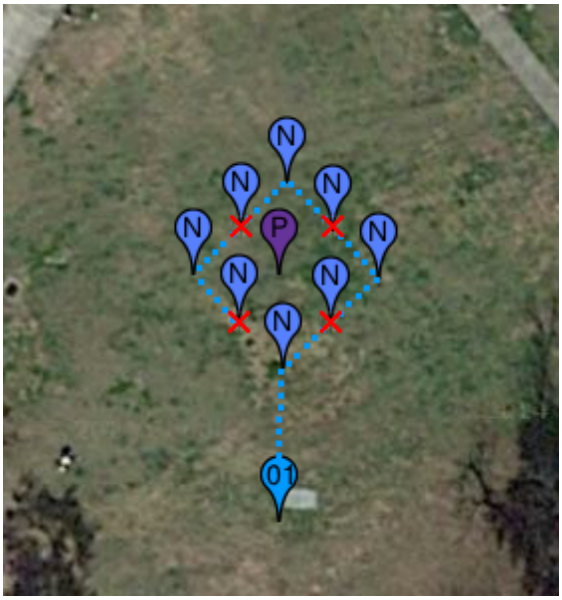
After experimenting and testing all drive patterns only the single waypoint pattern was chosen to be used in the final experiments. Due to its simplicity and small number of




(a) Single waypoint




(b) Star pattern




(c) Diamond pattern

Figure 4.2: Three dynamic driving patterns to approach a prototype

navigational actions this driving pattern showed the highest chance of having the subject's face in the camera's view after completion. Note that this method of approach only investigates the person's face from one angle and therefore the condition that the subject was facing the vehicle at the time of facial recognition was imposed. The reason for the ineffectiveness of the star pattern and the diamond pattern were the inaccuracy

of the agent's GPS module and the allowance by the system of the waypoint approach proximity. This system allowance was a result of the initial architecture setup to search large areas in broad lawnmower patterns in combination with the GPS inaccuracy. In the diamond pattern the 'roll-out' characteristic from the agents also had a bad influence on the agents performance in getting a person's face in camera view during the driving patterns. Because the facial recognition module has the need for a precise vehicle approach and correct vehicle heading in order to get the subject's face in view, the methods with the highest simplicity yielded the best results. Even though there is a requirement to have faces in the direction of the approaching vehicle's camera, the single waypoint driving pattern was used in the final experiments as being the most effective.

### 4.2.3 Measuring behavior performance

The person localization performance was measured in 5 experiments of two separate experiment types. Due to the "Interpretation of the Special Rule for Model Aircraft"[6] implemented by the Federal Aviation Administration on June 18, 2014 the current research could not make use of any UAVs in the experiments. Despite of the fact that the UAVs could not actually be deployed in the experiments, the current research does provide a proof of concept for the functionality of such vehicles providing human detection within the architecture. This proof of concept is further discussed in chapter 6. The first type of experiment was conducted twice with 6 test subjects and 2 UGVs. The second type of experiment was conducted three times with 1 test subject and 1 UGV. In all experiments actual subject locations were randomized, while keeping them apart at least 5 meters, and their locations were recorded by GPS for detection verification purposes.

In experiment type 1 a search area was created around the 6 test subjects, which was kept the same for both experiment instances. The deployed agents divided the search area and created lawnmower paths from side to side. Note that the lawnmower patterns do not guarantee that subjects will appear in the camera's field of view since the subject's positions are randomized and the agents have a certain coverage rate (the amount of spread between the lawnmower strokes). This coverage rate is limited by the quality of the GPS signal and its drift.

In experiment type 2 a manual path was created from a single UGV to a location straight behind the test subject to search along that line for the subject. Agent start and end position were kept the same. This experiment type was conducted to measure person localization performance with a high certainty of the subject being in the field of view of the camera. The specifications of the two experiment types are summarized in table 4.2. The $confidenceThreshold$ parameter was set according to environmental conditions and will be discussed in section 4.3.3.

The experiments were scored on 8 measures, namely (1) the total number of detections made by the agents, (2) the number of 'correct' detections through sufficient proximity to test subjects, which is determined by the parameter $correctDistance$ which

[6]The "Interpretation of the Special Rule for Model Aircraft" can be found at `http://www.faa.gov/uas/publications/media/model_aircraft_spec_rule.pdf`

| | **Experiment type 1** | **Experiment type 2** |
|---|---|---|
| Agents | 2 | 1 |
| Test subjects | 6 | 1 |
| Subjects in datasets | 3 | 4 |
| Search method | Lawnmower paths | Direct path |
| Subject view guarantee | No | Yes |
| $confidenceThreshold$ | 0.72 | 0.70 |

Table 4.2: Specifications of the differences between the two experiment types

will be discussed below, (3) the number of false-positive (incorrect) detections, (4) the total number of created prototypes, (5) the number of 'correct' prototypes through sufficient proximity to test subjects, also compared with the parameter *correctDistance*, (6) the number of false-positive prototypes, (7) the number of test subjects that were actually 'found' by the system, i.e. that had at least one prototype at a sufficient proximity, and (8) if a 'useful' dynamic drive pattern was created with respect to facial recognition possibilities, which is discussed below. The *correctDistance* parameter functions as a threshold to determine the correctness of a detection or prototype, i.e. to determine if the detection is in a sufficient vicinity of an actual subject position. The value of *correctDistance* was set to 5 meters for all experiments as a default. Measures 2, 3, 5, 6, and 7 were also calculated as a percentile of their respective categories. Note that measure 4 is mostly important for experiment type 1, since in experiment type 2 this measure yield boolean-like results (either 1 or 0) due to the experimental setup. The usefulness of any created dynamic patterns in measure 8 is a boolean result, which is determined by the human operator in terms of whether a test subject was in the camera's view after the completion of the dynamic pattern.

### 4.2.4 Command handling

The initial CongreGators architecture included 4 mission commands that could be sent to the agents, namely a *Start*, *Pause*, *Abort*, and *Returntobase* command. These commands were supplemented by mission information messages like for example waypoint sequences. For the purpose of obtaining the correct behaviors from the agents within the current research there were two options. Either the behaviors could be built by scaffolding the existing commands into a new behavior or new commands could be created like an *Approach* and *Investigate* command combined with an appropriate internal state of the agents. Due to resource and time constraints only the first option was tested and proved to produce a desired behavior.

The command sequences from the scaffolding method will be explained more thoroughly here. At the moment a prototype is created by the human detection algorithm a *Pause* command is sent to the agent that has made the detection. After the *Pause* command is sent and an acknowledgement from the corresponding agent is received, the agent's static path and the upcoming waypoint number are stored. Subsequently one of the dynamic driving patterns, which is specified by the user before the mission, is

applied to the prototype and current agent location. At this point the new dynamic mission is displayed on the GUI. Subsequently an *Abort* command is sent to abort the static mission within the agent, the dynamic mission is sent to the agent through a waypoint sequence message, after which a concluding *Start* command is sent to start the dynamic mission. While executing the dynamic mission, internal agent states are cues for the behavior to perform certain actions. A prolonged holding time on a waypoint is a cue to flip up the camera and start facial recognition. A *Completed* message from the agent is a cue that the dynamic mission is successfully completed and that the original static mission can be continued. This continuation can simply be performed by sending the stored static mission and the upcoming waypoint number back to the agent followed by a *Start* command. No *Pause* and *Abort* commands are necessary in this case.

Missions can be started or intervened through either the GUI or through human gesture control using the Kinect, as is discussed in sections 3.2.3 and 3.2.2.4. While all commands, mission settings, and waypoint sequences can be sent through the GUI, only *Start* and *Pause* commands can be issued through gesture control with the Kinect.

### 4.2.5 Conditions for dynamic navigation experiments

Some conditions were organized in the dynamic navigation experiments to ensure that the vehicles did not physically crash. Since there was no obstacle avoidance implemented on the vehicles the experimental conditions were imposed so that no obstacles were present in the experimental area, other that the test subjects. The subjects were asked to step aside if the vehicle was on a collision course with the subject. Since the research was set up to detect and recognize people but not track them, the condition was imposed that these subjects stood straight up at a stationary position at spread out locations. This would avoid frequent occlusion between subjects for view of the vehicle's camera. The conditions that the test subjects were of an average adult height (between 1.60 and 2.00 meter) was related to the former discussed conditions. Finally, due to the experimental setup of the single waypoint approach discussed in section 4.2.2.4, the condition was imposed that subjects faced the agents during facial recognition. In all experiments no other conditions were imposed on the subjects or the experiment environment than the ones discussed above. No clothing or appearance regulations were imposed on the subjects to test behavior performance on a diverse group of individuals.

## 4.3 Facial recognition

The facial recognition module in the current research is based on the work by Baggio et al (2012) and consists of three main phases, namely (1) a facial data collecting phase, (2) a classifier training phase, and (3) a facial recognition phase. The module makes uses of the combination of a facial detection Haar classifier (Viola & Jones, 2001), data conversion to Eigenfaces (Turk & Pentland, 1991), and an SVM (Cortes & Vapnik, 1995; Joachims, 1998) to classify new faces. Certain conditions (discussed below) were applied to the facial recognition phase for the sake of performance in the behaviors by increasing recognition confidence over time. When the facial recognition module is

started, a window per agent appears on the base station showing the live camera feed and any detections as rectangular bounding boxes for the user to review.

### 4.3.1 Collecting facial data

Collecting the facial data from a subset of the subjects used in the experiments was always done before the missions were started. The facial data collecting phase produces 50 images per subject out of which the face is cropped automatically. The camera with the highest resolution available was used for the data collection, which is the Logitech HD Pro Webcam C920 as can be deducted from table 3.1. Training images with the highest possible resolution yield the best results, even when a lower resolution camera is used for the recognition. In the data collection progress a frame from the camera is only stored if a face is detected in the image. This detection is performed by the Haar face detector from the OpenCV libraries called "haarcascade_frontalface_alt.xml". If a face is detected in the frame this is shown on the output window and the frame is cropped using the dimensions of the bounding box, which results in data of only facial crops from the subject. The image is also converted to grayscale without color filtering for pre-processing purposes for the classifier training module. If multiple faces are detected in the image only the largest faces is stored, assuming that the subject for training is closest to the camera. This functionality also prevents the storage of false-positives of for example the background, up to a certain level. Even though only the largest face is cropped, in data collecting for the experiments precautions were taken that only one subject was in the camera's field of view at a time. Subjects are asked to slightly and slowly move their head left, right, up, and down to ensure their face is seen from multiple angles. The data collection software is running until it has stored 50 crops of faces per subject. The data is labeled with the subjects name for recognition verification purposes. After this phase is completed the data can be reviewed and pruned by hand if necessary, i.e. when false-positives are stored erroneously.

### 4.3.2 Facial recognition classifier training

To train the system on the data collection, the facial dataset is first converted to eigenfaces and their eigenvalues with Principal Component Analysis (PCA), a technique first introduced in 1991 by Turk and Pentland (1991). The reason this conversion is not already performed in the data collecting phase is because the data would not be reviewed properly anymore by the user due to the visual distortions in PCA reconstructed faces. To briefly elaborate on PCA, the process first creates a so called average face from all the data. Subsequently the data is converted into a set of eigenfaces that represent the main differences between the average of the training images and how to represent each training image using a combination of those differences. The first eigenface represents the most dominant facial feature differences, the second eigenface represents the second most dominant facial feature differences, and so on. For a visual example of the differences between the average grayscale face and the conversion to the first 5 eigenfaces, see figure 4.3 which was taken from Heseltine et al. as an illustrated example (Heseltine,

Pears, & Austin, 2002). The eigenfaces and eigenvalues are now the data space in which

Figure 4.3: Average face image and the first 5 eigenfaces of a particular face. Figure taken from Heseltine et al. (2002)

new examples can be projected for classification by the facial recognizer. To evaluate new facial images, a classifier needs to be trained on the data space to segment the data and labels. This classifier must reliably separate multiple data clusters and also classify new data points that are not part of the trained data space as 'unrecognized'. An SVM that creates a model of the data space is used towards this goal. Setting the parameters for the SVM is the process of actually training the facial recognition module.

To elaborate briefly on how the SVM works, the algorithm creates a model that segregates the data into a category of labels. This model is a mapped representation of the data in which the training data points of the separate labels are divided by an as wide as possible gap. The data points of each label that have the closest proximity in the data space to differently labeled data points are called the label's support vectors. The new face images that are presented to the SVM are then mapped onto the model space and predicted to belong to a label category, based on which side separating regions it falls on to. The SVM used for facial recognition in the current research is trained with OpenCV's 'train_auto' function on an SVM object of the 'CvSVM'[7] class. This function trains the SVM model automatically by selecting the optimal parameters from a given range per parameter. The most important parameters for the SVM training are the $C$ and $\gamma$ parameter, which are automatically chosen from a logarithmic scale ranging from $1*10^{-5}$ to $1*10^{5}$ for C, and $1*10^{-14}$ to 10 for $\gamma$. Since the step size for a logarithmic scale must be larger than 1, the step size for both previously named ranges is set to 5. Parameters are considered optimal when the cross-validation estimate of the test set error is minimal. The average training time on a total of 150 training images from 3 individuals (50 images per person) over 6 sessions was 22.4 seconds, which is manageable in the case that training happens before any mission starts (which is the default).

The advantages of SVMs are their high generalization ability in high feature spaces (which facial recognition is dealing with), their redefinition of the training data to a model which saves memory, their robustness against noise and overlap in the training data, and the lack of parameter tuning requirement because the SVM will figure out the parameters by itself (Joachims, 1998). Disadvantages of SVMs can be the training time

[7]OpenCV's 'CvSVM' class description and it's functions can be found at
`http://docs.opencv.org/modules/ml/doc/support_vector_machines.html`

and computational demand, while they could take some irrelevant features into account. Tests showed that the former disadvantage appeared not to be an issue in the current research.

### 4.3.3 The facial recognition module

The actual facial recognition is performed by the facial recognition module that makes use of the data set and SVM discussed previously. It will attempt to match detected faces in the frames from the cameras on the vehicles to the training set. For the detection of faces, the same Haar detector is used as in the training phase. The module expresses the degree of recognition in terms of a confidence value that ranges between 0 and 1, being a total mismatch and being a complete match respectively. The confidence value is compared to an adjustable parameter named $confidenceThreshold$ which sets the threshold of confidence needed for recognition. The threshold parameter is the critical boundary which determines whether the new face is marked as a positive recognition or as an unknown face. If a face in a frame is recognized, the label of the face in the data space is retrieved and produced both as visual output on the recognition window (including label and confidence value) for the user to review and sent as a customized message over ROS. The message contains the label, the x and y pixel coordinates, and the height and width pixel values of the recognized face in the current frame. When a face is detected but not recognized (the confidence value does not exceed the value in $confidenceThreshold$) the label is absent in both the output window and the ROS message. The lack of this label in the message is a cue for the behavior that a face is detected but not recognized. When such a rejection occurs, the module outputs a different 'most-likely' label indicating the person to be most likely in view of the camera. This label is not shown in the output window. Figure 4.4 shows two processed frames from the facial recognition module on a video feed from a UGV, one in which a person with label "Bob" is correctly recognized and one in which an unknown person is correctly rejected (no label is shown). The input imagery for the facial recognition module from the vehicles, i.e. a facial view from almost ground level 30° upwards from a distance of approximately 3 meters, is not as ideal as a close up view at human facial height. Despite of the non ideal camera angle and distance to the subject's face, trials showed that the facial recognition module still performed satisfactory.

### 4.3.4 Conditions on facial recognitions

When the facial recognition module is performing within the behavior, errors occurring from noise and false-positives must be taken into account. Therefore a person must be recognized for a minimal time duration for the system to acknowledge the recognition, i.e. a certain number of face recognitions must occur within a set timespan. In the current research the recognition number versus the recognition timespan was set to 7 recognitions within 10 seconds. Using these parameters it is assumed that the frame rate would not drop under 1 fps, which has never occurred during tests and trials. The average frame rate of the facial recognition module was 4.9 fps. If no 7 recognitions were made within


(a) Correct recognition (b) Correct rejection

Figure 4.4: Examples from the facial recognition module on frames from a UGV

10 seconds the facial recognition module was deactivated and the remainder of the initial mission was executed. The number of rejections made by the facial recognition module during the 10 seconds of activation determined whether the module produced a subject classification as 'unknown' (enough rejections) or no classification at all (not enough rejections). Another condition imposed on the facial recognition module was that the dimensions and position of the recognized faces has to stay similar to some degree over the given timespan. The latter condition held rather large margins to account for some lag or noise in the system, while the recognition time and number were strict parameters. For all facial recognition parts of the experiments no conditions were imposed on the subjects their facial appearance, as long as their face was not obstructed by a mask or an equivalent concealing item. Other accessories like glasses or long hair partially occluding a forehead were allowed. These allowances in combination with a group of test subjects that varied heavily in appearance ensured robust performance testing on a diverse group of individuals.

### 4.3.5 Measuring facial recognition performance

Due to the nature of the facial recognition module, the performance is scored per subject that is in view of the camera at the time the module is activated. Statistics are derived from subject occurrences in the frames, correct classifications, correct rejections, false-positives, and confidence values. The measures per subject are (1) the total number of

frames stored while the facial recognition module was active, (2) the number of frames in which a subject recognition is made with respect to the $confidenceThreshold$ parameter (the correctness of the output label does not matter here yet), (3) the number of correct classifications of the subject in view, (4) the number of false-positives, i.e. incorrect labels in recognitions, (5) the number of frames in which no face is detected at all, (6) the number of frames in which a rejection occurred with respect to the parameter $confidenceThreshold$, (7) the number of rejections that nevertheless had a correct 'most-likely' label, (8) the average confidence value output when a face was detected, either recognized or unrecognized, over the entire facial module activation period, and (9) the total time that the facial recognition module was activated. Measures 3 and 7 were also calculated as a percentile of their respective categories. Note that if an unknown subject is in view (a subject that is not in the dataset), measure 3 transforms into the number of correct rejections, since a rejection is a correct classification in this case. The false-positives in measure 4 are all classifications which the classifier deems correct, but are in fact faces of other subjects that are not in view or subject recognitions in the background.

Shifting the $confidenceThreshold$ parameter yields various results. Through trial and error a value between 0.70 and 0.75 was found to yield the best results depending on the present lighting conditions. For experiments of type 1 a dataset of 3 out of the 6 test subjects was recorded with the procedure discussed above and the $confidenceThreshold$ was set to 0.72 (normal lighting conditions). For the experiments of type 2 a dataset of 4 people was recorded of which one was to be recognized and the $confidenceThreshold$ was set to 0.70 (very bright sunlight conditions). Note in experiment type 2 all recognitions with labels other than the one opted for recognition were false-positives. The conditions discussed in section 4.3.4 only applied to experiments of type 1. In the experiments of type 2 there was no time limitation on recognitions or rejections, but the experiments were ended by user interference after it became clear that the experiment was a success or a failure.

# Chapter Five

# Results and Discussion

## 5.1 Human detection classifiers

The results of the benchmark tests for the pedestrian view classifiers on the UGV field dataset are shown in figure 5.1. The results of the benchmark tests for all the classifiers (pedestrian view and top view) on the UAV field dataset are shown in figure 5.2 where a different scale for the secondary axis of the figure is used to show the frame rate section. Results of all benchmarks can be reviewed in Appendix A.1. Shown measures of the benchmarks results are correct detections, false-positives, and frame rate (fps). The percentage of correct detections of the classifiers on all datasets are shown in table 5.1. Here the number of correct detections is calculated as a percentile of the total number of subjects shown in the positive images as described in table 4.1. Note that the percentage of correct detections are independent of false-positive numbers which can be reviewed in Appendix A.1.

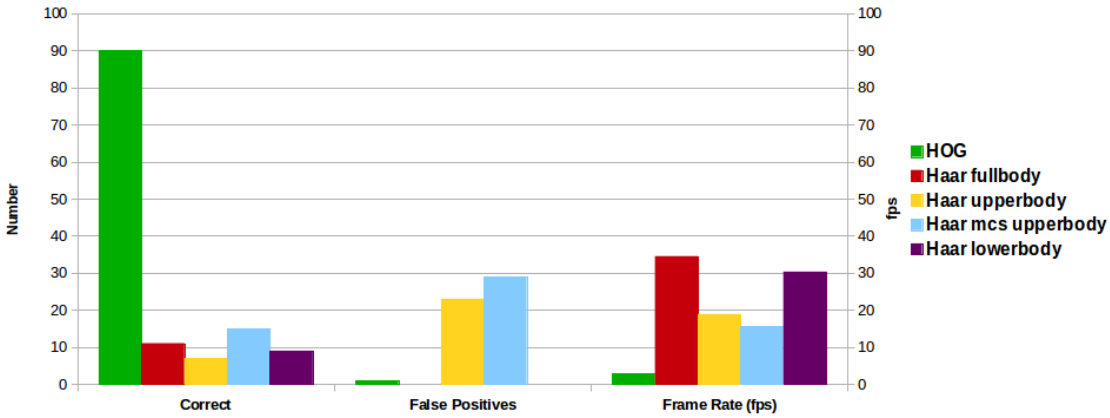



Figure 5.1: Benchmark results of the 5 pedestrian classifiers on the UGV field dataset with the measures correct detections, false-positives, and frame rate.

Optimal detection performance is observed from the HOG classifier on both the UGV

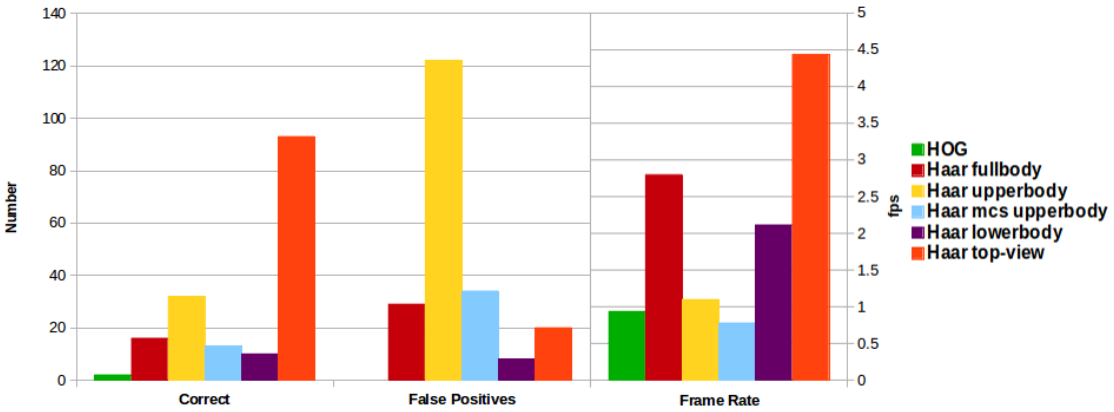



Figure 5.2: Benchmark results of all 6 classifiers on the UAV field dataset with the measures correct detections, false-positives, and frame rate.

| | **UGV field** | **UAV field** | **UGV campus** | **INRIA** |
|---|---|---|---|---|
| HOG | 90.0% | 1.6% | 76.0% | 61.4% |
| Haar fullbody | 11.0% | 12.5% | 0.0% | 43.7% |
| Haar upperbody | 7.0% | 25% | 14.0% | 43.7% |
| Haar mcs upperbody | 15.0% | 10.2% | 15.0% | 58.9% |
| Haar lowerbody | 9.0% | 7.8% | 5.0% | 61.4% |
| Haar top-view | 32% | 72.7% | 64% | 74.7% |

Table 5.1: Normalized correct classification results of classifiers on datasets

field and the UGV campus datasets. The created Haar top-view classifier outperforms all other classifiers on the UAV field dataset. In benchmarks on the INRIA dataset the Haar top-view classifier has the best correct detections score, but also the highest number of false-positives as can be seen in Appendix A.1. The HOG classifier on the other hand has an equivalent or higher score than the other pedestrian classifiers but the lowest false-positive count. In terms of processing time the HOG classifier is either equivalent to other classifiers in the INRIA test or much slower in other sets. Even though Haar classifiers prove to process real world data significantly faster than the HOG classifier in most cases, the latter still achieves an average processing frame rate of 2.8 fps. In a pedestrian view from the agents in the current research this is a sufficiently high rate for human detection at the agent's operating and navigation speed. Therefore the HOG classifier was used for human detection in pedestrian view data from the UGVs in further experiments. For the top view scenarios the Haar top-view classifier is chosen for human detection since it outperforms all other classifiers in every measure, except for the false-positive count of the Haar lowerbody classifier.

## 5.2 Person localization

As discussed in section 4.2.3 the person localization performance was measured in 5 experiments of two experiment types. A results representation of the first experiment of type 1 is shown in a Google Maps view in figure 5.3. The representation includes actual person locations (yellow markers), a search area (dark blue markers and borders), subject detections (smaller red markers), prototypes (purple markers), and a base station (green marker). The first agent produced 14 subject detections and 2 prototypes, and the second agent produced 10 subject detections and 2 prototypes.

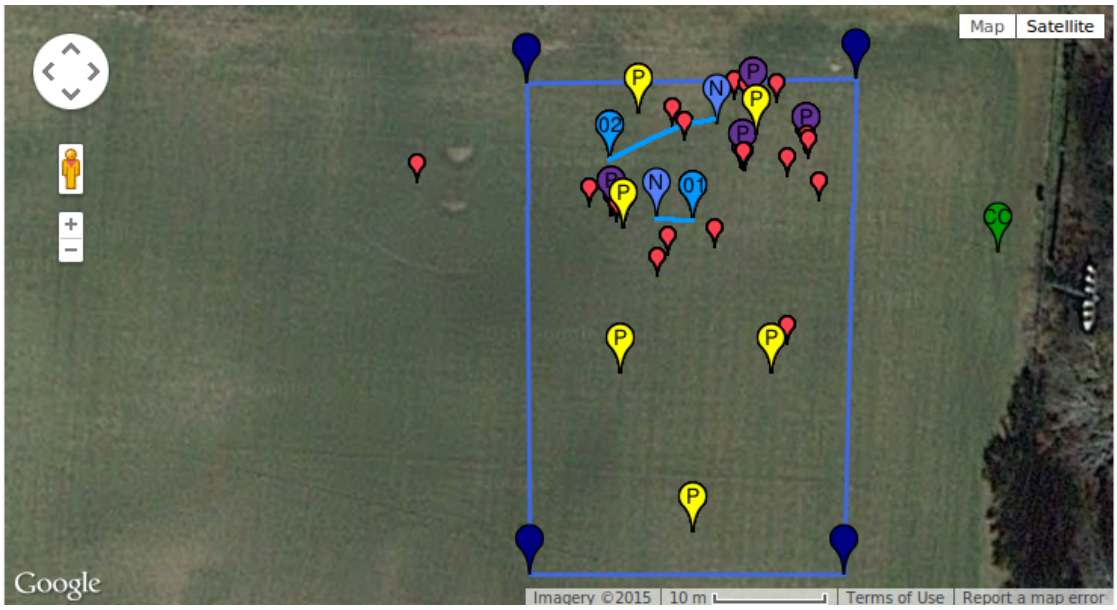



Figure 5.3: Google Maps view of experiment type 1 with results of two operating agents

Figure 5.4 shows the results of the first experiment of type 2. Next to the inclusion of the objects mentioned above for figure 5.3, the approach path of the agent (light blue line), the search path (dark blue line between start and end position), and the newly created waypoint (light blue marker containing an 'N') are also shown. The agent approached the subject from the south. Representations of all the results of the experiments are shown in Appendix A.2.

The experiments were scored on the 8 measures discussed in section 4.2.3. The measure results on all experiments are shown in table 5.2.

The localization results of the experiments will be reviewed here, while the facial recognition results will be discussed further in section 5.3. Experiment 1 of type 1 included a very large number of correct person detections and all prototypes were determined to be correct. Although 4 correct prototypes were created, 3 of those prototypes were representing the same test subject. Therefore the number of people 'found' is 2 out of 6. Two useful waypoints were created that led to the activation of the facial recognition module in which two subjects were in view of the camera at the time of module

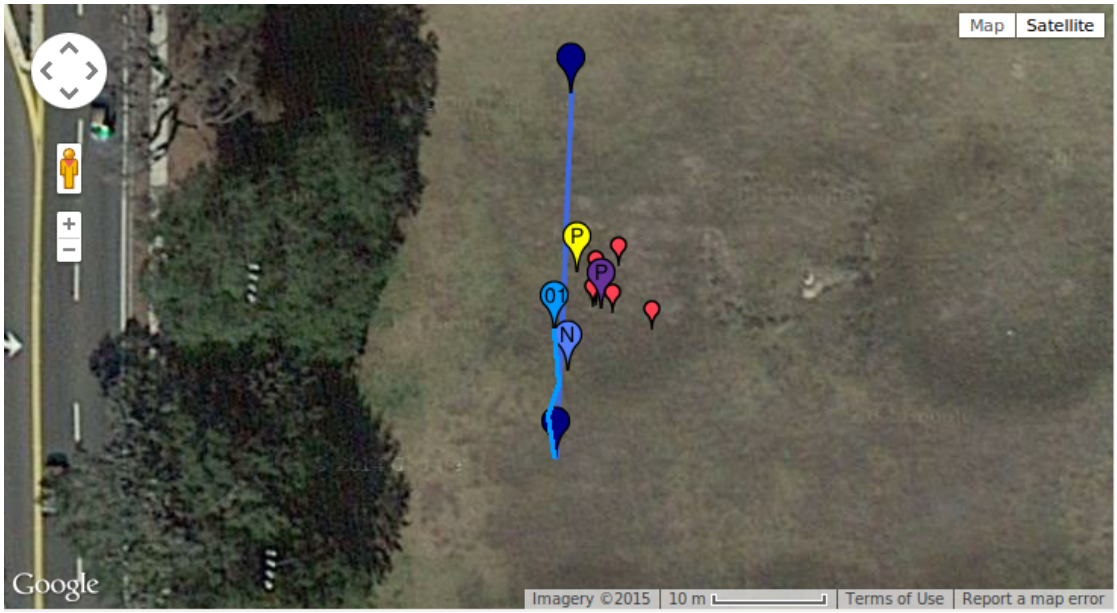



Figure 5.4: Google Maps view of experiment 1 of type 2 with the agent's driven path

| | Experiment Type 1 | | Experiment Type 2 | | |
|---|---|---|---|---|---|
| Experiment | 1 | 2 | 1 | 2 | 3 |
| # Detections | 24 | 29 | 6 | 5 | 4 |
| Correct detections | 21 (87.5%) | 8 (27.6%) | 5 (83.3%) | 1 (20%) | 3 (75%) |
| False-positives | 3 (12.5%) | 21 (72.4%) | 1 (16.7%) | 4 (80%) | 1 (25%) |
| # Prototypes | 4 | 2 | 1 | 1 | 1 |
| Correct Prototypes | 4 (100%) | 0 (0%) | 1 (100%) | 0 (0%) | 1 (100%) |
| False-positives | 0 (0%) | 2 (100%) | 0 (0%) | 1 (100%) | 0 (0%) |
| # Found subjects | 2 (33.3%) | 0 (0%) | 1 (100%) | 0 (0%) | 1 (100%) |
| Useful waypoints | Yes | No | No | No | Yes |

Table 5.2: Results of all experiments in 8 measures

activation. Of these two subjects one was in the dataset and the other was not.

Experiment 2 of type 1 yielded worse results than experiment 1. Only 27.6% of the detected person locations were in the vicinity of test subjects and two incorrect prototypes were generated. The mission data showed the reason for this poor performance, namely an internal process failure of an agent. The agent's location and heading were not updated anymore, while the agent's person detection kept working, since that module is handled on the base station. The mismatch in agent status and detections resulted in a cluster of detections that was placed on the wrong location away from the actual test subject. Three other test subjects were also detected a few times, but the detection rate was not high enough for a prototype creation (a cluster of at least 3 detections within a range of 3 meters must be formed for a prototype to be created). Due to the agent's process failure and the lack of prototypes from the other agent no useful waypoints were

created and the facial recognition module was not usefully activated.

In experiment 1 of type 2 the subject was detected multiple times, a correct prototype was generated, and the agent had stopped to investigate the subject. A dynamic waypoint was created, but was not useful. Due to the lack of brakes on the vehicle the agent had driven past the waypoint after it was created. If a subject is detected at a large distance, a 'roll-out' does not become a an issue since a relatively small distance is additionally traveled. Here on the other hand, the agent was detecting a subject at a small distance which makes the roll-out distance relatively more problematic. Both at the moment of stopping for investigation and after driving to the new waypoint the subject was not in view. Therefore the useful waypoint measure is scored as 'No' and the facial recognition module was not usefully activated.

Experiment 2 of type 2 involved a similar internal agent process failure as was observed in experiment 2 of type 1. The agent's heading and location were not updated properly which led to an incorrect localization of the test subject. This was also the reason no dynamic waypoint could be created, even though a prototype was present. Fortunately, at the moment of the process failure, the agent was fairly close to the subject and had the subject in view of the camera. With a user override, the facial recognition module was activated to see if the subject could be recognized.

Finally, experiment 3 of type 2 shows the best performance. The test subject was detected and localized correctly through a prototype at 2 meters distance of the subject, while a dynamic waypoint between the agent and the subject was created. The waypoint was also useful because the subject was fully in view after the agent had reached the waypoint which led to the useful activation of the facial recognition module.

## 5.3 Facial recognition

### 5.3.1 Facial recognition in experiment type 1

Only in experiment 1 of type 1 was the facial recognition module usefully activated. The results of the facial recognition modules for the two agents are shown in table 5.3 with the measures discussed in section 4.3.5. Note that agent 1 had a subject in view that was in the dataset for facial recognition, while agent 2 had a subject in view that was not in the dataset. From the results in table 5.3 we can conclude that agent 1 has successfully recognized its test subject and agent 2 has successfully rejected its test subject, i.e. classified as 'unknown'. Agent 1 made 7 correct classifications, after which the module was deactivated due to the applied conditions on the facial recognition module discussed in section 4.3.4. Even the 'most-likely' labels from the rejections were all correct. Agent 2 had recorded a total of 60 frames out of which 48 frames (80%) were correctly classified as a rejection. Thus after 10 seconds of facial recognition module activation the subject was correctly classified as 'unknown'. Average confidence values of both agents support these conclusions in a high average confidence value for agent 1 (symbolizing recognition) and a low confidence value for agent 2 (symbolizing rejection). The facial recognition in experiment 1 of type 1 was deemed a success.

| | Agent 1 | Agent 2 |
|---|---|---|
| Subject in dataset | Yes | No |
| # Total frames | 19 | 60 |
| # Recognition frames | 7 | 4 |
| # Correct classifications | 7 (100%) | 48 (80%) |
| # False-positives | 0 | 4 |
| # No face detections | 2 | 8 |
| # Rejections | 10 | 48 |
| # Correct 'most-likely' labels | 10 (100%) | 48 (80%) |
| Average confidence value | 0.7150 | 0.6632 |
| Total time (s) | 4.31 | 10.35 |

Table 5.3: Results of the facial recognition in experiment 1 of type 1 for two agents

### 5.3.2 Facial recognition in experiment type 2

In two of the three experiments in experiment type 2 the facial recognition module was activated, namely in experiment 2 and 3. The results of the facial recognition modules for the two experiments are both shown in table 5.4 with the measures discussed in section 4.3.5. Due to the setup of the experiments the test subject was in the dataset for both experiments.

| | Experiment 2 | Experiment 3 |
|---|---|---|
| Subject in dataset | Yes | Yes |
| # Total frames | 25 | 248 |
| # Recognition frames | 25 | 140 |
| # Correct classifications | 0 (0%) | 117 (83.6%) |
| # False-positives | 0 | 23 |
| # No face detections | 25 | 24 |
| # Rejections | 0 | 108 |
| # Correct 'most-likely' labels | 0 (0%) | 65 (60.2%) |
| Average confidence value | n/a | 0.7046 |
| Total time (s) | 21.1 | 52.7 |

Table 5.4: Results of the facial recognition in experiment 2 & 3 of type 2

After reviewing the results and data from experiment 2, it was revealed that no face was detected, and thus not recognized, during this experiment. The location of the agent was north of the test subject, which resulted in the sun shining straight into the camera causing underexposure of the face. Only the contour of the subject was seen and no face could be recognized, not even by a user inspection. For example purposes the vehicle was positioned south of the subject. Differences between northern and southern position of the vehicle with respect to the subject are shown in figure 5.5 to illustrate sun influence on module performance.


(a) Northern vehicle position with underexposure of the face

(b) Southern vehicle position with a correct performance

Figure 5.5: Facial recognition module result of two vehicle positions facing the subject

Results from experiment 3 showed that there were faces detected and recognized by the facial recognition module. Note that the false-positives in table 5.4 were incorrect recognitions of the other subjects in the dataset that were not in front of the camera. A number of 117 out of 140 recognition frames (83.6%) were correct classifications and 65 out of 108 rejections (60.2%) had the correct 'most-likely' label. The average confidence value exceeds 0.70 which made the recognition robust. Facial recognition in experiment 2 of type 2 was deemed a failure, while the facial recognition in experiment 3 of type 2 was a success.

# Chapter Six

# Conclusions and Future Work

## 6.1 Conclusions

In this research methods for the detection and recognition of humans in an outdoor environment by an autonomous heterogeneous swarm of agents were proposed and tested. A combination of human detection classifiers, dynamic navigation methods, and facial recognition were implemented towards this goal. The swarm consisted of ground and aerial vehicles for which a software architecture was in place to perform the autonomous navigation of the agents. Camera modules were installed on the agents to provide live video imagery for the human detection and recognition.

Towards the goal of robust human detection from both types of agents in the swarm, multiple human detection classifiers were tested. A great performing pedestrian HOG classifier was chosen for human detection on the ground agents. A custom Haar classifier was created in this research for the detection of humans from a top view. The created classifier was tested on datasets consisting of images of humans from a top view and it proved to outperform all other existing classifiers in these benchmark tests. Classifiers showed to be non interchangeable between viewing scenarios. Therefore the HOG classifier was merely used for the pedestrian view human detection tasks and the created Haar top-view classifier for the top view human detection tasks.

The localization algorithms performed fairly well in determining the location of the subjects after detection, provided that the agents functioned correct without internal process failures and the discussed assumptions were met. False-positive detections were sifted out effectively by a clustering algorithm in combination with subject representation through prototypes. Multiple dynamic approach patterns were created and tested, after which the simplicity of the single waypoint in combination with the agent's behavior as a result from the setup of the architecture (including waypoint and GPS inaccuracy) proved to be the most effective pattern.

The facial recognition module performed satisfactory on the imagery from the UGVs even though the angle from camera to face was not ideal. The performance of the module within the entire behavior was heavily dependent on the dynamic navigation module, which in turn is dependent on the human detection module. Environmental factors

and parameter settings also played a significant role in the performance of the facial recognition module. In spite of the previously named dependencies, module performance in the final experiments showed robust facial recognitions and rejections, therefore the module was deemed as performing satisfactory.

Even though the UAVs could not be deployed in the final experiments due to the limitations on flying these aircrafts for the current research by the Federal Aviation Administration, the present research does provide a proof of concept of the UAVs functioning and usefulness within the used architecture including human detection. A reliable top view human detection classifier was created through Haar cascade training, which in combination with the localization algorithm using adjusted parameters for a different viewing angle, and the clustering algorithms, proves that human detection and localization can also be performed through the use of UAVs. In theory a top view overview of a search area through the use of UAVs would significantly decrease search time and subject location certainty, in part because the UAVs have a larger area of view with their cameras and the background behind the subjects would be less busy than from a pedestrian view (green grass versus trees and a sky). Therefore the deployment of UAVs is deemed to presumably be a very useful addition to the human detection modules within the current architecture. The reason for not implementing and testing facial recognition with the UAVs is because potential hazardous situations could occur if such a vehicle would come up close to a subject, which is a requirement for the currently used facial recognition module. Instead the locations of the subjects would be transferred as a task for the UGVs to go and investigate the detected subjects. All of the software for this behavioral concept was already in place in the current research through the generality of the architecture and the modules.

The main research questions that drove the current research consisted of comparing human detection algorithms, creating and testing the functional use of dynamic search patterns with the agents, and implementing and testing facial recognition on any detected and approached subjects. The functionality and performance of the three created modules are the product of answering the three research questions. The human detection module was a result of the implementation and comparison of the different human detection algorithms. The dynamic navigation module was a result of creating and testing different dynamic search patterns to approach test subjects. And finally, the facial recognition module was a result of implementing and testing facial recognition in the current research.

Concluding, a complete system for the autonomous detection and recognition of human subjects through dynamic navigation with a heterogeneous swarm of autonomous agents was created in this research. Even though certain conditions were imposed and environmental factors had some influence, the entire system illustrated a good performance within a noisy outdoor environment on a diverse group of subjects in the conducted experiments, if internal agent processes functioned as they should. The performance of the human detection module and the facial recognition module on a diverse group of subjects, shows those modules to be robust and reliable assets to the system.

## 6.2 Future Work

A number of proposals for future work in human detection and recognition through dynamic navigation with the currently used swarm of agents will be discussed here. Hardware improvements from which the system would greatly benefit include sensor upgrades to for example thermal or infrared imaging sensors and processing board upgrades. The thermal sensors would be able to detect body heat instead of body appearance of the human subjects which would significantly increase detection rates, an idea illustrated by Breckon et al. (2011). Upgrading the agents their CPUs and GPUs would enable the agents to process their imagery on-board on a feasible frame rate, which in turn would dismiss the use of Wi-Fi connections and could also lead to upgrading to cameras with a better resolution. This would certainly improve system performance and functionality.

Using the parallel computing platform and programming model CUDA[1] to process the input images for human detection and recognition on GPUs (either off-board or on-board) would also increase the performance of the human detection and facial recognition modules significantly.

The system would also benefit from improving the agent's position accuracy through for example odometry (using data from motion sensors on the wheels to estimate change in position over time) or magnetoception (using the detection of the earth's magnetic field to perceive the agent's direction, altitude or location). The performance in agent localization with the currently used GPS module in combination with the system's setup to reckon with the GPS's inaccuracy and drift, yields some unwanted agent behaviors like holding near a waypoint while not reaching it. Relying on more sensor input than solely the GPS module would theoretically benefit the system.

In terms of approaching subjects for facial recognition with dynamic navigation a new driving pattern is proposed here. In this drive pattern the current position of the sun would be included in determining the angle from which the subject is approached. Sun position would be computed from the general location and the time at which the experiments are held. A waypoint pair would be generated, like the ones used in the star pattern discussed in section 4.2.2, that is in line with the sun and the subject. For a clear representation of this proposal see figure 6.1. This proposal was made to theoretically gain the best angle for facial recognition since the sun shines straight on the subject's face, and moreover does not shine into the camera causing underexposure of the face. The facial lighting appeared to be a large factor in facial recognition, as discussed in section 5.3.

[1] More information on CUDA can be found at `https://developer.nvidia.com/about-cuda`

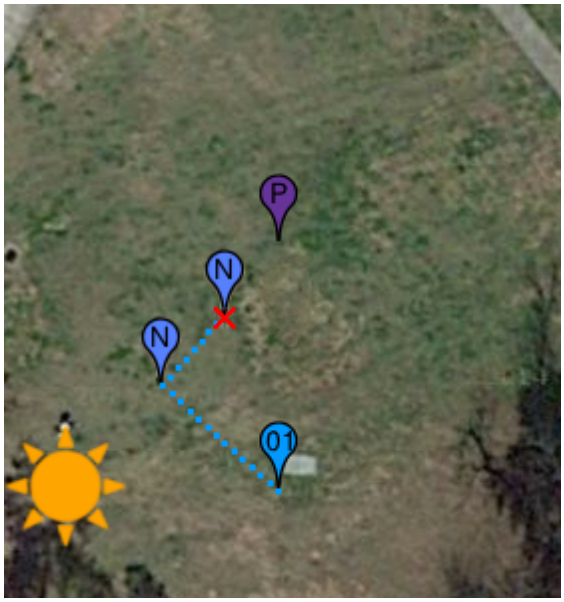
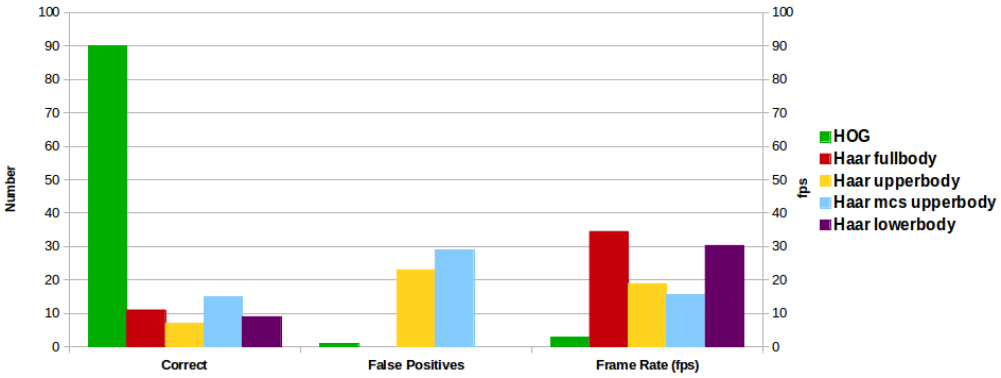



Figure 6.1: A dynamic driving pattern to approach a prototype dependent on the current position of the sun

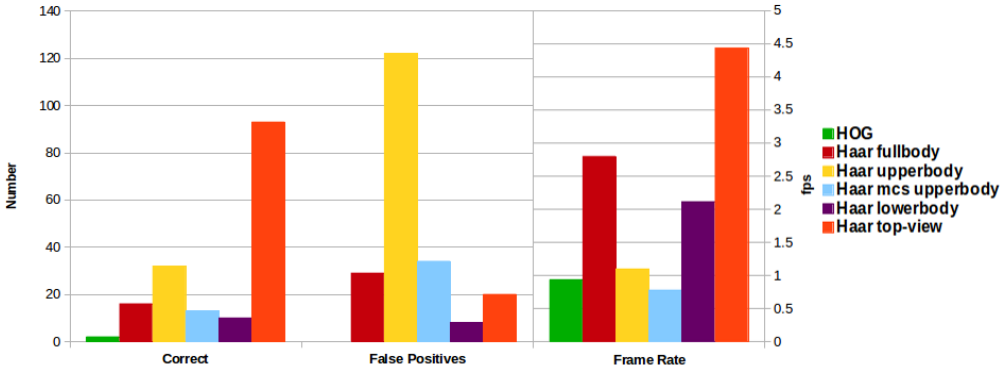
# Appendix A

## A.1 Benchmark results

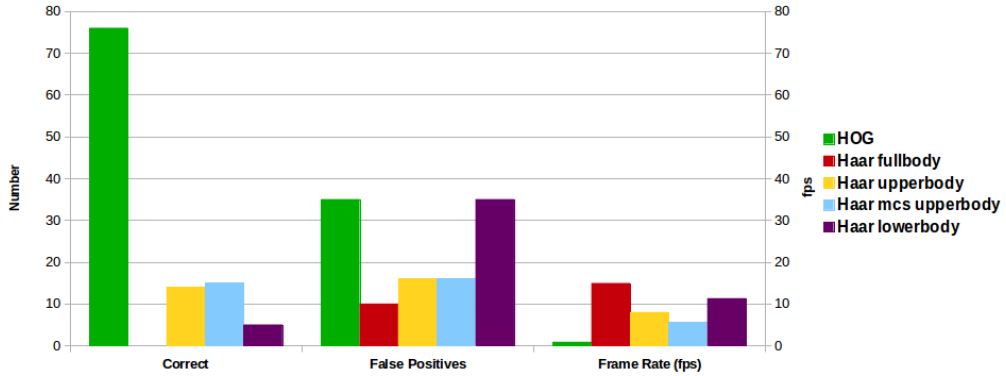
To review all the benchmarks from the human detection classifiers on four datasets, results were plotted in bar graphs. Benchmarks were scored on three measures, namely correct classifications, false-positives, and processing rate in frames per second. For the UGV field and UGV campus datasets the 5 pedestrian classifiers were tested, namely HOG, Haar fullbody, Haar upperbody, Haar mcs upperbody, and Haar lowerbody. All six classifiers (including the Haar top-view classifier) were tested on the UAV field and INRIA dataset. The result graphs on the four datasets (a) UGV field, (b) UAV field, (c) UGV campus, and (d) INRIA are shown in figure A.0. Note that figure A.0c and A.0d have a different scaled secondary axis to show the frame rate.



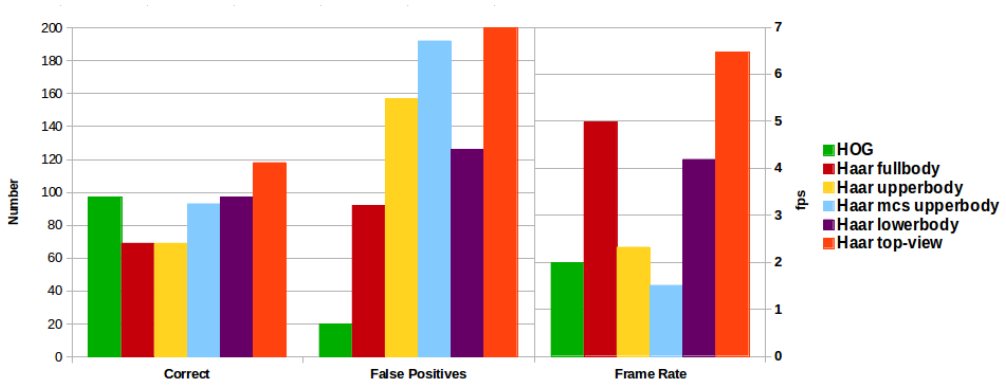

(a) Benchmark results from 5 pedestrian classifiers on the UGV field dataset including 200 images with 100 positive examples




(b) Benchmark results from 6 classifiers (pedestrian and top view) on the UAV field dataset including 200 images with 128 positive examples




(c) Benchmark results from 5 pedestrian classifiers on the UGV campus dataset including 200 images with 100 positive examples

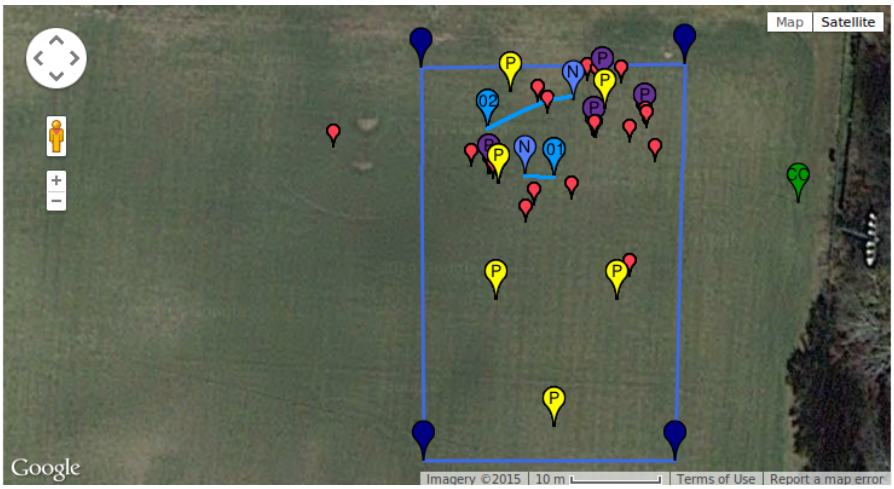



(d) Benchmark results from 6 classifiers (pedestrian and top view) on the INRIA dataset including 200 images with 158 positive examples

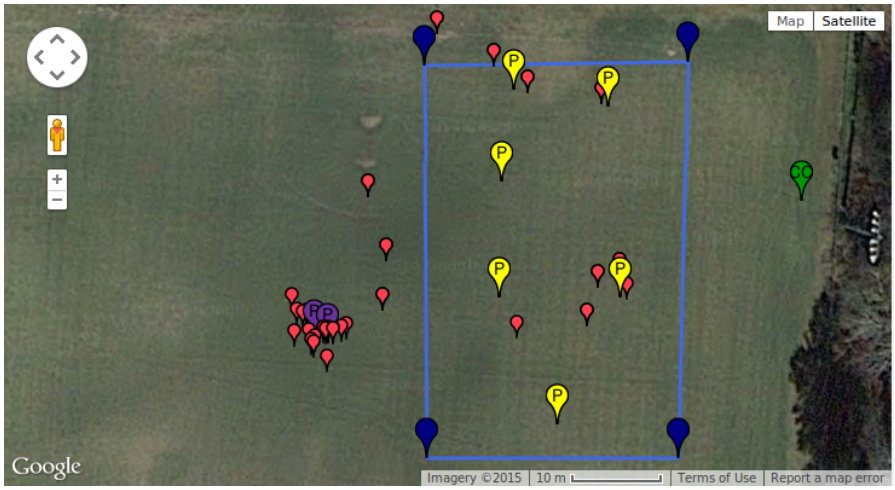
Figure A.0: Benchmark results of six classifiers on the four datasets with the measures correct classifications, false-positives, and process time.

## A.2 Person localizations

Result representations of the experiments may include actual person locations shown as yellow markers, a search area or direct path as dark blue markers and borders, subject detections as smaller red markers, prototypes as purple markers, a base station as a green marker, an agent path as light blue line, or newly created waypoints as light blue markers containing an 'N'. Results of the two experiments of experiment type 1 are shown in a Google Maps view in figure A.1. Both experiments were performed by two simultaneously operating agents. Results of the three experiments of type 2 are shown in a Google Maps view in figure A.2.

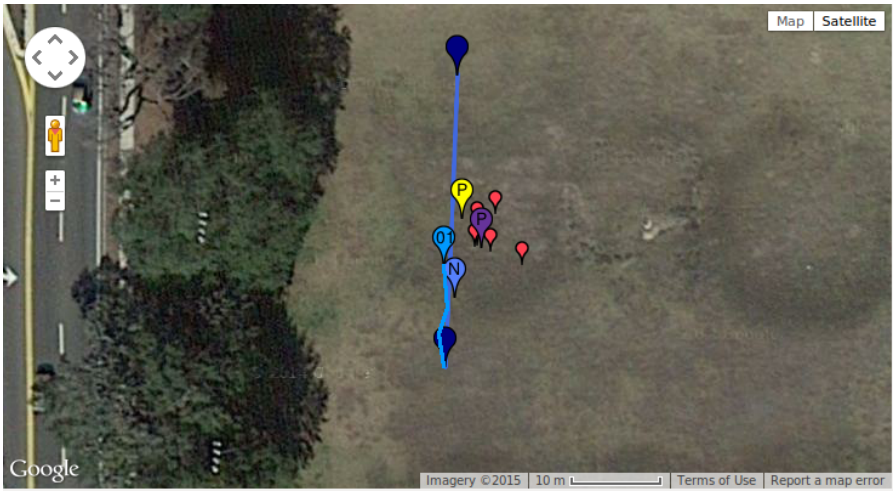



(a) Experiment 1

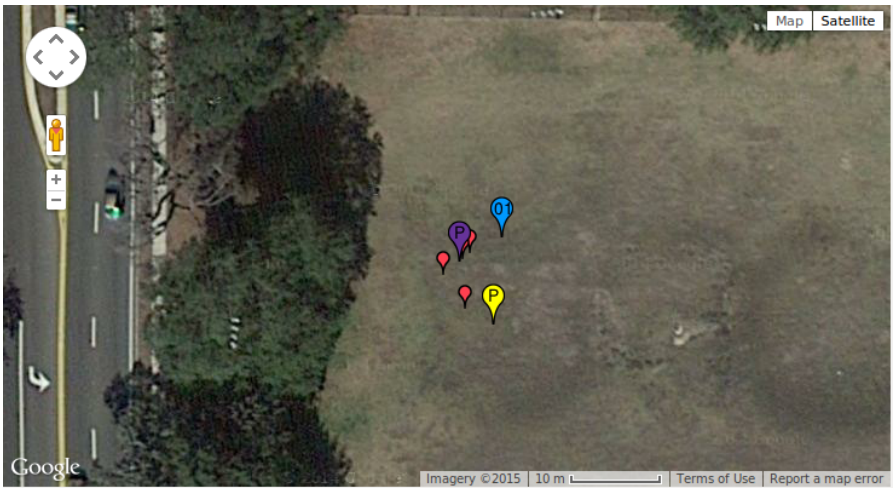



(b) Experiment 2

Figure A.1: Results representations of the two experiments in type 1 in a Google Maps view




(a) Experiment 1




(b) Experiment 2


(c) Experiment 3

Figure A.2: Results representations of the three experiments in type 2 in a Google Maps view

# References


Andriluka, M., Schnitzspan, P., Meyer, J., Kohlbrecher, S., Petersen, K., Von Stryk, O., et al. (2010). Vision based victim detection from unmanned aerial vehicles. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* (pp. 1740–1747).

Baggio, D. L., Emami, S., Escriva, D. M., Ievgen, K., Mahmood, N., Saragih, J., et al. (2012). *Mastering OpenCV with practical computer vision projects.* Packt Publishing Ltd.

Beni, G. (2005). From swarm intelligence to swarm robotics. In *Swarm Robotics* (pp. 1–9). Springer.

Breckon, T. P., Gaszczak, A., Han, J., Eichner, M. L., & Barnes, S. E. (2013). Multi-modal target detection for autonomous wide area search and surveillance. In *SPIE Security+ Defence* (pp. 889913–889913).

Breckon, T. P., Han, J. W., & Richardson, J. (2012). Consistency in multi-modal automated target detection using temporally filtered reporting. In *SPIE Security+ Defence* (pp. 85420L–85420L).

Campoy, P., Correa, J. F., Mondragón, I., Martínez, C., Olivares, M., Mejías, L., et al. (2009). Computer vision onboard UAVs for civilian tasks. In *Unmanned Aircraft Systems* (pp. 105–135). Springer.

Castrillón-Santana, M., Déniz-Suárez, O., Antón-Canalís, L., & Lorenzo-Navarro, J. (2008). Face and Facial Feature Detection Evaluation-Performance Evaluation of Public Domain Haar Detectors for Face and Facial Feature Detection. In *VISAPP (2)* (pp. 167–172).

Chellappa, R., Wilson, C. L., & Sirohey, S. (1995). Human and machine recognition of faces: A survey. *Proceedings of the IEEE*, *83*(5), 705–741.

Chen, H., & Bhanu, B. (2004). Human ear detection from side face range images. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* (Vol. 3, pp. 574–577).

Chun, B.-G., & Maniatis, P. (2009). Augmented Smartphone Applications Through Clone Cloud Execution. In *HotOS* (Vol. 9, pp. 8–11).

Cohen, B. J., Chitta, S., & Likhachev, M. (2010). Search-based planning for manipulation with motion primitives. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on* (pp. 2902–2908).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In *Swarm robotics* (pp. 10–20). Springer.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886–893).

Flynn, H., & Cameron, S. (2013). Multi-modal People Detection from Aerial Video. In *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013* (pp. 815–824).

Gaszczak, A., Breckon, T. P., & Han, J. (2011). Real-time people and vehicle detection from UAV imagery. In *IS&T/SPIE Electronic Imaging* (pp. 78780B–78780B).

Gavrila, D. M. (1999). The visual analysis of human movement: A survey. *Computer vision and image understanding*, *73*(1), 82–98.

Heseltine, T., Pears, N., & Austin, J. (2002). Evaluation of image pre-processing techniques for eigenface based face recognition. In *SPIE* (Vol. 4875, pp. 677–685).

Islam, S. M., Bennamoun, M., & Davies, R. (2008). Fast and fully automatic ear detection using cascaded adaboost. In *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on* (pp. 1–6).

Jee, H., Lee, K., & Pan, S. (2004). Eye and face detection using SVM. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004* (pp. 577–580).

Joachims, T. (1998). *Text categorization with support vector machines: Learning with many relevant features*. Springer.

Karaaba, M. F., Schomaker, L., & Wiering, M. (2014). Machine learning for multi-view eye-pair detection. *Engineering Applications of Artificial Intelligence*, *33*, 69–79.

Kruppa, H. (2004). *Object detection using scale-specific boosted parts and a Bayesian combiner*. Diss., Technische Wissenschaften, Eidgenössische Technische Hochschule ETH Zürich, Nr. 15676, 2005.

Leira, F. S. (2013). *Infrared Object Detection & Tracking in UAVs*. Unpublished master's thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on* (Vol. 2, pp. 1150–1157).

Meier, L., Tanskanen, P., Fraundorfer, F., & Pollefeys, M. (2011). Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and automation (ICRA), 2011 IEEE international conference on* (pp. 2992–2997).

Mondragon, I. F., Campoy, P., Correa, J. F., & Mejias, L. (2007). Visual model feature tracking for UAV control. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on* (pp. 1–6).

Papageorgiou, C., & Poggio, T. (1999). Trainable pedestrian detection. In *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on* (Vol. 4, pp. 35–39).

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., et al. (2009). ROS: an

open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, p. 5).

Ray, L. A., & Nicponski, H. (2005, September 6). *Face detecting camera and method.* Google Patents. (US Patent 6,940,545)

Rudol, P., & Doherty, P. (2008). Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery. In *Aerospace Conference, 2008 IEEE* (pp. 1–8).

Sobel, E. C. (1990). The locust's use of motion parallax to measure distance. *Journal of Comparative Physiology A*, *167*(5), 579–588.

Turk, M. A., & Pentland, A. P. (1991). Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on* (pp. 586–591).

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I–511).

Viola, P., Jones, M. J., & Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on* (pp. 734–741).

Walter, W. G. (1950). Imitation of Life.

Weaver, J. N. (2014). *Collaborative Coordination and Control for an Implemented Heterogeneous Swarm of UAVs and UGVs.* Unpublished doctoral dissertation.

Zhang, Z. (2012). Microsoft kinect sensor and its effect. *MultiMedia, IEEE*, *19*(2), 4–10.

Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face recognition: A literature survey. *Acm Computing Surveys (CSUR)*, *35*(4), 399–458.