Control Strategies for Heterogeneous, Autonomous Robot Swarms

Stefan Thamke[1], Markus Ax[1], Lars Kuhnert[1], Klaus-Dieter Kuhnert[1], Marco Langerwisch[2], and Thomas Remmersmann[3]

[1] Dept. of Realtime Learningsystems, University of Siegen
Hoelderlinstr. 3, 57076 Siegen, Germany
{stefan.thamke,markus.ax}@uni-siegen.de
{lars.kuhnert,kuhnert}@fb12.uni-siegen.de
[2] Institute of Systems Engineering, Leibnitz University of Hannover
Appelstraße 9A, 30167 Hannover, Germany
langerwisch@rts.uni-hannover.de
[3] Fraunhofer Institute for Communication, Information Processing and Ergonomics
Neuenahrer Straße 20, 53343 Wachtberg
thomas.remmersmann@fkie.fraunhofer.de

Abstract. Having robots as reliable and robust mobile sensor platforms in unknown environments is getting more and more attractive. The control of each robot as a single machine is often complicated enough. But if there are more than one robots in a given scenario, the task get's even harder. The operators then do not only have to care about steering their robot, but they also have to cooperate with each other. In this paper we describe the results of a research project regarding control strategies for a group of heterogeneous, autonomous robots. The swarm receives orders from a central control station, that uses a Battle Management Language, which is abstract, but human readable. An abstract language postulates a certain degree of intelligence within each robot of the cooperation, because the orders are mostly more complex than simple moves from A to B. There is a command hierarchy within the swarm, but every robot implements its own control strategies, to fulfill the overall goal of the orders. The technical as well as the operational realizations are described and discussed mostly with the focus on the unmanned aerial robots.

Keywords: unmanned aerial vehicles, control strategies, autonomous robots.

1 Introduction

Today single robots are getting more and more robust and reliable in terms of navigation, map building and data preprocessing. Therefore the next step is to combine several heterogeneous machines into a swarm. Even more interesting is the combination of ground and airborne systems at the same time to make use of each ones complementary features.

Unmanned ground vehicles (UGVs) are very useful for various tasks of reconnaissance and transportation, if they are tele-operated, or even autonomous. In that case human operators are greatly supported and do not have to enter possibly dangerous areas personally. Depending on their size, ground robots do have a high operating distance by being able to carry energy reserves, combined with huge loading capacities for cargo. Their handicaps on the other side are a typically limited maneuverability in heavy terrain and their relatively slow speeds.

Unmanned aerial vehicles (UAVs), especially ones that are capable of vertical takeoff and landing (VTOL), usually present complementary characteristics. Since they normally do not have to cope with obstacles they are highly maneuverable and can traverse great distances very fast. But their energy reserves are typically very low, which is directly proportional to loading capacity and flight duration.

Within the presented project, three different research groups had to incorporate their individual systems into one scalable organization. Each of these groups uses its own software implementation and communication protocol, which makes it necessary to define precise interfaces for global communication processes. Existing standards have been used to solve this problem and to enable even more systems to be integrated into the organization with very little implementational effort. This paper is focused mainly on the aerial parts of the project, but the overall concepts for all partners will be presented as well.

Generally ground air cooperation is a fast rising field of research, since UAV platforms are getting cheaper and more robust. But the amount of published results coping with the fusion of heterogeneous systems from different institutions is quite small. In [1] an UAV was used to enhance the localization quality of the ground vehicle. In [2] and [3] simulation tools are used to test theoretical control approaches. This completely eliminates the need for a communication middleware, since software implementations on the real system are completely faded out. In [4] UAVs are used to discover interesting areas in an urban surveillance scenario. After that is done, the ground vehicles are sent to these locations to execute further reconnaissance tasks.

2 Project Description

For military as well as for civil organizations it gets more and more important to have suitable tools at hand, while coping with (partly) unknown environments. Especially in dangerous areas like war zones or contaminated sites after a nuclear or chemical disaster, the need for unmanned and even autonomous robots will increase. Since there is not the one robot to solve all problems connected with reconnaissance, the cooperation of heterogeneous robots and the potential for a unified control method is evaluated using a realistic military scenario.

Our goal is to analyze the feasibility of a robot swarm, consisting of unmanned ground and aerial vehicles within a reconnaissance scenario. The main focus lies on the definition and standardization of interfaces used to control unmanned robots without having to know everything about their, often proprietary, software installations. The research is done in cooperation between the University of Siegen (EZLS), the Leibniz University of Hannover (RTS), and the Fraunhofer Institute for Communication, Information Processing and Ergonomics (FKIE).

In the following sections the most important components of the project will be described:

2.1 Battle Management Language

The main focus for evaluation lies on the concept of interpretation of an abstract but human readable language. The North Atlantic Treaty Organization (NATO) developed the battle management language (BML) as an unambiguous language used to command and control units and technical equipment conducting military tasks. Additionally there is a possibility for reporting information about the current status of units and the environment. Currently it is used for real troops and simulation purposes. The language is readable for machines in order to enable simulations and therefore should be usable for robot control too. Having a common standard for communication also enables the interoperability between forces from different members of the coalition.

The German version of BML is realized in the *Command and Control Lexical Grammar (C2GL) Specification* [5] and is used as the command language for this project. C2GL is human readable and the sentences are constructed by following a set of formal and context-free rules of a type 2 grammar. Thus sentences like "move to position-X as soon as possible" can be generated from a rule like "order → task where when". The arrow means that the symbol on the left side is transformed into a sentence defined by the rule on the right side of it. The non-terminal symbols *task*, *where* and *when* are defined by rules too and are transformed accordingly until only terminal symbols remain which represent a complete order.

An important part in reconnaissance scenarios is of course the information coming back from the reconnaissance unit. C2GL defines rules for so called *reports*, which are sent back to the control station. Reports do not intend the receiver to do something, but to inform him about something. They are also in most cases not as precise as orders, because a lot of information that is gathered through reconnaissance is normally attained without the counterpart knowing about being observed, or because of noisy sensor data. There are rules for various kinds of reports within C2GL (see [5]). The interesting ones for this project are task- and status reports. They inform the controller about the process of order execution, or the technical condition of the robots.

2.2 Robot Operating System

Choosing a language for expressing commands and reports raises the question of information transport. Since we do have a multi robot scenario, messages have to be distributed between these systems and the control station. Instead of implementing a proprietary interface, we chose the robot operating system (ROS) [6] as our communication middleware.

ROS does not fulfill the tasks of process management or scheduling, like traditional operating systems do. It is rather a communication service lying on top of a traditional operating system. The main goals in the development of ROS are to build a free and Open-Source, multi-lingual, peer-to-peer platform.

Since the commands and reports must be compatible to C2GL a ROS message is defined, which encapsulates all necessary data fields needed for communication. Additionally a converter is written, that translates pure C2GL messages into the ROS format and vice versa. This makes it very easy to connect processes from different partners together, because the interface is strictly defined and it is mostly platform and language independent.

2.3 User Interface

The graphical user interface (GUI) is designed by the FKIE. It is the standard user interface, which is used to communicate with military simulation environments employed by the NATO. Therefore it is natively only able to use C2GL as a communication protocol. Since ROS is used as the communication middleware, we implemented a C2GL to ROS connector, which translates between the two interfaces where possible. For many things that can be expressed in ROS there are no analogies in C2GL, like pictures, or generally sensor data. So we added a separate communication channel between the connector and the GUI which transports all not in C2GL expressible information.

2.4 UGV HANNA

The University of Hannover takes care of all work concerning the ground robot part of the project. They are using the unmanned ground vehicle *RTS HANNA* [7][8]. This robot is based on a Kawasaki Mule 3010, which is equipped with a drive-by-wire system from PARAVAN GmbH. It can move with up to 40 Km/h and has a maximum payload of 600kg. For environment perception the robot features 2D and 3D laser scanners and cameras. Together with a high precision localization module, backed by differential GPS and inertial sensors, it delivers high resolution occupancy grid maps of a given area, as well as live-video.

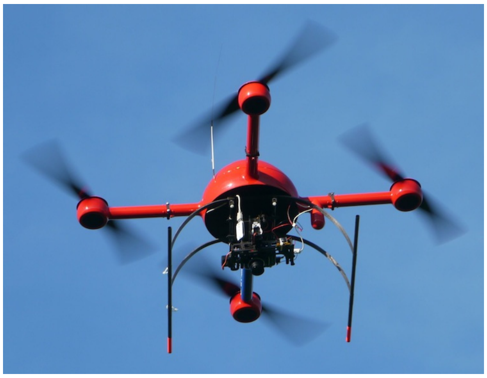

Fig. 1. Unmanned aerial vehicle PSYCHE 1000

2.5 UAV Psyche 1000

Within this project the University of Siegen utilizes two UAVs *Psyche 1000* (see Fig. 1) which are modified drones *MD4-1000*, originally built by Microdrones[1]. They are electronically driven helicopters with four rotors, so called quadrocopters, providing a maximum flying weight of 6 kg. Having four rotors leads to a system which is controlled only by changes in rotational speed of each rotor. Every one of this rotors is driven by its own brushless engine. That makes the UAV almost completely maintenance free. In

[1] www.microdrones.com

comparison to a conventional helicopter design with a lot of moving parts like a swash plate, the possibility for technical failures is widely reduced.

The UAVs do have a high precision position stabilization and location estimation system. As in most localization systems a GPS receiver is used to obtain information about the absolute position. Together with measurements from accelerometers, gyroscopes, a magnetometer and a barometer the manufacturer provides the user a filtered position estimation. We extend the capabilities of the UAVs by adding a system on a chip running a specialized embedded Linux distribution that supports both a communication channel with the base station and other robots, as well as autonomous flight control. Our control modules utilizes the localization and attitude estimations from the UAV and generates signals that are fed back into the proprietary control software of the drone. The interface we use is the same, as the original radio control connection. This allows us to make use of all position stabilization functionalities provided by the manufacturer, as we electronically simulate a human operator. By the flick of a switch a real operator can obtain control over the UAVs at any given time for safety reasons.

Communication from and to the UAV is realized with two wireless connections. The original radio control device is connected over a bi-directional low-bandwidth, but high distance 2.4Ghz channel. It sends control commands to the UAV and receives information about the height, attitude and battery level, that are shown on a built-in LCD. The second channel is a 5Ghz Wireless LAN connection, used to transmit data with high bitrates.

Since this project is mainly focused on reconnaissance, the drones are equipped each with a 14.7 MP zoom camera. Since the UAVs have to alter their attitude to make changes in movement, a fixed mounting of the cameras would result in mostly blurry images. To resolve this, they are mounted within a moveable frame, which is deflected by two servos. The angle control input is taken from the attitude estimation made by the manufacturer. Pictures are accessible in two versions. One way is the live video preview, which is normally used on the camera display. These pictures do have a resolution of 320x240 pixels, an average file size of 9kB and are available at 15Hz. The second way is the single picture mode, where a high resolution picture is taken. In this case the pictures do have a resolution of up to 4416x3312 pixels and an average file size of 4MB.

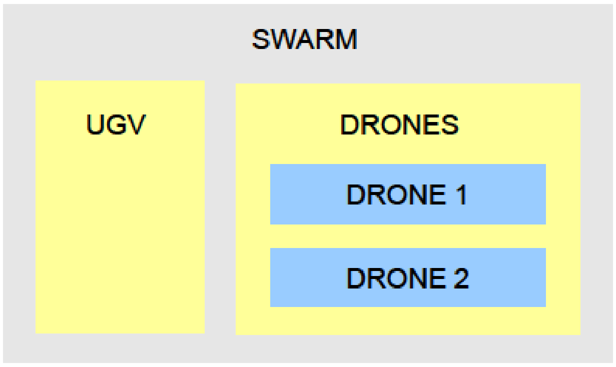

Fig. 2. Schematic hierarchy of the different robot groups, down to single robots that can be commanded

2.6 Scenarios

During the design process for the project we decided to formulate a scenario which is close to reality in terms of given propositions and usability. The basic task is to do reconnaissance in a roughly known area using a swarm of three heterogeneous, unmanned robots. In order to give the operator a clear overview of the environment, one UGV will traverse a known road map of the territory and deliver a occupancy grid map, while two UAVs provide image intelligence from above. To simplify the control process for the human operator the robots can be accessed separately, or in different groups (see Fig. 2). By standard, all orders are sent to the whole swarm only once. But if the current circumstances demand different behaviors, all orders can be given to every other entity within the hierarchy as well.

As mentioned above, BML is a rather high-level language, what makes it easy to read for machines but not very versatile in altering commands. We selected six commands from the pool and defined actions for the different types of robots to perform. Commands are described by so called BML orders. Each order has specific production rules defined by the C2GL rule set. Every one of them begins with the header, consisting of the *sender*, the *addressee* and the *sendingTime*. The header is followed by the order body (OB), which contains all information needed for the task.

All commands do share some common symbols. *Tasker* and *Taskee* are the names of entities that give, respectively receive orders. *At-Where* and *Route-Where* are spatial modifiers, describing a point, a list of points or an area. *Start-When* and *End-When* are temporal modifiers, and define when the command must be executed, or finished. And finally *Label* is set to a unique name for each given task.

1) Move: As the name implies this command moves robots from their current position to a specified point, which is the most basic ability the swarm has to be capable of. The rule for this order hast the following structure:

$$OB \rightarrow move\ Tasker\ Taskee\ At\text{-}Where\ Start\text{-}When\ (End\text{-}When)\ Label$$

If this command is send to the whole swarm it is processed by the ground vehicle in first place. It calculates a path from its current position to the target point, using the information from a previously known road map. While executing the order, the vehicle generates continuously (with a frequency of 1Hz) new *move* commands for the drone swarm, using its own position as the target point. All drones will then process these new messages and therefore follow the vehicle on its way to the destination in a formation depending on the number of UAVs.

2) Patrol: Sometimes it might be more interesting for the operator to have a look from different perspectives at the ground vehicle while it is moving. Therefore now the drones orbit around the ground vehicle, while it is traversing a number of target points. During this time the cameras of the drones are always headed towards the ground vehicle. The *patrol* command is structured as follows:

$$OB \rightarrow patrol\ Tasker\ Taskee\ Route\text{-}Where\ Start\text{-}When\ (End\text{-}When)\ Label$$

From the technical perspective the execution of this order is quite similar to *move*. The UGV traverses all the points given by the order using the information from the known road map. The UAVs now receive a patrol order from the UGV each second with its own position. With this information they calculate their position on an orbit around the UGV, using the absolute time from the GPS receiver and the knowledge of the number of drones in the swarm. Additionally the required angles for the camera are continuously updated using the position difference between each drone and the UGV.

3) Observe: This command lets the swarm approach a given point of interest. When this point is reached, all robots will deliver live video to the control station.

OB → observe Tasker Taskee At-Where Start-When (End-When) Label

If this command is given to the swarm, the ground vehicle internally uses the *move* command to guide the UAVs to the point of interest, so that the target is in the field of view of its camera. When approaching the destination the *observe* order is given once to the drones. They start to orbit around the target, by continuously generating patrol commands addressing themselves with a fixed target point.

4) Distribute: While the previous commands only took care of single points, or a route defined by points, the next ones describe actions taken on areas defined by polygons. Now the *At-where* in rule (4) does not consist of a single point, but of three or more, depending on the complexity of the polygon describing the area.

OB → distribute Tasker Taskee At-Where Start-When (End-When) Label

As the result of this command the swarm moves to the area of interest, lead by the UGV using *move* orders and then splits up. All robots distribute themselves equally within the area and send back live video to create an overview. The final positions for the robots are calculated by determining the center of gravity (CG) of polygons. The UGV uses the CG of the initial polygon and finds the closest point on a road within the map. When this point is reached the UAVs receive the *distribute* order, containing the polygon and split it up into smaller parts, depending on the number of drones. Then they apply the same algorithm as the UGV on this smaller polygons to determine their final positions for this scenario.

5) Guard: In comparison to the Distribute command, Guard is more challenging from the drones point of view. The ground vehicle executes the same behavior as during the Distribute scenario, except that it sends a single Guard command to the drone swarm, when it reaches the target area. The UAVs split up the given area into smaller parts, depending on the number of drones, and then traverse this smaller areas continuously while delivering high-resolution pictures to the control station.

OB → guard Tasker Taskee At-Where Start-When (End-When) Label

The waypoints needed for the UAVs to traverse the area are generated by overlaying the polygon with lines that are parallel to the line that divides the original polygon into smaller ones. The intersection points between this lines and the polygon boundaries are used for navigation, which leads to a saw tooth pattern of movement.

6) Reconnaissance: This is the most complex command for the swarm. The ground vehicle traverses all known roads within the given target area and builds up an occupancy grid of the environment while the drones deliver images from above.

OB → reconnaissance Tasker Taskee At-Where Start-When (End-When) Label

The ground vehicle guides the drone swarm to the target area using the move command. When the area is reached, one UAV is given the reconnaissance command, which lets it traverse all corner points of the polygon and deliver high resolution pictures from there. The second UAV is continuously given the patrol command to follow the ground vehicle as described above.

3 UAV Hard-/Software Modules

Besides the software implemented by the manufacturer, that takes care of stabilization and localization of the UAV, we added an additional system which executes supplementary tasks and generates the control signals for flight maneuvers, like the ones described in the last chapter. This system consists of a ARM9 system on a chip (SOC) with a total of 128MB of RAM and several interfaces like USB 2.0 and three serial connections. It is realized as a densely packed circuit measuring 54x54mm in size and 30g in weight.

3.1 Psyche Embedded Linux

As the operating system for this SOC we use a custom made Linux distribution. The kernel is highly adapted to the used hardware and the included libraries are reduced to a minimum to save as much resources as possible for user applications.

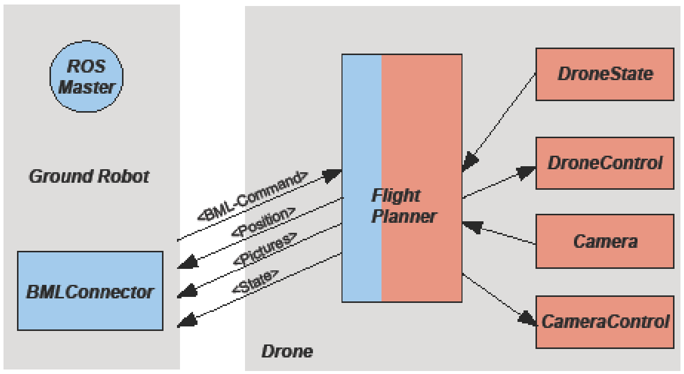

Fig. 3. ROS integration into the system. Blue parts show ROS modules. Red parts show proprietary UAV modules.

3.2 Psyche Software Environment

The main proprietary software modules running on the UAV are sketched with red color in Fig. 3. A more detailed information about the architecture is given in [9], but core functionalities should be mentioned shortly:

1) DroneState: This module makes all data coming from the drone available to a client via a UDP connection. Examples for types of data are position and attitude informations, battery status, wind-speed, etc.

2) DroneControl: Through this module users send orders to the UAV. Those are basically waypoints, or lists of waypoints attributed with a desired velocity and attitude.

3) Camera: Users are able to access the pictures shot with the USB attached camera through a TCP connection by using this module. Both, the low- and high-resolution pictures mentioned above are transported to the clients.

4) CameraControl: This module allows the user to remotely adjust all parameters of the camera. Typical use cases are zoom, exposure time, desired resolution and also the tilt angle of the moveable camera frame.

3.3 Reactive Collision Avoidance

Although all robots in the swarm are commanded from the control station, there is no central path planning system for the UAVs. The flying formations are chosen carefully to avoid small gaps between the drones, but while formation changes distances may fall below the safety limit. Therefore each drone is equipped with a software module, that realizes a reactive collision avoidance.

Due to a lack of sensors that are able to detect other UAVs during flights, the drones permanently publish data about their positions and movements via a UDP broadcast connection. This communication is established via an ad-hoc wireless network. WLAN is chosen, because it is very cheap and easy to implement into the system and it is stable and reliable when used for short distance communication, which is always given, if the threat of collision is present.

3.4 ROS Interface

ROS is used as the communication layer within this project, so every robot has to have the ability to publish and receive messages. Therefore the software parts, which implement the algorithms for task execution, also build a bridge between ROS and the proprietary interfaces on the robots. This part is called *Flight planner* in the case of the UAVs and its interfaces are shown in Fig. 3. The implementation on the UGV is mostly identical and not separately described here. The drones receive commands from an operator or from the ground vehicle in form of BML orders, embedded in ROS messages. The messages are processed and depending on the current scenario, different kinds of data are sent back to the control station. In this stage of the project such data mainly consists of reports of the robots positions (*geometry msgs/Pose*), pictures or videos (*sensor msgs/CompressedImage*) from interesting points and occupancy grids (*nav msgs/OccupancyGrid*) for operator defined areas.

3.5 Scenario Interpretation

As mentioned above all algorithms needed for the execution of the different scenarios are implemented within the *Flight planner* module onboard of the UAVs and accordingly on the UGV. The Planner is realized as a state machine with transitions triggered by incoming new, or completions of existing orders. If a new order arrives the first thing to check is the *Taskee* entry of the BML message. The different entities, that can be used as tasks are shown in Fig. 2. If the message must be processed on the given robot the next barrier is the level of the *Tasker*. If this level is higher, or on the same level as the currently active one, the message is further handled. Otherwise it is dismissed.

If the order has to be executed, the robots will start a behavior as described in 2.5. Thus the state machines makes a transition into the appropriate state and the level for the current tasker is set, for further comparisons. If the order is fully processed, before a new one arrives the tasker level is reset to the lowest one possible, so that orders from each entity are processed again.

4 Results

This project shows, that it is possible to control a heterogeneous robot swarm by a very abstract, but formal and context-free language. Using an abstract language requires a certain degree of autonomy on the robots, depending on the degree of abstractness. If this autonomy is available, a single operator can take care of many cooperating systems, without getting overstrained by caring too much about single robots. All scenarios described above were successfully demonstrated at a designated test ground with one ground robot and two aerial robots. All tasks were assigned to the whole robot swarm by a single operator using a simple and intuitive graphical user interface. Mission times, longer than 30 minutes without manual intervention or malfunctions of the systems were effectively shown.

But since the communication channel is strongly constrained by using a formal language, there are some information that cannot be transferred. Rather simple values like desired flying height, or an attitude cannot be expressed in BML. Even more problematic is the task of transmitting sensor data. To overcome this problem, either the language has to be extended to be able to handle binary data. Or the robot swarm has to be more intelligent. In this case sensor data has to be preprocessed on the individual robot and only the higher level interpretation of the information is transferred in the form of BML reports.

References

1. Kuhnert, L., Ax, M., Langer, M., Nguyen Van, D., Kuhnert, K.-D.: Absolute high-precision localisation of an unmanned ground vehicle by using real-time aerial video imagery for georeferenced orthophoto registration. In: Dillmann, R., Beyerer, J., Stiller, C., Zoellner, J.M., Gindele, T. (eds.) Autonome Mobile Systeme 2009. Informatik aktuell, pp. 145–152. Springer, Heidelberg (2009)

2. Phan, C., Liu, H.H.T.: A cooperative UAV/UGV platform for wildfire detection and fighting. In: Asia Simulation Conference 7th International Conference on System Simulation and Scientific Computing, ICSC 2008, October 10-12, pp. 494–498 (2008)
3. Tanner, H.G.: Switched UAV-UGV Cooperation Scheme for Target Detection. In: 2007 IEEE International Conference on Robotics and Automation, April 10-14, pp. 3457–3462 (2007)
4. Hsieh, M.A., Cowley, A., Keller, J.F., Chaimowicz, L., Grocholsky, B., Kumar, V., Taylor, C.J., Endo, Y., Arkin, R.C., Jung, B., Wolf, D.F., Sukhatme, G.S., MacKenzie, D.C.: Adaptive teams of autonomous aerial and ground robots for situational awareness. Field Reports. J. Field Robot 24, 991–1014, November 11-12 (2007)
5. Schade, U., Hieb, M.R., Frey, M., Rein, K.: Command and control lexical grammar (c2gl) specification. Fraunhofer FKIE, Tech. Rep (2010)
6. Quigley, M., Gerkey, B., Conley, K., Faus, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
7. Langerwisch, M., Reimer, M., Hentschel, M., Wagner, B.: Control of a semi-autonomous ugv using lossy low-bandwidth communication. In: The Second IFAC Symposium on Te-lematics Applications (TA), Timioara, Romania (October 2010)
8. Hentschel, M., Wagner, B.: Autonomous robot navigation based on openstreetmap geodata. In: 13th International IEEE Conference on Intelligent Transportation Systems (ITSC) (September 2010)
9. Ax, M., Kuhnert, L., Langer, M., Schlemper, J., Kuhnert, K.-D.: Architecture of an autonomous mini unmanned aerial vehicle based on a commercial platform. In: Joint 41th Symposium on Robotics and 6th German Conference on Robotics (June 2010)