SAUC-E 2011 - The Hanse Team

Jan Hartmann, Dariush Forouher, Marek Litza, Helge Klüssendorff, Benjamin Meyer, Tjorven Mintzlaff, Patrik Stahl, Christian Strakerjahn, Thomas Tosik, Patrick Zenker and Erik Maehle

***Abstract*—This paper describes the University of Lübeck's entry to the 2011 SAUC-E competition. Our AUV HANSE has been specifically designed for the SAUC-E competition by students in practical courses and master thesis' throughout the last three years. Besides the thrusters, sonars, and the orientation sensor, it is built of commercially available commodity products.**

As we are joining the SAUC-E for the third time since 2009, in this year the focus for innovation in our project has shifted from hardware to software. We have further greatly improved the reliability of the robot. This paper will describe our new localization, navigation, and object recognition algorithms as well as the improvements to the hardware reliability and test procedures, including our new simulator.

Fig. 1. The Hanse AUV.

I. INTRODUCTION

The main goal of the Hanse project is to encourage our student's interest in robotics in general and the challenges of underwater robotics specifically - and of course to build a robot which provides the features of robustness and expandability necessary for a project designed to run for several years. Our notion is that this can be best achieved in a competitive environment.

The AUV Hanse has thus been developed specifically for the SAUC-E in a series of practical courses (bachelor as well as master) and several master thesis' by students over the last three years. Hanse's main housing, a waterproof Peli case, is mounted to a Polypropylene (PP) base frame. A number of Buccaneer connectors provide a generic interface from the controlling laptop to thrusters and sensors, which are also mounted to the base frame. This ensures the expandability of the design concerning sensors as well as control hardware.

The robot control software runs on a 12.1" standard laptop. A modular software framework was developed in 2010 and greatly expanded in 2011. It is based on the multi-platform C++ libraries Qt[1] as a general API and OpenCV[2] for image processing. It runs on Windows as well as Linux, while the latter one is preferred because of the better control over the operating system.

The Hanse robot won the innovation prize in the SAUC-E 2009, where the focus was on building the robot itself and the handmade thrusters. In 2010, the latter were replaced by the more robust SeaBotix thrusters and a scanning sonar was added. The new software framework was developed and due to changes in the mission rules, new algorithms were developed. In retrospective, because of the vast amount of software modules the interaction of software and hardware could not be tested sufficiently before the competition and the result was not as good as hoped.

This year, we tackled the problems of 2010, mainly the stability of hardware and software and the lack of real testing. Few changes have therefore been applied to the robot, including a new Absolute Heading Reference System (AHRS) and an additional single beam sonar. The effort was focused on software development and therein on navigation and task planning algorithms. A simulator - including 3D visualization, underwater physics, visual and sonar sensors - was developed to ease the testing procedure.

The remainder of this paper will be structured as follows. First, the robot hardware will be discussed in more detail in Section II. In Section III the software architecture and the main algorithms will be presented. A financial summary and risk assessment will be given in Sections IV and V. Then hard- and software will be evaluated in comparison to the last two years to give conclusion on this year's project innovation. Finally, the team members and their responsibilities within the team will be introduced.

II. HARDWARE

A. Overview and mechanical actuators

The base frame of our AUVs owes the form to a sledge. This form was chosen because the thrusters can be attached at any position on the sides of the scaffolding. Thus the position of each thruster can be evaluated during the test runs, and can be mounted to its optimal fixing point. Another advantage of this form is that the AUV can be carried conveniently by

[1] http://qt.nokia.com

[2] http://opencv.willowgarage.com

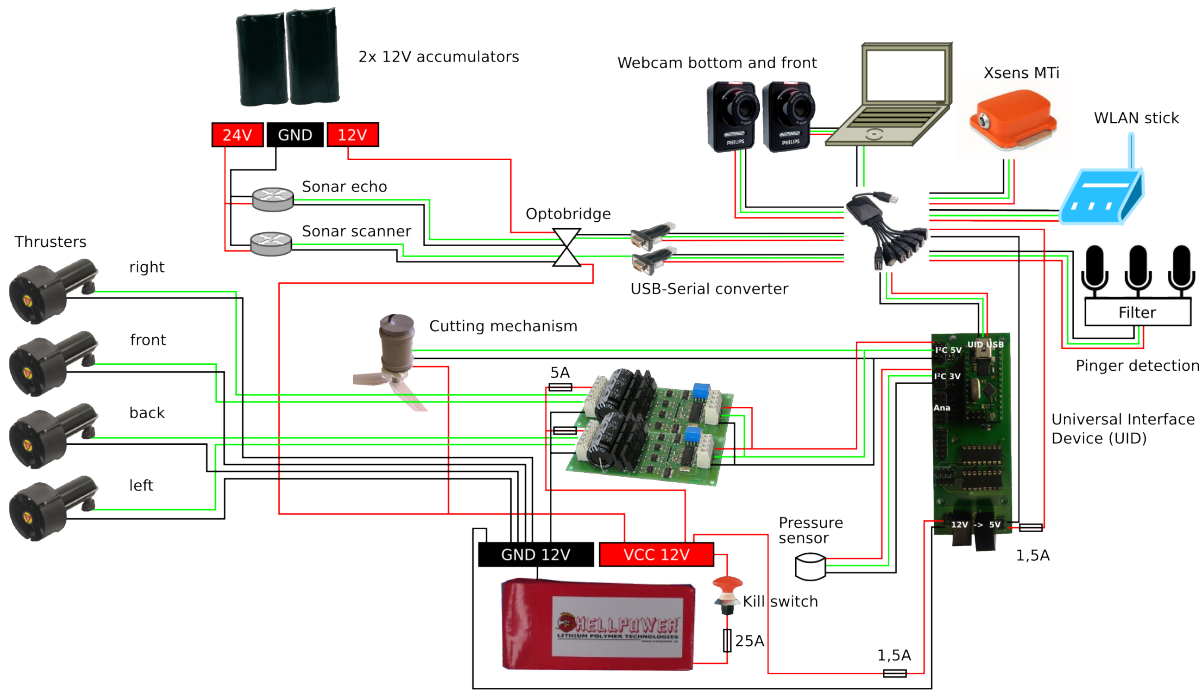


Fig. 2. Hardware and electrical overview of the Hanse AUV. Power supply in black/red, signals in green.

two people. The base frame of the AUV is made out of 50 mm Polypropylene (PP) tubes. We have chosen this material because of its light weight and the possibility of welding single parts together easily. To increase the solidity of the frame it was strengthened by glass fiber sheathing. Additional holes, which are drilled in the frame in distances of 10 cm apart, allow the flooding of the frame, so it has near neutral buoyancy.

On the base frame a waterproof case (Pelicase 1400) is fixed, which is the main pressure hull of the robot. An internal frame, made of steel and wood, makes the relatively soft case inherently stable against expected amounts of pressure. With its inside measurements of 30x22.5x13.2cm the case offers enough space for most electronic parts and power supply.

Several modules like thrusters and sensors are placed outside the main case. Their exact mixture and position on the frame can be adapted to the actual mission. All modules are attached by waterproofed BULGIN Buccaneer (PX0748/P) connectors to the main case. To ensure stable horizontal orientation of the robot, external weights can be attached on to the frame. The Hanse AUV in the current version can be seen in Fig. 1.

1) Motors: For propulsion four SeaBotix BTD150 thrusters are attached on the main frame. Two are placed on the sides for horizontal movement and rotation around the yaw axis. The other pair is placed in front and back of the main body and allows diving. Because of positive buoyancy the thrusters must work to submerge.

2) Camera housing: As camera housing for our webcams we use a lamp housing that is usually used for illuminating garden ponds. These cases have a 5 mm glass panel, and are waterproof up to 10 m.

3) Cutting mechanism: This year we build an active cutting device. It consists of V-formed guiding arms and a small knife, moved by a servo. The servo is placed inside a waterproof container together with a SD20 servo-driver (Devantech). The module is connected to the main system by I2C-Bus.

B. Electronic design

The general electronic design can be seen in Fig. 2. Each component will be introduced in detail in the following sections.

1) Power Supply: The Hanse power supply consists of three electrical circuits. The first contains the notebook, the central processor unit, and one webcam connected over USB. The second circuit is providing power for the sonars. It contains two small serial connected lithium-polymer-accumulators with 740mAh and 11.1V each.

The third, the main power circuit supplies all remaining modules. It is powered by a three-cell high performance lithium-polymer-accumulator providing 10 Ah at 11.1 V. The battery is secured by a fuse before its first connector. This circuit can be disconnected by the kill switch, located on the top of the case. Pressing it interrupts the electricity supply of the engines immediately, and the AUV emerge.

2) Main Computer: Similar to the last year we use the ACER “AspireTimeline1810TSpecial Edition” notebook as onboard computer. It contains an Intel(R)Core 2 Duo processor SU7300 (1, 2 GHz), and

500 GB hard disc. With a width of 285 mm we have around 5mm space between the notebook and the case at the left and right side. This demands careful space management inside the case.

3) Xsens MTi/AHRS Sensor: We added the Xsens MTi/AHRS Sensor to the system this year. It replaces the compass and our old IMU-Unit. The Xsens MTi is connected to the system by USB, which allows us to read the sensor at 1 Mbps. It provides the system with reliable attitude and heading informations, that is used inside the navigation and localization algorithms and to correct sonar images.

4) Pressure Sensor: To measure the actual depth, we use a 'MS5541-CM' pressure sensors from Intersema. It has an absolute pressure range from 0 to 14 bar put out as 16 Bit value and achieves an accuracy of 2 cm. It is connected over a self built SPI-to-I2C translator that is located in the housing of the pressure sensor. This translator additionally computes the temperature compensated pressure data as described in the datasheet of the sensor and gives the result to the I2C-Master.

5) Sonar Modules: We use the "Model 852 ultra-miniature scanning sonar" from Imagenex for localization. This sonar has a beam width of 2.5 degrees x 22 degrees. Adjusting the gain, ranges from 150 mm up to 50 m are reachable. Additional it has two step sizes: normal (3 degrees) and fast (6 degrees). With a maximum range of 50 m one rotation requires 16 seconds in case of the normal mode and eight seconds in case of the fast mode. The sonar can work with 675 or 850 kHz. We are using 850 kHz to minimize the cross noise with the second sonar.

The second sonar module, the "Model 852 ultra-miniature echo sounder", uses 675kHz as working frequency. It has a conical beam of 10 degrees width, and range scales up to 50 m. We use it primary for wall detection, so it is oriented to the port side.

Both sonar modules are connected over an RS232 serial interface. To avoid noise from the Motors we shield the sonar modules by using an "Expert Opto-Bridge" optical coupler module (Gude) for communication and a separated power supply.

6) Cameras: We use two USB webcams 'SPC1030' from Philips. The first is facing forward primarily for ball detection. The second camera is mounted facing downwards for pipe detection. The cameras grab 640x480 pixel images with a frame rate of 5 Hz. The "SPC1030" webcam has a lens view angle of 80 degrees that is decreased by the water to 60 degrees.

7) Pinger detection: In order to detect the 12kHz pinger three hand-made hydrophones are mounted onto the 'sledge'. The first step of signal processing contains a set of analogue filters which form a bandpass with constant amplification rate. In the second step a simple adjustable amplifier is located, to adept the required amplification to identify the pinger. The processing and orientation computing is done by an ATmega microcontroller, that estimates the direction by analyzing the different times of arrival (TOA) of the three hydrophones.

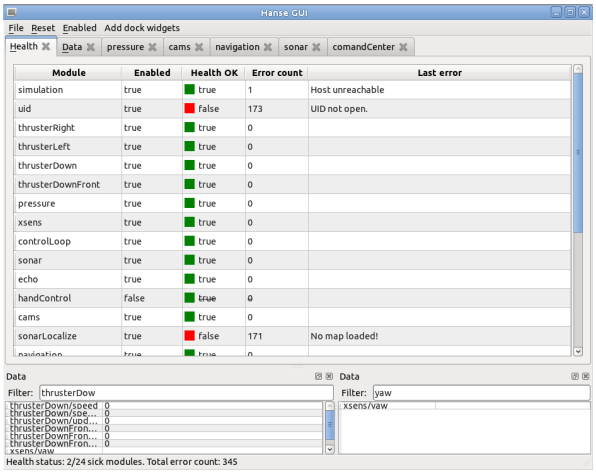


Fig. 3. Snapshot of the software in action.

8) Bus Network & Universal Interface Device: The communication interface between external modules and the notebook is done by our self-built "Universal Interface Device" (UID). As hardware platform we use a small ATmega168-Board from chip45, but the UID architecture is not determined to this board, it can be used for almost any type of Atmel 8-bit processors. The UID is connected by USB and is addressed by a serial interface with a configurable speed from 2400 bps up to 2 Mbps. The standard communication speed is set to 115200 baud in order to allow the using of a normal terminal program to communicate with the UID. To buffer the incoming and outgoing serial data, a 256 Byte ring buffer both for receive and transmit unit is implemented. Beside the I2C and SPI communication, additional features like GPIOs, 8 ADC channels, a small servo-controller for up to three servo motors as well as RS485 Transceiver are implemented.

III. SOFTWARE

We developed a software framework specifically written for Hanse and the SAUC-E competition. The design goals were:

- The framework must run under Microsoft Windows.
- The framework must be modular.
- It should have a graphical control and visualization interface.
- It must be written in C/C++.

The first requirement was due to the fact that Windows was the robot's operating system at the time of the specification. We have switched to Linux since then.

We settled on implementing the framework using Qt 4, a cross-platform application framework, written in C++. Qt provides an easy-to-use GUI toolkit, which

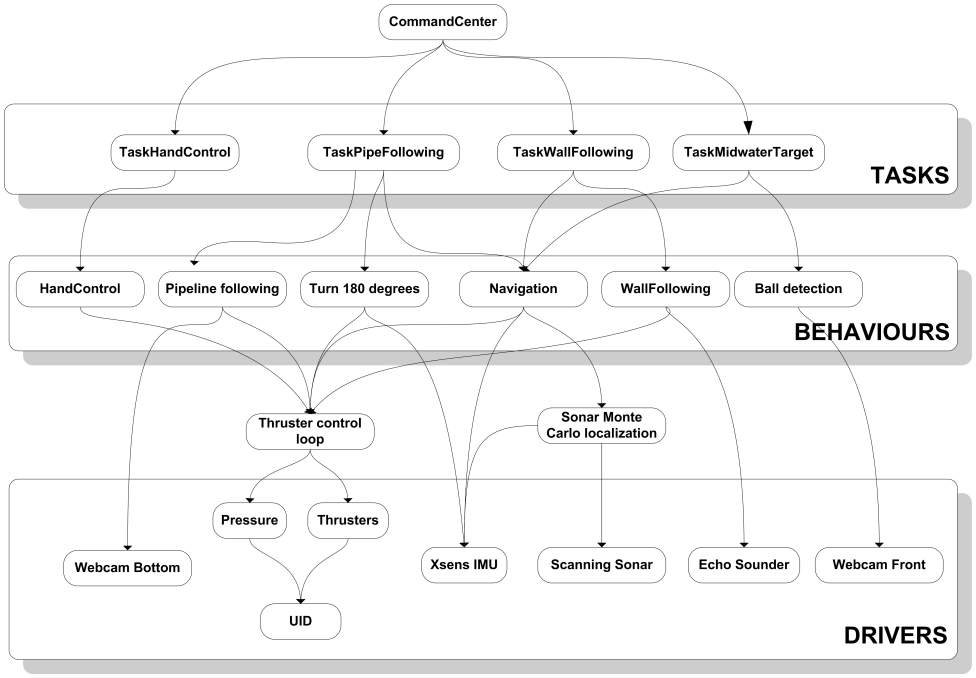


Fig. 4. Modules and their dependencies. Task use several behaviors to perform a SAUC-E mission. Behaviors adjust motor speeds based on sensor and localization information.

we used to display runtime information on the state of the robot. It also provides platform independent access to many important functions like file access, threads, locking and dynamic memory management. One of the most interesting aspects of Qt is its Signal/Slot implementation of the *Observer pattern*, which we used heavily.

The software part of the Hanse project will be described in more detail in the following sections. First, the general architecture will be shown. Then some of the more interesting algorithms will be highlighted, including image processing, localization, navigation, and behavioral algorithms. Finally, the simulator will described.

A. Architecture

The basic architecture of our framework is shown in Fig. 4. It consists of a slim core, which includes the raw graphical user interface as well as a couple of library classes. The core provides an message-based logging interface (using Log4Qt[3]), a serial port interface (using qextserialport[4]) and some code for calculating and plotting control loops (an implementation of a PID control loop). It also provides a configuration interface (using .ini files) and a data logging library, which logs accumulating (numerical) data into CSV files.

The rest of the framework consists of modules, each implementing a C++ base class *RobotModule*. At the time of this writing roughly two dozen modules have been implemented, ranging from low-level hardware drivers to controlling behaviors. All modules are organized in a tree, with each module depending on one or more other modules.

Each RobotModule has its own thread attached to it. All *Qt Slots* are executed within a modules own thread. To mitigate the difficulties inherent in multi-threading, some design decisions have been made to reduce the dangers of deadlocks and unprotected access to common data structures. The first design decision was to ensure that the graph, which the modules span, is a directed acyclic graph (DAG) and thus contains no loops. This avoids circular data dependencies between modules. Another important side-effect of this is to keep the overall complexity on a bearable level. Even a modular software framework can become unmaintainable if too many dependencies crop up between modules over time.

The second and more important design decision to avoid threading problems was to heavily use the Signal/Slot infrastructure of Qt. Signals/Slots allow objects in different threads to communicate asynchronously, e.g. if a sensor receives new data, it will emit a Signal containing that data. If another module has connected a Slot to this Signal, that Slot will be called once the Signal is emitted.

B. Image Processing

The two cameras equipped on Hanse are used in two tasks: the inspection of the pipeline and the freeing of the mid-water target. As the webcams that are used have a low image quality, we decided to use a color-based object recognition approach rather than edge or

[3]http://log4qt.sourceforge.net/

[4]https://code.google.com/p/qextserialport/

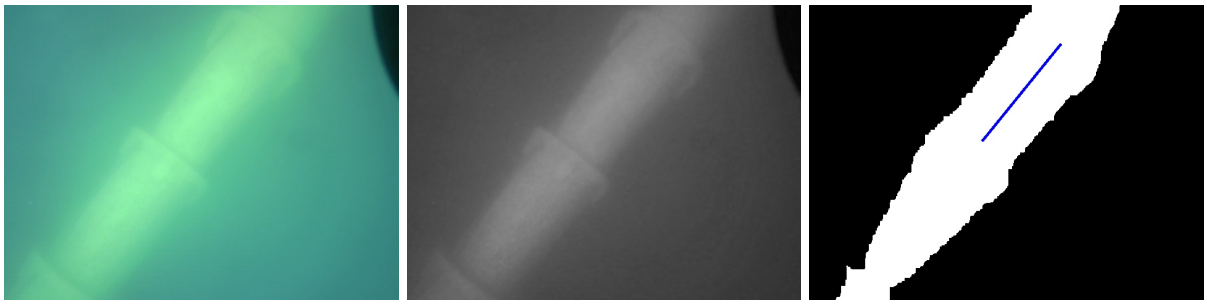


Fig. 5. Finding the location of the pipe relative to the robot. Left: original rgb image taken at the SAUC-E 2010. Center: segmentation channel (in this case blue channel). Right: binary image after applying the Otsu thresholding algorithm. The line in the middle of the white patch indicates the extracted position and direction of the pipe, which is obtained using 2D moments.

corner-based feature extraction techniques. Each image is therefore segmented in the appropriate color channel (blue for the pipe and saturation for the mid-water target). A good segmentation threshold is found using Otsu's algorithm, which tries to minimize inner class variance and maximize outer class variance [1]. This provides some degree of invariance to lighting.

The presence of either the pipe or the mid-water target is decided based on the number of pixels belonging to the object class and the mean position of these pixels. The pipe orientation can further be found using centralized image moments [2] on the segmented image S, where

$$S(x,y) = \begin{cases} 1 & \text{, if the pixel (x,y) belongs to the object} \\ 0 & \text{, else} \end{cases} \tag{1}$$

The centralized moment of order $p+q$ is then defined as

$$\mu_{pq} = \sum_x \sum_y (x-\overline{x})^p (y-\overline{y})^q S(x,y), \tag{2}$$

and the orientation of the object can be found as

$$\theta = \frac{1}{2}\text{atan2}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right). \tag{3}$$

An example of the image processing algorithm can be found in Fig. 5. Image processing for the mid-water target closely follows the one presented for the pipe.

C. Localization & Navigation

This section will describe the localization and navigation technique in detail. The former can be subdivided into two main parts: the sonar image feature extraction and the localization algorithm itself. The latter will be presented in form of a state chart.

1) Feature Extraction: The task of the sonar image feature extraction is the identification of prominent parts of a sonar image, i.e. a full 360°sonar scan. We specifically search for wall-like features, as they are easy to identify and present in most man-made environments. Such features have a characteristic signature in a sonar image: they show a strong echo at and before the location of the wall and a significant drop in intensity behind the wall.

The filter chain described here is based on the one presented in [3] with further improvements and simplifications. We designed a multi-scale gradient filter to enhance the regions of the sonar image characteristic to the wall features. 1D Haar wavelet responses at different scales are multiplied for each beam pixel to form the beam gradient G as

$$G(x) = \prod_{k\in K}\left(\sum_{i=x-k}^{x} B(i) - \sum_{i=x+1}^{x+k} B(i)\right), \tag{4}$$

where K is the set of all scales to be evaluated and $B(i)$ is the echo intensity at distance i. The gradient filter is illustrated in Fig. 6(a).

The Haar wavelet responses can be efficiently calculated using integral images. The integral image is in this case defined as the sum of intensities up to the current distance:

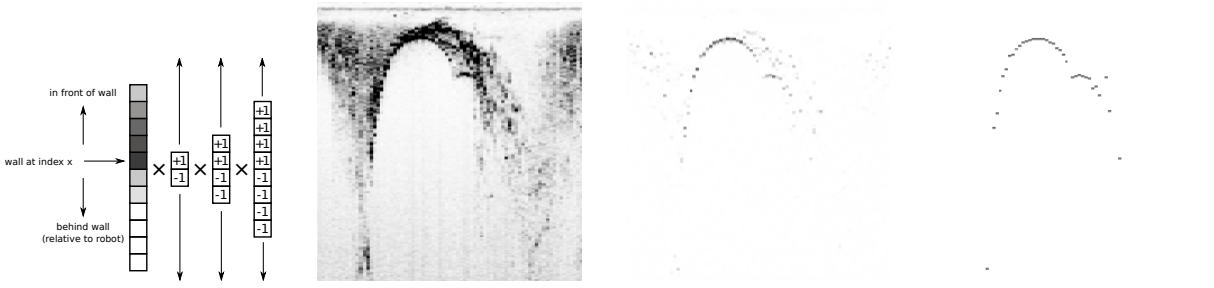
$$I(x) = \sum_{i=0}^{x} B(i). \tag{5}$$

The gradient filter response can now be calculated as:

$$G(x) = \prod_{k\in K}(2I(x) - I(x-k) - I(x+k)). \tag{6}$$

2) Heuristics: The filtered sonar image is now analyzed to find wall locations. First, a non-maximum suppression is performed to identify potential walls. For each beam, those local maxima are discarded that have a gradient intensity of less than one tenth of the maximum gradient intensity in the last sonar image. Of the remaining maxima, for each beam, all but the maximum furthest from the robot are discarded, as we assume walls to absorb the whole sonar beam.

Now, further heuristics concerning the neighbors of wall features are applied. First, walls are assumed to be continuous. Assume the distance of the wall feature of beam j to the robot is w_j. Then the location of that wall



(a) Close-up of a wall feature and the 1D Haar wavelets to be applied.

(b) A 360°scan as provided by the scanning sonar.

(c) Sonar image after applying the gradient filter.

(d) extracted wall features after applying non-maximum suppression and heuristics.

Fig. 6. Illustration of the sonar image filter chain, including the gradient filter and heuristics.

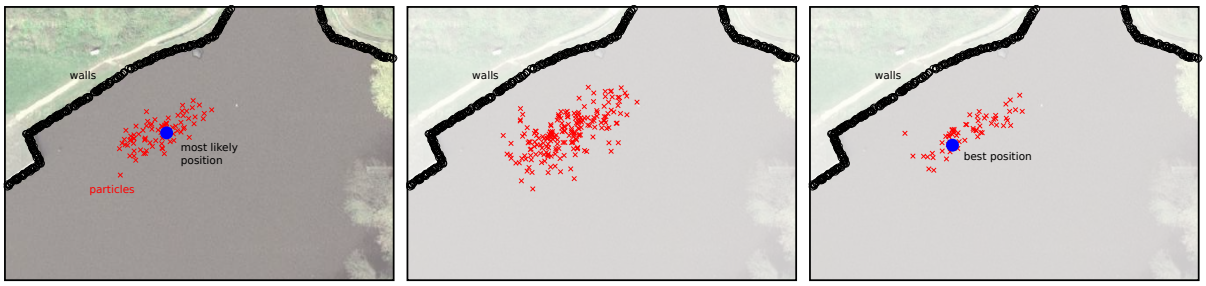


Fig. 7. Illustration of a particle filter at work. Left: The set particles (red) is the estimation of the robots position. The most likely position is assumed to be the centroid of the cloud. Middle: After predicting the movement of the robot using the AHRS and a Gaussian motion model, the variance of the particle cloud increases. Right: The particles are weighted according to how well the observations match the map, and then resampled to weed out the bad particles.

feature is only correct if the distances of the preceding and following wall features do not differ too much, or

$$|(w_j - w_{j-1}) - (w_{j+1} - w_j)| < T_{cont}, \qquad (7)$$

where T_{cont} is a fixed threshold.

Secondly, walls are assumed to have some lengths. Therefore, if the number of wall features in a window of size k surrounding a certain wall feature is less than $\frac{k-1}{2}$, then this wall feature is assumed to stem from image noise rather than real walls. Such features are therefore also discarded. The remaining wall features are finally transformed to a 2D position relative to the robot, depending on the corresponding sonar head position and AHRS heading. The transformed wall features are then passed to the localization algorithm.

The result of the filter chain can is illustrated in Fig. 6(b) to 6(d), which show the original sonar image, the result of the gradient filter, and the extracted wall features after applying the heuristics.

3) Localization: The filter chain finishes with a set of observations - a point cloud with positions relative to the robot. We localize our robot using these observations and an a priori map, which has been created beforehand. The position of the robot is estimated using a particle filter (which is an implementation of the recursive Bayes filter, [4]). A particle filter consists of a set of particles, each one a hypothesis of where the robot might be (see Fig. 7). Once a new observation has passed through the filter chain, the movement of the robot since the last particle filter update is estimated (using a Gaussian motion model and data from the XSens AHRS). This results in a *spreading* of the particles, as the uncertainty of where the robot might be has increased.

Now for each particle a confidence value is calculated, which denotes how well the observation would match the map if the robot were in this particle's position. Then the particles are weighted with this confidence value and normalized so that the sum of all weighted particles is exactly one. Bad particles (which don't represent the robot's true position) thus get a small weight whereas good particles receive a large weight.

Finally the particle set is resampled. This is called *importance sampling* and is the most important step of a particle filter. Each particle's importance weight equals the likelihood that this particle represents the true position of the robot. During resampling a new set of particles is chosen based on the old set. The new set will have the same size as the old set. However bad particles with a low importance weight will likely be missing in the new set. Particles with a high importance weights on the other hand will be represented with multiple copies.

After the resampling, the new set of particles is an approximation of the Bayes posterior representing the position estimate of the robot.

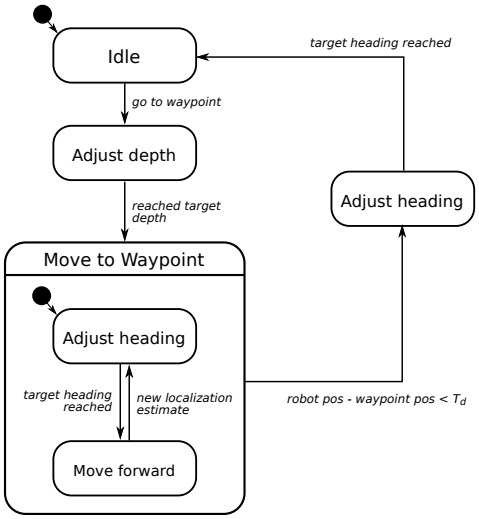


Fig. 8. State chart for the navigation. To reach a waypoint, the robot first moves to the target depth. Then the heading towards the goal is adjusted and the robot moves forward until a new location is estimated (approximately every 8 seconds). When a waypoint is reached, the robot is turned to the target heading.

4) Navigation: The navigation capabilities of Hanse are limited by the time difference between two location estimates, which is defined by the time it takes the scanning sonar to rotate 360°. Therefore, between two estimates, there are approximately 8 seconds in which only changes in orientation (through the AHRS) and no other movement information is known. This needs to be taken into account by the navigation algorithm.

The navigation algorithm is designed in a "point and shoot" manner, keeping it as simple as possible. Fig. 8 describes the algorithm as a hierarchical state machine. When a new waypoint is set, first, the robot is moved to the target depth. Then, until the target position is reached, iteratively, the heading towards the goal is adjusted and the robot moves forward until a new location is estimated. The adjustment of the heading is therein controlled by the AHRS and the forward speed is set relative to the distance between the robot and goal.

D. Planning and Task Structure

In our framework, a *task* encapsulates everything that belongs to one single SAUC-E mission. A task is a concatenation of several *behaviors*. A behavior is an elemental portion of of a mission with a define entry and exit state. A task will start a set of behaviors in the defined order. If a behavior enters its exit state, the next behavior is started.

A task performing the pipe inspection mission could thus consist of a list of the following behaviors:

1) move to waypoint "pipe center"
2) do pipe following until at waypoint "pipe end 1"
3) perform 180°turn
4) do pipe following until at waypoint "pipe end 2"

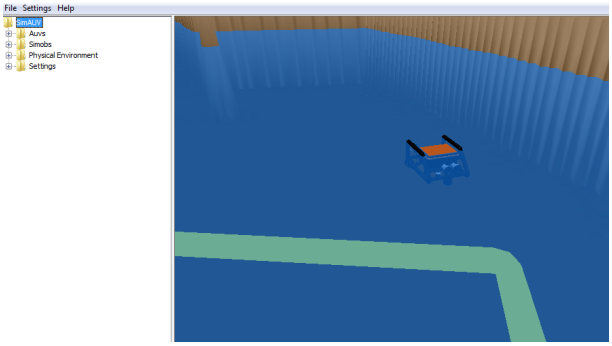


Fig. 9. Snapshot of the simulation.

Each of these behaviors is implemented as a subclass of RobotModule. They usually consist of a simple state machine and a controller for angular and forward speed. The input of the controller may be raw sensor measurements (e.g. for the 180°turn), data from the image processing algorithms (e.g. for the pipe following), or localization data (e.g. for determining the position relative to the pipe).

E. Simulation

This year we used a physical simulation of Hanse to perform thorough dry-dock testing of all behaviors and higher level code in Hanse. The simulator we used was developed [5] by Thomas Tosik[**?**], specifically for this purpose. It is based on the bullet physics engine and the JMonkey 3D engine and models the behavior of our AUV in an underwater environment sufficiently accurate. Fig 9 shows a snapshot of the simulator. The simulator runs independently to the software framework of the robot and is connected to it via a TCP connection.
The simulator supports multiple AUVs. Special attention was laid on the correct physics and the sensor systems, especially the cameras and the sonar. The simulator graphics allow for more natural looking underwater environments through the use of various filters like underwater-fog and depth-of-field.

IV. FINANCIAL SUMMARY

A table of the total expenses for the Hanse AUV can be found in Tab. I on the last page, excluding traveling costs for the competitions. It lists the expenses of the last two SAUC-E combined and the expenses for the SAUC-E 2011. Total new expenses are at 4,484 Euro, where the new AHRS and echo sounder account for the 3,700 Euro. Total expenses are now at 13,843 Euro for the last three years.

V. RISK ASSESSMENT

Potential risks and precautions taken are listed in Tab. II on the last page. In comparison to SAUC-E 2010, additional precautions have been taken considering communication and software errors.

VI. CONCLUSION

This section will provide some final conclusions on the state of the Hanse project as well as an evaluation of the work done between the last and the current SAUC-E. Sections II and III showed the current state of the hardware and software. We described some of the most important new algorithms in the fields of image processing, navigation, and planning in detail.

In terms of hardware, few additions were made, so that the existing hardware could be better tested. As of now, we were therefore able to put more effort into elaborately testing old and new parts of the software, while in the last years the focus for tests was much more on new hardware parts. The simulator further helped the testing procedure, as new algorithms in navigation and task planning could first be tested in an ideal environment to wield out most of the conceptual errors.

New hardware was integrated to help in those missions that we were not able to complete in SAUC-E 2010. The XSens MTi AHRS is used to improve the localization performance, so that we can navigate between different missions. The echo sounder is used to perform the wall following while tracking the AUV position using the scanning sonar. The new pinger filter chain will allow us to follow the ASV.

Specific new or improved algorithms are shown in Sec. III-B to III-D. While in the last years, most higher level parts of the software were first tested in the competition itself, due to the simulator and the more robust hardware, we will likely be able to perform sufficient testing for all parts of the software before the competition. We will then, for the first time, be able to tackle all but the last mission (for which we would need an acoustic modem).

REFERENCES

[1] N. Otsu, "A Threshold Selection Method from Gray-level Histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[2] S. O. Belkasim, M. Shridhar, and M. Ahmadi, "Pattern recognition with moment invariants: A comparative study and new results," *Pattern Recognition*, vol. 24, no. 12, pp. 1117–1138, 1991.

[3] D. Forouher, J. Hartmann, M. Litza, and E. Maehle, "Sonar-based fastslam in an underwater environment using walls as features," *ICAR 2011*, 2011.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[5] T. Tosik, "Physikalisch realitätsnahe simulationsumgebung für das auv hanse, mit integration in die bestehende steuerungssoftware," Master's thesis, University of Lübeck, 2011.

VII. HANSE TEAM MEMBERS

Dariush Forouher Project team leader. Dariush Forouher is working as a Ph.D student at the Institute of Computer Engineering, University of Lübeck. He is a member of the Hanse team since its foundation and is involved in underwater robotics since 2007. His interests lie in navigation and software architecture. His responsibilities are the general software framework as well as low-level drivers.

Jan Hartmann Project team leader. Jan Hartmann is working as a Ph.D student at the Institute of Computer Engineering, University of Lübeck. He is also a member of the Hanse team since its foundation. His interests lie in image processing and machine learning. His responsibilities are all things related to image processing. He is also overseeing the development of higher level behaviors and navigation.

Helge Klüssendorff Senior software engineer. Helge Klüssendorff is a student at the University of Lübeck and currently working on his Master thesis. He is a member of the Hanse team since 2009. He has done work on the core framework, navigation, computer vision and behaviors. His interests lie in computer vision and mobile robots.

Marek Litza Academic advisor of Hanse Team since its foundation in 2009. Marek Litza is research assistant at the Institute of Computer Engineering, University of Lübeck. Besides technical aspects, his main interest lie in the question: "How can we teach a student team successful robot development?"

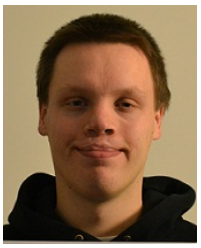
Benjamin Meyer Hardware managment, construction of the robot and maintenance. Benjamin Meyer is working as a Master student at the Institute of Computer Engineering, University of Lübeck. He is a member of the Hanse team since 2009. His interests are located in the construction of mobile robots, hardware design and sensor networks. He is responsible for robot maintenance and further development of the robot Hanse.

Christian Strakerjahn Hardware design. Christian Strakerjahn is a B.Sc. student at the University of Lübeck. He has joined the Hanse team in last winter term and has since been working on the design and integration of the pinger filter chain.

Tjorven Mintzlaff Construction of the robot and hardware development. Tjorven Mintzlaff is currently in the process to make his B.Sc. in computer science. He is a member of the team since the last winter term. His interests are located in the hardware construction of robots and the programming of embedded systems. His responsibilities are the development of new periphery and the construction of the backup hardware.

Thomas Tosik Thomas Tosik is a student at the Institute of Computer Engineering, University of Lübeck. He is part of the Hanse team since its foundation. In the last year he developed a simulation environment for AUVs as his diploma thesis. The simulator is now heavily used in the Hanse development and testing.

Patrik Stahl Software engineer. Patrik Stahl is also in the process of making his B.Sc. in computer science. He, too, is a member of the team since last winter term. His responsibilities are the implementation of new behaviors and software testing within simulation.



Patrick Zenker Software engineer and hobby photographer. Patrick Zenker is currently in the process of completing his B.Sc. in computer science. He is a member of the team since the last winter term. His responsibilities and interests are located in the implementation of new behaviors, especially wall following and sequential processing of different behavior tasks. Through his hobby, photos and videos are generated at every opportunity.

Year	Item	Cost (Euro)
2009/10	Mechanical (base frame, case, connectors)	355
	Electrical (fuses, emergency switch, cables, etc.)	50
	4x SeaBotix BTD150 Thrusters	1,280
	2x MD22 Motorcontrollers	140
	11.1V 10Ah Lithium Polymer Cells	130
	2x 11.1V 740mAh Lithium Polymer Cells	40
	2x Acer 12,1" notebooks	1,000
	Honeywell HMC6343 compass	100
	Analog Devices ADIS16354 IMU	250
	Intersema MS5541-CM pressure sensor and casing	14
	Imagenex Model 852 Scanning Sonar	6,000
	total	9,359
2011	XSens MTi AHRS	1,400
	Imagenex Model 852 Echo Sounder	2,300
	Improved USB cables and hubs	60
	Improved WLAN stick and antennas	50
	Pinger filter chain	30
	Spare case	
	Peli case	130
	Buccaneer connectors	140
	Electrical (fuses, emergency switch, cables, etc.)	50
	2x MD22 Motorcontrollers	140
	11.1V 10Ah Lithium Polymer Cells	130
	2x 11.1V 740mAh Lithium Polymer Cells	40
	Intersema MS5541-CM pressure sensor and casing	14
	total	4,484
	sum	13,843

TABLE I
TOTAL EXPENSES FOR THE HANSE AUV. EXPENSES FOR SAUC-E 2009 AND 2010 ARE LISTED ON TOP, THIS YEAR'S EXPENSES BELOW. ALL COSTS ARE LISTED IN EURO AFTER TAXES.

Risk	Precaution
Loss of control	• timers end tasks if a behavior fails • thruster speed will be set to 0 on communication failure • kill switch
AUV Recovery	• AUV will surface if thrusters are off • AUV can be easily recovered using e.g.
Collisions with objects or wall	• AUV speed too low to damage AUV in collision
Injuries due to lifting the AUV	• low weight • AUV can be easily lifted by two people at any point of the base frame
Injuries due to sharp edges	• most parts on the robot made of plastic • the cutting mechanism is protected by the guiding arms so that e.g. fingers cannot get near the cutting knife • no sharp edges on other exposed parts (frame and case)
Injuries due to thrusters	• propeller casing secures the thrusters
Injuries due to electrical shocks	• low voltages and currents • fuses in all main power lines in the case

TABLE II
RISK ASSESSMENT FOR THE HANSE AUV.