

# Mission Level Design of Autonomous Underwater Vehicles

Thomas Liebezeit, Volker Zerbe  
Department of Automatic Control and System Engineering,  
TU Ilmenau, Germany  
e-mail: *thomas.liebezeit@tu-ilmenau.de*

## Abstract

In this paper we present the idea of Mission Level Design (MLD) for autonomous underwater vehicles. Mission Level Design can be thought of as system design on an abstract level as well as an overall simulation in virtual worlds to test the design of complex systems. Missions (generalized use cases) embed the system and describe user requirements for the system design. This concept helps to find problems in early design phases and gives developers a first idea of system performance and characteristics.

We use MLDesigner, a design tool of the latest generation, for the design of an autonomous underwater vehicle (AUV). In this paper we present an overall system model with functional, architectural and environmental aspects. This model is used to answer design questions through the simulation of user defined missions in a virtual world. Results are presented.

## 1 Mission Level Design

Today more and more complex systems have to be developed. That's why independent specialists design modules described by the system specification. If there is an error within the specification it will be found during the testing period of the system. But this probably causes iterations back to the design phase, which are expensive. Therefore it would be better to find this kind of errors in the early design phase.

Previous design concepts began to deal with more complex systems by using levels of abstraction. This started with hardware description languages and continued with those for algorithms and systems. In this way, Mission Level Design (MLD) is just another step to the consequent formalization of the design process.

Mission Level Design deals with the fact that complex systems have too many parameters to check them all in every possible situation. Therefore possible operational requirements have to be defined and included in the formalization of the specification. These missions mirror typical use cases. The validation takes place by the simulation of the overall system model, where missions directly define driver and evaluation.

The overall system model consists of functional, architectural and environmental components, which define differ-

ent aspects of the system in a virtual world. The properties of those components are listed below:

- **Functional models** are the feasible parts of the specification of the system or its components. They describe the functional characteristics (continuous or discrete systems, software), therefore the behaviour of a system or component.
- **Architectural models** give a description of the available resources. They are used to examine the performance characteristics of a system or component. Normally, one functional model is mapped on different architectural models to find the best configuration.
- The **environmental model**, a set of mathematical models, describes relevant parts of the surroundings. Not yet implemented components may be integrated into the environment, too.

The aim of Mission Level Design is to test the design of the system with respect to its behaviour on a generalized model. A guaranteed error-free behaviour under operational conditions is essential for mission critical systems or systems with high financial risks. So, Mission Level Design is the description of a mission in a virtual world, an overall system simulation for defined operational requirements and the design test on mission level.

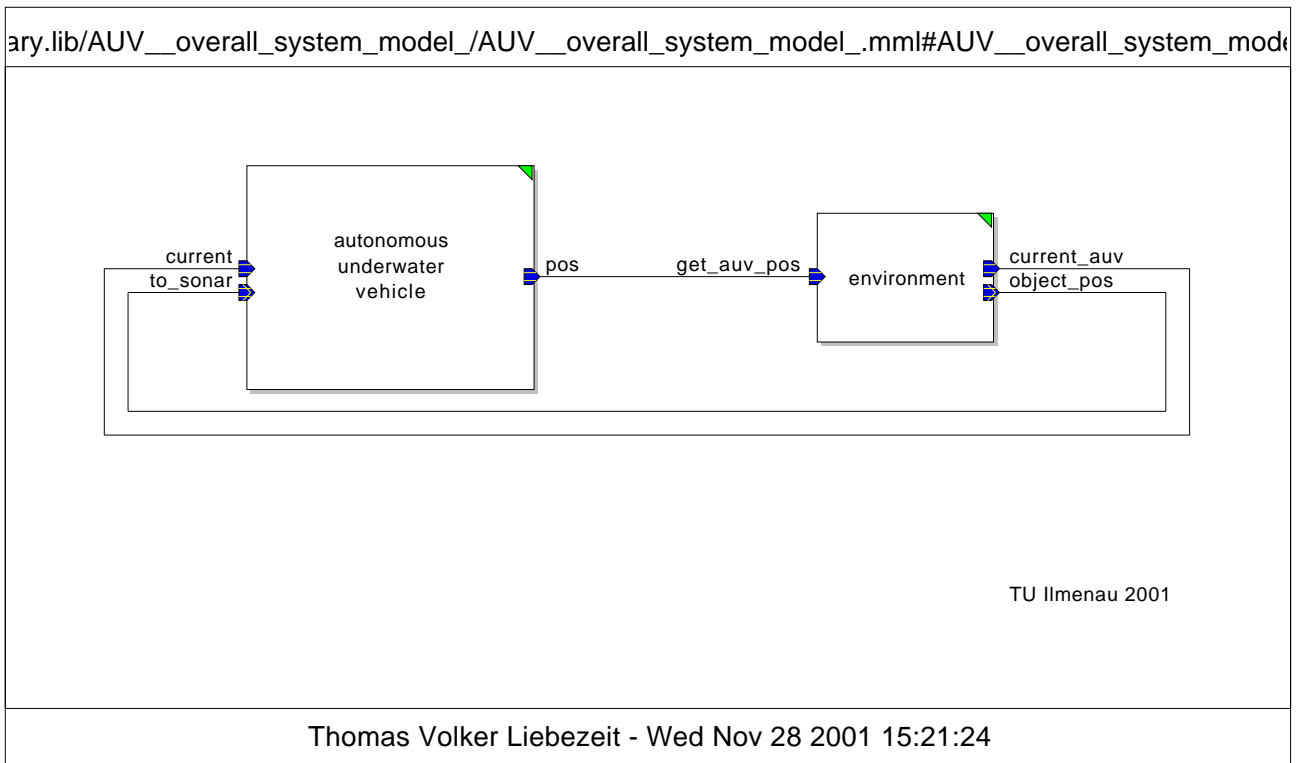


Figure 1: Overall system model

## 2 Autonomous underwater vehicle

### 2.1 The aim

We use the Mission Level Design concept for the design of an autonomous underwater vehicle (AUV). An AUV is a highly complex system and its loss would be very expensive. The aim of our work is to find answers to the following questions:

- Are the manoeuvres successful under limited resources and given environmental conditions?
- What happens with the battery power during the mission?
- Is the data bus design correct?
- Do we get enough sensor data?

### 2.2 MLDesigner

For this reason we need a simulation tool, which supports Mission Level Design, MLDesigner [4]. It supports the

simulation of different model types in one model. There exist basic models for time discrete, synchronous data flow, finite state machines and continuous time models. It is also possible to import BONEs and Cossap models or to develop own models in C++.

### 2.3 Overall system model

We have started to build the overall system model for the autonomous underwater vehicle, which is hierarchical and parametric as depicted in figure 1.

The environment and the autonomous underwater vehicle are placed at the highest level. Simultaneously, this mirrors the idea of Mission Level Design, which is to examine a system inside its environment.

**The environment** involves all relevant information about the surrounding. Therefore it consists of three relevant parts: First, there is the current of the sea, which is position dependent. Second, there may be other targets or obstacles. Finally, a collision detection for all the objects has been realized.

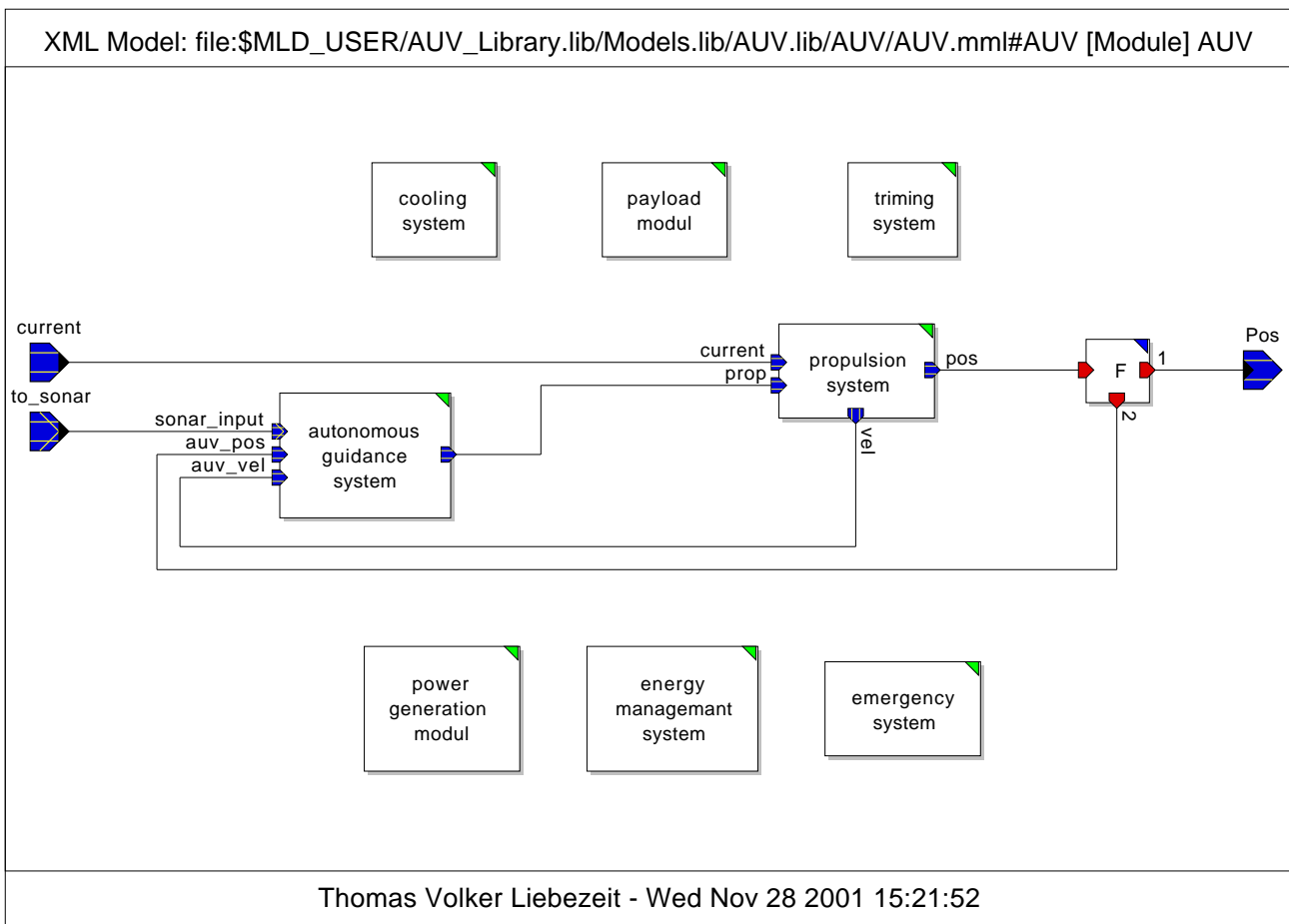


Figure 2: AUV model

**The autonomous underwater vehicle** is modeled by a set of sub-models representing important components. For an overview look at figure 2. It is taken from MLDesigner and shows the systems and modules of the AUV. These are:

- *cooling system*
- *payload*
- *trimming system*
- autonomous guidance system
- propulsion system
- *power generation*
- *energy management*
- *emergency system*

At the moment, most of the systems (the italic ones) are only predefined and will be implemented later. Each sub-system stands for fundamental functions. They will be designed for a real autonomous underwater vehicle by specialists on this topic. These experts use special models, which are too detailed for an overall simulation. Because of that, we use generalized versions of those models for our simulation.

**The propulsion system** includes engine and AUV dynamics. The equations of motion are adopted from Fossen[1] and Brutzman[2]. This model supplies the global position and the local velocity of the AUV.

**The autonomous guidance system** consists of the control computer, the sonar and the inertial navigation system (INS). This model is shown in figure 3.

The control computer controls the velocity and the position of the AUV. A path planning module creates the

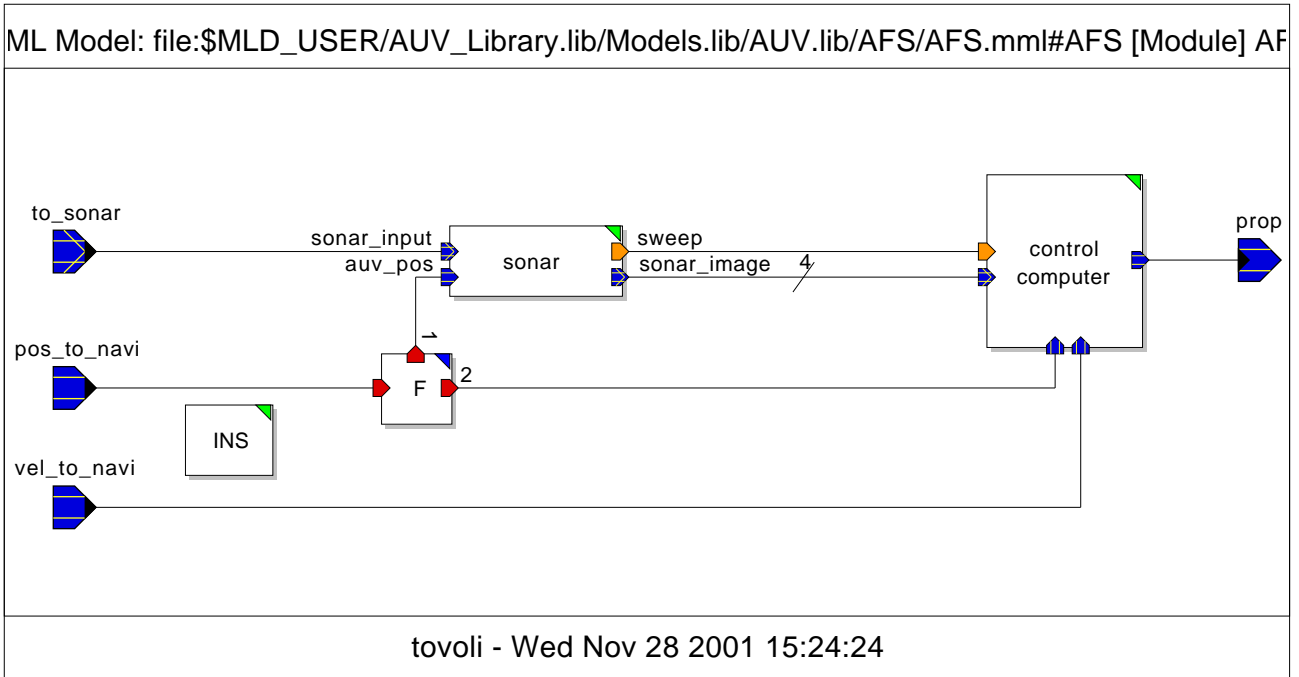


Figure 3: Autonomous guidance system model

desired values. Additionally, the control computer includes an image processing module to analyze the sonar image. Its results are used by a collision avoidance module, which is interacting with the path planing module.

The sonar model, depicted in figure 4, is used to detect objects in the surrounding of the boat. As a functional model it does not base on exact physical behaviour. Instead a geometrical approach is used, in which physical parameters are mapped to geometrical ones. The sonar model receives the position of the AUV (trans) and those of the other objects (input), which are expressed in global coordinates. That's the reason why the object positions are first transformed into AUV coordinates. The SonarMotion-block controls the sonar's field of vision and behaves as follows: It pans from *minPing* to *maxPing* and back in the sonar plane with a steering increment of *delta*. The sonar waits *wait* iterations after each ping. A complete sonar image will be signaled on the *sweep* output. The sonar position is used to transform the object positions in the sonar coordinate system. Here the SonarView-block checks, if the objects are in the field of vision. This happens in dependency of the *run* signal from the SonarMotion block. The accepted space for the field of vision is defined by  $\pm maxAngleH$  and  $\pm maxAngleV$  (in the horizontal and vertical plane, respectively) and the maximum distance *maxDist*. The SonarView output is a stream of objects in the field of vision. Their local sonar positions are transformed back

to local AUV positions. These coordinates build together with the *sweep* signal the sonar output. For the visualization of the simulation result some values are logged by the print blocks.

## 2.4 Mission

As a first test mission we have chosen the following situation: The boat has to hold its velocity and shall avoid collisions with appearing objects. Therefore one drifting and three fix objects are lying in the path of the AUV. This mission is used to test the parameter settings of the sonar.

The starting configuration is shown in figure 5. The AUV starts at  $(0, 0, 0, 0, 0, 67)$  with  $(2, 0, 0, 0, 0, 0)$ , whereas the  $(6 \times 1)$ -vector describes the three translational and three rotational positions (*m, deg*) or velocities (*m/s, deg/s*). The fix objects are at  $(55, 100, 0, 0, 0, 0)$ ,  $(60, 90, 0, 0, 0, 0)$ ,  $(65, 80, 0, 0, 0, 0)$  and the drifting obstacle starts at  $(30, 40, 0, 0, 0, 0)$ . The sea has a current of  $(1, 1, 0, 0, 0, 0)$ .

As mentioned above, the aim of this mission is to hold the starting velocity and to avoid collisions with objects, that appear in the sector  $\pm 10^\circ$  in front of the autonomous underwater vehicle. The sonar scans a sector of  $\pm 22.5^\circ$  and waits one simulation step after each ping, which has

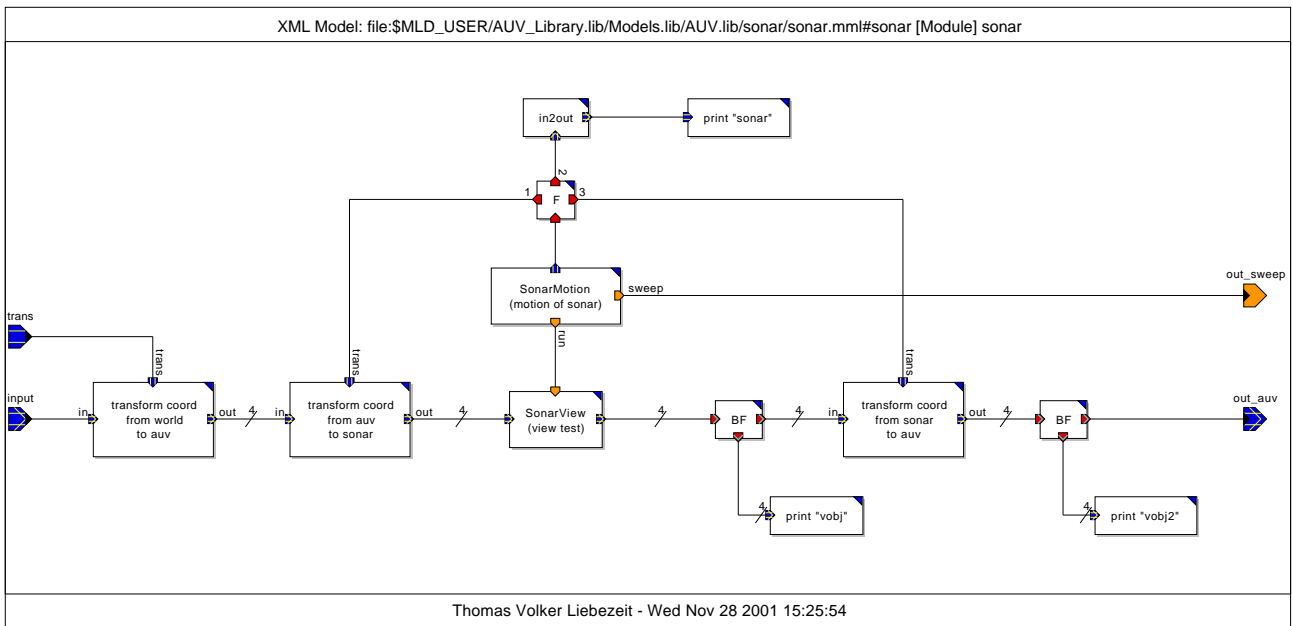


Figure 4: Sonar model

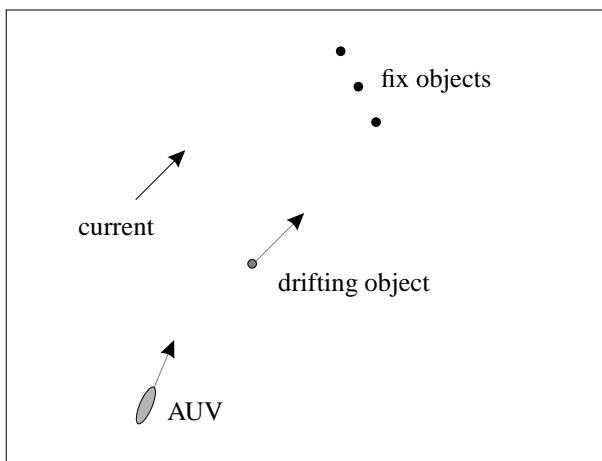


Figure 5: Start configuration

a beam shape of  $1.8^\circ$ . The maximum range of the sonar is  $80m$  and the steering increment amounts to  $1.8^\circ$ .

### 3 Results

The simulation results are shown in figure 6. The sonar image in subfigure 6(a) consists of four curves, one for each obstacle. This means that every object was scanned by the sonar during the simulation. All sonar data is presented in local sonar coordinates.

Figure 6(b) plots the distance of the scanned objects to the AUV over the simulation time. First the drifting object appears in the sonars field of vision. Later the three fix obstacles are detected.

The detection of the last object after approximately 1100 simulation steps causes a collision avoidance manoeuvre, as depicted in the control computer decisions in figure 6(c). This happens because the obstacle appears inside the  $\pm 10^\circ$  sector of the sonar.

The resulting trajectory of the AUV in the  $xy$ -plane of the virtual world is shown in figure 6(d). The bend of the curves, caused by the avoidance manoeuvre, is clearly recognizable.

### 4 Conclusion

We have presented the idea of Mission Level Design on the example of an autonomous underwater vehicle. The concept of MLD was explained, an overall system model was presented and used for testing the parameter settings of an AUV sonar.

For a better evaluation of trajectories, a 3D visualization tool will be used. Another important fact of our future work is the management of different missions in one database.

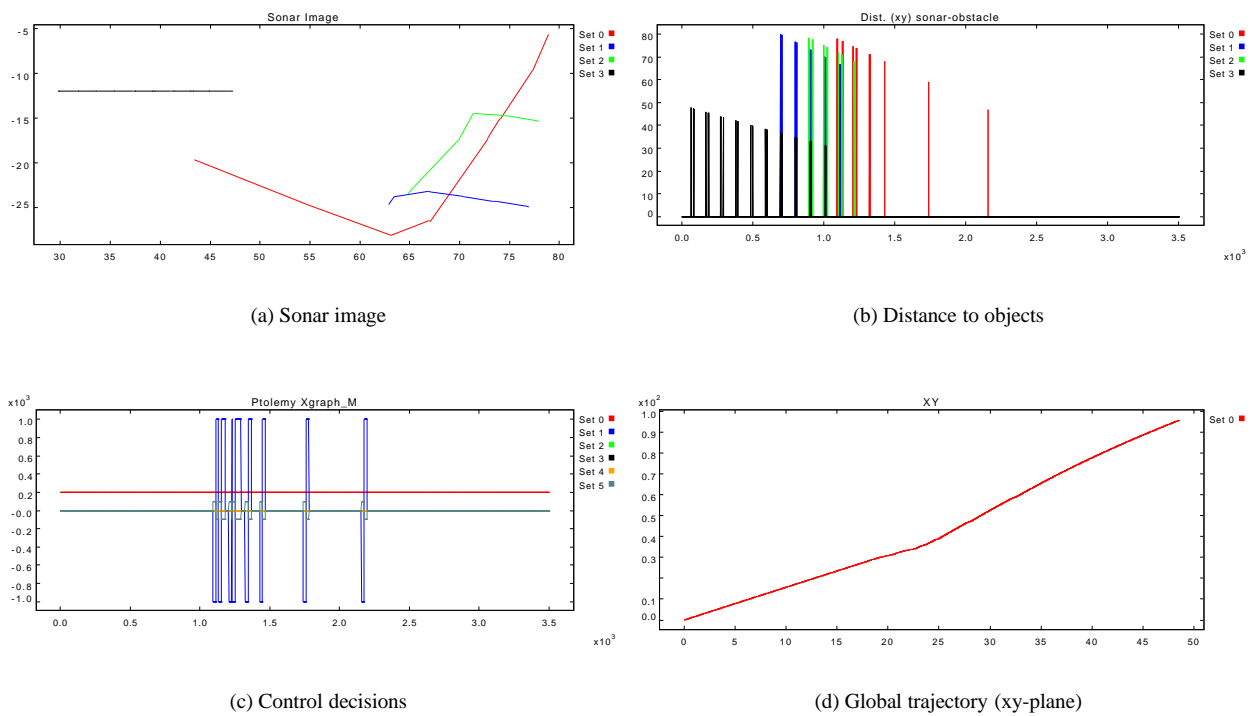


Figure 6: Simulation results

## References

- [1] Fossen, Thor I.: Guidance and control of ocean vehicles, John Wiley & Sons Ltd. 1994
- [2] Brutzman, Donald P.: A virtual world for an autonomous underwater vehicle, dissertation Naval Postgraduate School Monterey California 1994
- [3] Zerbe, V.; Liebezeit, Th.; Radtke, T.: Mission Level Design in Robotics. In: Proc.-CD 10th International workshop on robotics in Alpe-Adria-Danube region (RAAD'01) Vienna, May 2001
- [4] MLDesigner: <http://www.mldesigner.com>