Safety in Robotics: the Bremen Autonomous Wheelchair

Axel Lankenau, Oliver Meyer and Bernd Krieg–Brückner
{alone, emm, bkb}@informatik.uni-bremen.de
Bremen Institute of Safe Systems
TZI, University of Bremen
P.O. Box 330440, 28334 Bremen, Germany

Abstract

This paper describes the development of a safety layer for an electrically driven wheelchair that is equipped with a ring of ultrasonic sensors. Designing this safety layer, which consists of software modules as well as hardware components, involved formal methods. An important outcome of the presented formal approach is a detailed set of hypotheses that specify the requirements the environment has to satisfy in order to allow the robot to behave as intended.

The safety layer builds an abstraction of the real wheelchair which ensures that collisions are avoided, and guarantees that communication between software modules in a multi–PC environment takes place in real–time. As a first application on top of the safety layer, a driving wizard has been implemented. Its task is to adapt the wheelchair's speed depending on the situation of obstacles in the surrounding of the vehicle. The driver has no longer to bother with the question whether the planned trajectory will lead to a collision or not.

1 Introduction

The impact of the correct behaviour of a robot upon the health of human beings increases with the spatial proximity in which man and machine operate.

Therefore, the aspects of both safety and functionality according to the specification play a paramount role, especially within the field of rehabilitation robotics. In contrast to conventional industrial robots or even common service robots, rehabilitation robots, such as intelligent manipulators or smart wheelchairs, have to satisfy strict requirements with respect to the dependability of the technical system and with respect to the specific needs of handicapped and elderly users.

Within the framework of the Bremen Autonomous Wheelchair project a commercial Meyra electrically driven wheelchair (cf. Fig. 1) serves as an experimental platform for navigation experiments [5] and as a prototype of a sophisticated aid in daily life for the handicapped.

Figure 1: *The second prototype of the Bremen Autonomous Wheelchair*

2 Safety in Robotics

So far, pursuing the idea of designing a robot as a *safe* embedded system has not been the dominating concern for researchers in science and industry. Partly, this is due to the fact that many projects aimed and still aim for a mere existence proof (cf. [1]). Often, safety aspects are indeed taken into account but nevertheless are neglected badly, e. g. by merely setting up a solid fence around an industrial robot in order to avoid that unforeseen conditions cause the software or the hardware to behave catastrophically.

Especially in the field of rehabilitation robotics such a policy is no longer appropriate. Here, in contrast to other robotic applications, the health of a human user – who is likely to be a handicapped or elderly person – directly depends on the correct behaviour of the robot. Therefore, it is indispensable to take into consideration safety aspects during system design. In order to make a verification of essential system properties possible, a formal specification is necessary.

For the wheelchair application described in this paper, it proved useful to extend Laprie's [7] definition of "safety". He states that safety is *nonoccurrence of catastrophic consequences on the environment.* Interpreting "catastrophic consequences" in a broader sense not only as collisions of the vehicle but also as the nonexecution of a user command ("Go to the medicine cabinet"), results in an adapted conception of safety. It subsumes the ideas of classical safety, availability and reliability.

These two major safety topics (collision avoidance vs. unlimited user–control) are inherently self–contradictory as the scenario shows in which the handicapped user wants to pass through a doorway by pushing a door open. To cope with these conflicting requirements, a hierarchy of safety items has been set up stating that at every time a collision has to be avoided. If there is no obstacle dangerously close to the wheelchair, the commands of the user must be realized unchanged.

3 System Architecture

The safety layer mentioned above has to serve as a safe interface between the real wheelchair and the application modules. As there are strict real–time requirements with regard to sensor and actuator I/O for the PC hosting this safety layer, the real–time operating system QNX is used. On the other hand, many devices commonly employed in robotics (cameras, robot arms) need standard operating systems. To cope with the resulting diversity of operating systems, a multi–computer environment with an underlying real–time network has to be installed on the wheelchair. These obligations directly lead to the system architecture described in the sequel (cf. Fig. 2).

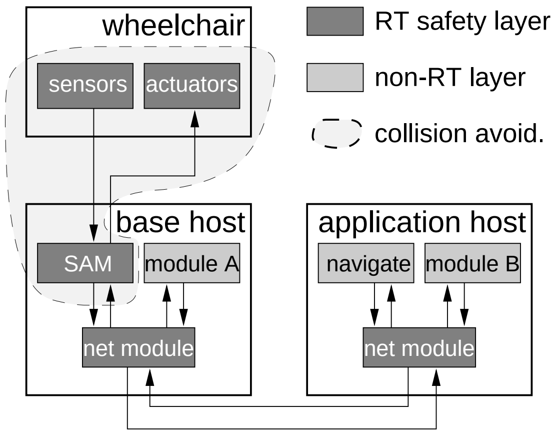

Figure 2: *System architecture.*

3.1 The Real–Time Safety Layer

The real–time safety layer forms the basis of the whole system by providing an anti–collision module as well as a real–time communication protocol for information exchange between the software modules. As depicted in Fig. 2, the real–time safety layer consists of sensors and actuators, the Sensor/Actuator–Module (SAM) and the net modules. These components will be explained in some detail in the following paragraphs.

The Hardware

The wheelchair is equipped with a ring of 27 sonar sensors (see Fig. 1). Via two multiplexer boards, the sensors are connected to an ISA–card in the PC (base host in Fig. 2). Apart from the internal sensors of the wheelchair measuring actual speed and steering angle, the sonars are currently the only source of information about the environment the real–time safety layer makes use of. The maximum speed of the wheelchair is about 3km/h. Manoeuvring the wheelchair, which is driven by its front wheels and steered by its back wheels, is similar to driving backwards in a car.

The Sensor/Actuator–Module

The Sensor/Actuator–Module (SAM) serves as an interface to the wheelchair and its sensors. SAM receives commands from the applications and passes them to the wheelchair. Besides, it makes sensor readings as well as additional internal data available. SAM processes the information gathered by the internal and external sensors and brakes to a standstill whenever an obstacle is dangerously close to the wheelchair. Note that the intervention is performed at the latest possible point in time, i.e. depending on the current speed the stop command is issued such that the wheelchair comes to a standstill in a distance of not less than $5\,cm$ to the object. That way, SAM, the sensors and the actuators build the part of the safety layer that averts the threat of a collision.

The Net Modules

The other fundamental safety requirement, dealing with communication ensuring unlimited user control, is covered by the net modules. They are implemented as parts of the net interrupt handling routines on the hosts. The net module of the base host controls the communication protocol used on top of the Ethernet network. It is a frame–protocol using a time division multiple access (TDMA) approach to ensure data transmission that is free of collisions.

The net modules provide an interface for the applications enabling them to send and receive data, independent of the specific computer hosting the application, within guaranteed real–time bounds.

3.2 The Application Modules

An application module is a piece of software that produces data for or receives data from SAM or other application modules. These modules may be regarded as clients whereas the real–time safety layer is the server. For example, in Fig. 2 module B might be a self–localization module that determines the position of the robot in its environment by processing data gathered by a video camera. navigate might be a navigation task that steers the robot according to the information it received from the self–localization module. These modules may run on any hosts in the net, they communicate with each other as well as with SAM.

It is worth mentioning that an application module need not necessarily be subject to timing constraints, i.e. the interface between the real–time safety layer and the application modules may be regarded as an interface between hard real–time and non real–time conditions. The system architecture described above avoids the common problem often arising when integrating real–time and non real–time components in one system: the propagation of the strict timing constraints of the original real–time kernel to the outer, originally non real–time parts of the system.

The first implemented application module, the *driving wizard*, is presented in section 5.

4 Formal Approach

Due to the importance of safety aspects in this rehabilitaion robotics application, a new formal system–design method has been developed. This approach, first described in [6], is summarized in this section.

4.1 Threat Analysis Using Fault Trees

When designing a safety critical system, a threat analysis is a necessary step to determine the possible problems and faults a system has to cope with. One commonly applied method to systematically derive the threats to a system is the fault tree analysis technique that is used here (cf. [8],[3]). It results in a fault tree that gradually links abstract to more specific threats. The leaves of the tree represent the "atomic threats" which cannot adequately be split into more specfic threats. The safety system later has to ensure that the highest threat in the fault tree never occurs.

Figure 3 depicts a small segment that is part of the fault tree developed for the Bremen Autonomous Wheelchair. The entries which have the same indentation level are a refinement of the higher threat. The first character describes the logical relation between threats at the same level. A higher threat occurs if and only if all lower threats occur (`&`) and if one of the lower threats occurs (`|`), respectively. For example, a measuring error may cause a collision if and only if it occurs (`X.1.1`) *and* leads to an overestimation of the distance between the wheelchair and that particular obstacle (`X.1.2`). Leaves marked with a hash (`#`) do not have to be taken into account in the system design since they are averted by already existing external mechanisms. Instead, they have to be considered for the definition of the environment properties (see next section). Leaves marked with a dot represent atomic threats which have to be averted by appropriate saftey mechanisms.

In the fault tree covering the Bremen Autonomous Wheelchair the threat number `X` leads to a more abstract threat called `collision` which, for its part, leads to the overall threat `Safety requirements of the whole system violated` (cf. [6]). The complete fault tree consists of about 170 nodes and leaves. The high–level nodes of this tree deal with problems that are common in (mobile) robotics applications and therefore at least the upper parts of it can be adapted easily for other projects. Since the fault tree leads to a functional decomposition of the threat space, more specific nodes can also be reused if similar requirements exist. For example, the segment in Fig. 3 can be employed in any mobile robotics application that

```
X Disastrous failure of external sensors
  | X.1 Measuring error able to cause a collision
        & X.1.1 Too large values measured by distance
                sensors
                | X.1.1.1 Too large values measured
                          up to n consecutive times.
                | X.1.1.2 Too large values measured
                          more than n cons. times #
        & X.1.2 Obstacle distances overestimated.
  | X.2 Disastrous breakdown of external sensors
        & X.2.1 No distances measured #
        & X.2.2 Error not detected.
  | X.3 Obstacle not detectable by distance sensors #
```

Figure 3: *Extract of a fault tree segment dealing with the problems of external sensors.*

uses some kind of sensors to measure the distance to the surrounding obstacles.

4.2 Derivation of Safety Requirements

The atomic threats in the fault tree are usually employed to define safety requirements which can be determined by negating the statements of the leaves. The safety system has to ensure that these requirements always hold with respect to the fault tree hierarchy, i.e. it is sufficient to avert only one of several AND–connected threats to ensure that the higher threat cannot occur. The next sections describe how this information can be used to design the safety system and verify its correct behaviour.

Sometimes it is impossible for the safety system to ensure that a safety requirement holds. For example, consider the leaf `X.3` in the fault tree depicted in Fig. 3. The corresponding safety requirement is `All obstacles are detectable by the distance sensors`, which cannot be guaranteed *by the wheelchair* without employing perfect sensors. Unfortunately, such sensors do not exist and, moreover, the costs of a system are a major topic to be taken into account. Therefore, a compromise has to be found between a sophisticated but expensive sensor equipment and a suitable environment specification. This means that surface and structure of the obstacles have to be chosen with respect to the available sensors in order to ensure that the obstacles are detectable.

The Bremen Autonomous Wheelchair is equipped with a ring of ultrasonic sensors that are able to detect obstacles at a certain height only. This leads to the environment requirement stating that all furnitures etc. in its surrounding need to have the largest extent at this particular height because otherwise they are potentional obstacles but they are not detectable by the robot. Moreover, due to technical reasons ultrasonic sensors have difficulties to detect very smooth surfaces. This results in a claim for at least slightly rough surfaces.

The fault tree analysis technique is an appropriate means to derive all environment requirements systematically. This way it is possible to get a complete specification of the environment properties the designed safety system relies on.

4.3 Safety Mechanisms

In order to ensure that the highest threat in the fault tree cannot occur, it is necessary to develop safety mechanisms which guarantee that the safety requirements are never violated with respect to the fault tree hierarchy. The safety mechanisms have to prevent the environment and a naive realization of the system from performing undesired actions, i.e. they have to block all actions that may lead to a catastrophic system state (see also Fig. 4 and caption).

Concerning the threat represented by node `X.1.1` this means that a collision avoidance safety mechanism is needed that can cope with `n` consecutive measuring errors. Such an algorithm has been developed in [6] and is outlined in the sequel.

If a mobile robot has to rely on distance sensors that may have an *unbounded* number of consecutive measuring errors, the only safe state would be a standstill. Therefore, it is essential to determine the upper bound `n` of acceptable successive measuring errors by defining the necessary reliability of the whole system. This reliability depends on the remaining probabilities of a violation for the selected upper bound. Note that for the presented threat only measuring errors potentially leading to an *overestimation* of the distance between the robot and an obstacle have to be taken into account since other measuring errors cannot cause a collision.

With this precondition at least one value of `n+1` successive readings must be considered to be either correct or too small. Using only the smallest of these `n+1` values (worst–case assumption) never leads to an overestimation of the distance to the obstacles. However, the wheelchair moves and the obstacles change their relative position to the robot so that old sensor values have to be adapted in each step. This is achieved by accumulating the most recent `n+1` sensor readings in a local grid map that always represents the current surroundings of the wheelchair. The map entries are shifted according to the movement of the

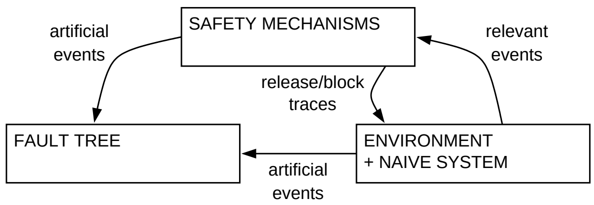

Figure 4: *Fault tree based verification. The naive system realizes the desired functionality without taking into account the possible threats.*

wheelchair. In order to guarantee the use of the *closest* relevant entry in the map, "virtual sensors" are employed which measure the distance between the robot and the obstacles not in the real world but in the local grid map. They take into consideration the kinematics of the wheelchair to ensure that the movement will not be limited unnecessarily.

This safety mechanism can avert the threat `X.1.2` and, therefore, the occurrence of at most `n` measuring errors with too large values is acceptable since it cannot lead to an activation of the threat `X.1`.

4.4 Verification Approach

The previous section shows that the development of safety mechanisms is a nontrivial task and may be erroneous. Therefore, verification against the undesired behaviour, specified in the fault tree, is necessary. The verification approach described in [6] uses the fault tree as an active observer of the system behaviour. The tree has been implemented in the formal language CSP (cf. [4]) and runs in parallel to the specifications of the environment, the naive system and the safety mechanisms (see Fig. 4).

The model of the environment is able to behave maliciously with respect to the acceptable faults specified in the fault tree, for example, it will never cause more than `n` consecutive measuring errors. Whenever a threat occurs, the environment will also trigger an "artificial event" to signal it to the fault tree. Note that the safety mechanisms can also trigger artificial events since they might introduce new threats to the system. The occurred threats are propagated up the tree according to the specified hierarchy. The occurrence of the highest threat means that the system failed. In order to verify that the highest threat can *never* occur, all interleavings of all system components have to be examined. This can be achieved by using a model checker, such as FDR (cf. [2]). In [6] this procedure has been used to verify that the safety requirements, concerning the real–time protocol indicated in Sec. 3.1, hold.

5 The Driving Wizard

The first application module using the safe interface of the black box delivered by the safety layer is the driving wizard. This module reads the user commands from the interface of the joystick. These commands are passed to the safety layer and, if no obstacle is too close to the wheelchair, the commands are passed to the actuators as target speed and target steering angle. If an obstacle comes close, the driving wizard reduces the speed of the wheelchair. Thus, a "smooth" driving behaviour is obtained — in open areas, the driving wizard does not affect the instructions of the user whereas in narrow space, the driving wizard regulates the speed with regard to the remaining distance to the closest obstacle.

Note that the driving wizard behaves different from SAM: By accumulating the latest measurements, SAM is able to cope with sensor misreadings. It brakes as late as possible, i.e. it sets the target speed to zero whenever an obstacle is too close. This way, SAM ensures that the wheelchair never collides with objects in the environment but it restricts the application modules as *little* as possible. However, a human driver feels rather uncomfortable because the braking manoeuvres are very abrupt, due to the immediate reduction of speed to zero. In contrast to SAM, the driving wizard already intervenes at an earlier stage. It processes the *current* sensor readings and, depending on the distance to the closest object, it reduces the speed by ten percent. If current sensor readings are wrong, the driving wizard fails to perform a smooth braking manoeuvre. Despite this fact, the wheelchair remains in a safe system state because the underlying safety layer (SAM + hardware) would cause a speed reduction to zero in time.

Experiments have shown that the system is able to avoid collisions in an environment that satisfies the restrictions specified during the threat analysis. The user may concentrate on simply indicating the target direction and the driving wizard pays attention to obstacles dangerously close to the wheelchair. A typical experimental set–up is shown in the upper part of Fig. 5. The wheelchair drives towards a doorway. As the obstacles come close, the driving wizard reduces the speed until the current sensor readings signal that no object is in the way. This situation is depicted by the light grey wheelchair and its measurement range.

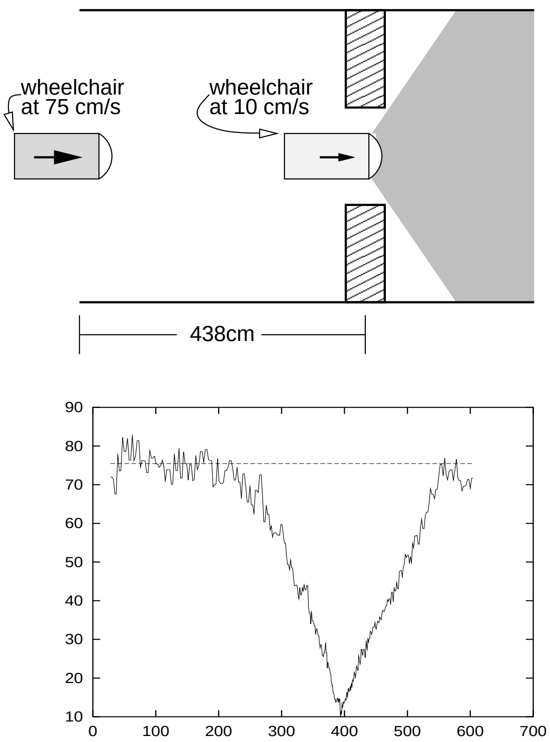

Figure 5: *Experimental set–up (upper part) and visualization of current (solid line) and target (dashed line) speed (lower part). Same scaling of horizontal axes in both figures.*

The lower part of Fig. 5 shows the corresponding current and target speeds during the experiment. The target speed is constantly about $75\,cm/s$ (represented by the dashed line). As the velocity sensor is not very accurate, the current speed values build a zig–zag curve (solid line). Note that it takes about $2\,m$ to *smoothly* reduce the speed to $10\,cm/s$, whereas an abrupt braking process would only have taken $40\,cm$. The speed is increased again at a total distance of $4\,m$ from the starting point of the experiment, because the relevant front sensors do not detect any close obstacle in their sensing range (depicted as dark grey area in the upper part of the figure).

Using the driving wizard the user does not have to bother with the question how fast the wheelchair could be driven in a certain situation. He or she simply sets the intended direction while the driving wizard adapts the speed according to the environment. This way it becomes easy even to safely pass through a $90\,cm$ wide doorway with the $70\,cm$ wide wheelchair.

6 Conclusion and Future Work

The new formal design method presented in this paper leads to a specification of a dependable wheelchair and a definition of the environment in which it is able to operate safely. Both aspects are essential in rehabilitation robotics applications, but have been neglected in most research and commercial projects so far. Nevertheless, some topics have to be refined in the future in order to broaden the applicability of the approach. The formally developed collision avoidance mechanisms should be verified in a way similar to the presented method for the protocol verification, with the reservation that the employed model checking approach seems not to be sufficient to cope with a model of the real environment of a robot.

In addition, further applications on top of the safety layer will be developed, e.g. the extension of the functionality of the driving wizard to an obstacle avoidance module that bypasses objects and returns to the original trajectory afterwards.

References

[1] T. Duckett and U. Nehmzow, "Quantitative Analysis of Mobile Robot Localisation Systems", *Proc. TIMR UK 97*, Manchester 1997

[2] Formal Systems Ltd., *Failures–Divergence Refinement*, FDR2 User Manual, Oxford, 1997

[3] K. M. Hansen, *Linking Safety Analysis to Safety Requirements*, PhD Thesis, DTU Technical University of Denmark, 1996

[4] C. A. R. Hoare, *Communicating Sequential Processes*, International Series in Computer Science, Prentice Hall, 1985

[5] B. Krieg-Brückner et al., *A Taxonomy of Spatial Knowledge for Navigation and its Application to the Bremen Autonomous Wheelchair*, in: Ch. Freksa et al. (Ed.), Spatial Cognition. LNAI 1404, Springer, Berlin (1998)

[6] A. Lankenau and O. Meyer, *Der autonome Rollstuhl als sicheres eingebettetes System*, Masters thesis, Universität Bremen, 1997

[7] J. C. Laprie et al., *Dependability: Basic Concepts and Terminology*, Springer–Verlag, Berlin Heidelberg New York, 1992

[8] U.S. Nuclear Regulatory Commission, *Fault Tree Handbook*, NUREG–0492, 1981