

Automated model verification of the International Space Station for path planning

W.F.W. Haak^{a,*}, F.C.A. Groen^a, E. Holweg^b

^a *University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, Netherlands*

^b *Fokker Space, P.O. Box 32070, 2303 DB Leiden, Netherlands*

Received 1 December 1998

Abstract

The goal of this project is an automated model verification tool for the International Space Station. The 3D model of the ISS is used as a basis for path-planning the European Robotic Arm. The ISS contains several different types of video cameras. The images from these cameras are used to verify the model.

Each camera image is compared with an artificially generated image by a graphics engine. An iterative approach optimizes the artificial image generation by calculating the exact camera orientation. The 2D match can be traced back to a position in 3D space.

The system was tested with the use of a mock-up of the ISS. Test results show that the model verification system detects object changes as small as 2% of the field of vision. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Autonomous systems; Computer vision; Model verification

1. Introduction

Autonomous systems often rely on a 3D model of the real world. Real-world objects are not static but prone to change. Most systems that interact with the real world should be able to detect these dynamic changes and update the model.

The 3D model verification system that has been studied in this paper is that of the International Space Station (ISS). On the ISS the European Robotic Arm (ERA) will be operating. The path planning of the ERA will be based on a 3D computer model of the ISS. The aim of this project is to build a model verification system that safeguards against discrepancies between the computer model and the real world. Many systems have a-priori knowledge of the real world and it is the purpose of this project to use this knowledge to build a robust model verification tool that is based on a-priori information.

The ISS has a variety of on-board cameras. These will be employed to generate visual data. The verification of the 3D model can of course be accomplished by human analysis of the image data. The sheer amount of data and

* Corresponding author. E-mail: haak@xs4all.nl

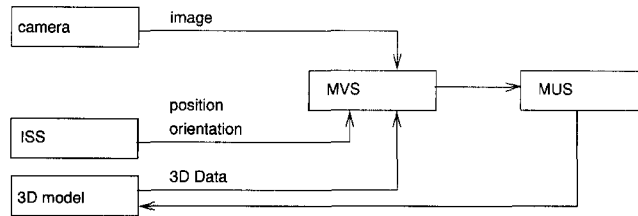


Fig. 1. Model verification and update system functional diagram.

the relatively small amount of “interesting” images, i.e. images where some discrepancy could be detected, point against this approach and require an automatic model verification system.

In order to test the model verification system, a mock-up of the ISS is used.

2. Automatic verification and update system

The basic requirements of the model verification and update system are:

- (1) The system should verify the validity of a 3D computer model on the basis of camera images of the space station presented to the system.
- (2) The system should decide which portions in the image contain discrepancies and link the discrepancies with the 3D computer model.
- (3) A suggestion for the model update should be given.

Besides that, the system should be modular and open so that extensions could be easily made.

The two basic systems are the model verification system (MVS) that detects changes, and the model update system (MUS) that incorporates these changes in the model (see Fig. 1). Essential modules for these two systems are the 3D model database and the interpreter of this database, the camera module, and the module that keeps track of the absolute position and orientation of ISS in space.

2.1. Model verification system

The basic requirement for the MVS is to match an image with the 3D computer model. The result of this match should be a decision which regions in the image match with the 3D object and which regions do not. The matching of 3D data with 2D data involves a transformation. Either the 2D image is transformed to 3D or the 3D object is projected on the 2D plane. Both ways, the 2D→3D match and the 3D→2D match, are possible.

The matching of 2D objects is much easier than the matching of 3D objects. There is also no simple way to incorporate information like shadows and reflections in the 3D domain. The choice was therefore made to match in the 2D domain.

For the 3D→2D match there are two ways to match the camera image with the 3D model. The first is to extract features from the camera image and compare these with features that were directly extracted from the 3D model. The other way is to model the real camera image by a virtual camera and a graphics engine that produce a synthetic image. After the synthetic image has been produced, both images can be approached in the same way.

The extra step of producing a synthetic image in order to extract features that could have been extracted directly from the 3D model may seem superfluous. The reason that the synthetic image helps solving the matching problem is because the synthetic image models extra information like secondary light sources, lens aberrations, reflections, cast shadows, and occluding objects. Many features in the camera image are the result of these and other factors.

2.2. Model update system

The model update system is the connection between a decision of an object change and the actual update of the 3D computer model. The model update system can be implemented in many different ways. To create a fully autonomous update system is dangerous, so the most logical choice would be to create a tool which suggests 3D model changes. The final decision would be by a human operator. There are three possibilities for a model change:

- (1) *The disappearance of an object.* The object needs to be removed from the model.
- (2) *The transformation of an object.* The object needs to be replaced by the transformed object in the model.
- (3) *The addition of a new object.* The object needs to be added to the model.

The first possibility is easy to implement. Once it has been established which object needs to be removed from the model, this is a trivial task. The second and third are not so trivial.

The transformation of an object results from the non-rigidity of the object. Take for example a solar-array that folds in or out. The system will have to use a-priori information about such objects.

Object addition is a complex operation. First of all, the location of the object needs to be calculated relative to the main object. Second, the characteristics of the new object need to be calculated. This is a difficult task since it cannot be absolutely certain that the object is entirely visible from the camera viewpoint, so in general multiple viewpoints will be needed.

In this paper, the focus is on the model verification system.

3. Graphics engine

The generalized viewpoint of the real camera is approximately known. We need an artificial camera that can take an artificial picture of the 3D computer model that comes as close to reality as possible. This means that the ISS position relative to lightsources (sun, earth and moon) has to be calculated.

In order to be able to match the image, it is necessary to model shadows. Shadows come in two shapes: cast shadows and shaded objects. Object shading is not difficult to model. If the orientation and texture of a surface is known, the shading can be easily calculated. Cast shadows are difficult to model because they depend on the relationship between objects, light sources, and object characteristics. The modeling of shadows is very important because they play such an important role in space.

A graphics engine that is able to come up with realistic images needs (listed in order of increasing complexity): perspective projection, direct sunlight, reflected sunlight from earth and moon, color, shading, texturing, camera and lens model, cast shadows and reflections.

A number of graphics engines have been tried and the two most promising are Rayshade [6] and Performer [3]. The Rayshade library meets all the above requirements. The Performer library also meets all of the above but the cast shadows are faulty and inaccurate and it cannot model reflections. Rayshade employs ray tracing, a computationally expensive algorithm. Performer models shadows using projected texture mapping. Performer is incredibly fast. The setup and production of an entire image take about 0.5 s whereas the same image produced by Rayshade takes about 50 s.

Both Performer and Rayshade are employed. If the original image can be matched using the Performer image, the image can be discarded. If unexplainable parts arise, Rayshade is used.

4. Feature detection

The matching of images is illustrated in Fig. 2 and is based on the matching of features between real and artificial images. The matching takes place in two steps. First the camera parameters are determined to obtain an optimal matching between a real and artificial image. Then the differences between the images are determined. Both steps use image features in order to accomplish the match.

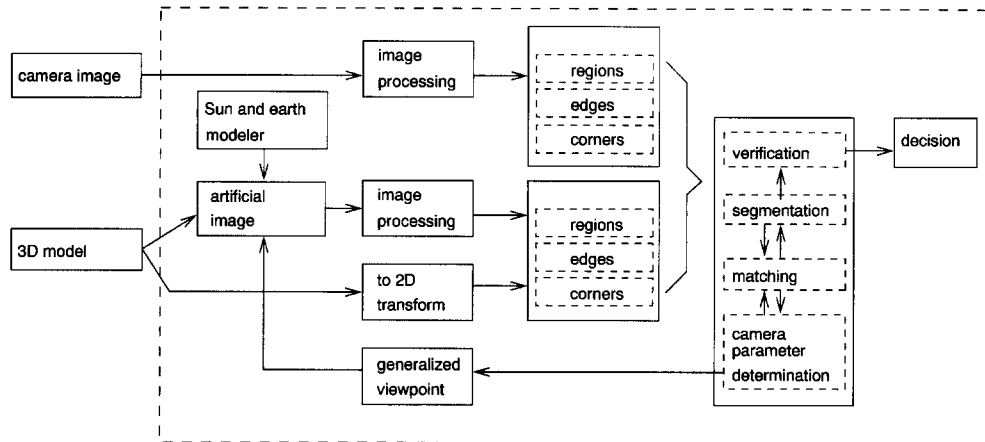


Fig. 2. Model verification system functionality.

There are three types of image features that can be used:

- (1) *Region-based feature matching*. Image represented by surfaces with parameters such as: color, texture, intensity.
- (2) *Edge-based feature matching*. Image represented by line segments and curves with parameters such as position and orientation, line length, width, and strength.
- (3) *Special points*. Image represented by points with parameters such as position and strength.

Each technique has different strengths and weaknesses. Surfaces contain information about texture and intensity but their position cannot be accurately determined. This means that region-matching algorithms give information about color and texture but not about the possible difference in position of the two images. Lines and points give exact information about the difference in position but they cannot be found everywhere in the image. We have used the three different approaches in combination to exploit the strength of all three.

4.1. Line and point detection

We have used the second derivative operator [2] for edge detection. An edge in an image is transformed to a contour [4]. These contours are approximated by straight line segments [8] and represented in a graph.

Similar edges in both images will produce comparable series of line segments, but the line segments will differ at several points. The length of an edge is not robust. Edges may be intersected by noise or shades or parts of an edge may fall below a certain thresholding level. The transformation of an edge to line segments is therefore not robust with respect to line lengths. The same holds for the starting and endpoint of a line, and also for branches and corners. So these special points are not necessarily robust matching features.

Robust features are the location of intersections of long line segments. The graph consists of several line segments. These line segments intersect each other at several points which may or may not lie on the lines itself. In fact, many of the intersections may be special points in the original image but other intersections may not be part of the original image at all. Our method builds an extended graph from a normal graph. The result is a graph of clusters of related edges that intersect the same point. Now both the angle of the lines and a set of points with related lines are robust features of the graph.

4.2. Image segmentation and regional features

In order to use region-based features, the images need to be segmented. One way to segment images is by using a region growing technique. A standard technique to segment the image is the split and merge algorithm [1].

After the image is subdivided into regions by the split and merge algorithm, regional features need to be defined. The principal question is how to represent color, intensity, and texture for this application. The structure should be robust for small color (camera saturation setting) adjustments and incorporate the knowledge we have about the space application. Textural features are well documented [5]. A relatively cheap and robust technique to describe texture is the use of color and intensity histograms. They do not describe the local spatial correlation but that makes them also robust [1]. We have chosen for this last approach given the types of texture to be expected.

5. Camera parameter determination

The position, orientation and zoom of the cameras are only approximately known because the space station is not entirely rigid due to thermal deformations. The first objective is to estimate the generalized viewpoint of a virtual camera in the 3D model that comes as close as possible to the generalized viewpoint of the real camera. The closer the virtual camera parameters are to the real camera parameters, the easier the matching becomes.

Better camera parameters cause a better match, which in turn causes a better image segmentation and easier object recognition and model verification. The three problems depend on each other. The problem is solved by creating an artificial image with the use of the initial camera parameter estimation. This image is compared with the real image. The mapping of the two images onto each other can be transformed to the camera domain. This way a second, better, artificial image can be produced with the newly found camera parameters.

5.1. Inverse perspective

The first question that needs to be answered is whether it is possible to determine a camera position in the 3D world from a 2D image. This is called the problem of camera calibration and inverse perspective [1]. The solution can be achieved analytically if six independent (u, v) points in the 2D image can be related with six (x, y, z) points in the 3D world. The 3D→2D transform is usually accomplished with the perspective transformation matrix. The elements of the inverse of the perspective transformation matrix can be found by solving 12 equations that can be constructed using the (u, v) and (x, y, z) pairs [1].

There are four parameters that need to be calculated more accurately: field of vision, camera pan, camera tilt, and camera roll. The relationship between camera pan, tilt, and roll and the image domain transformations is used for that purpose.

The problem of determining camera parameters has been reduced to determining an optimal image transformation. The artificial image is *translated*, *magnified*, and *rotated* in order to accomplish an optimal match with the real image. The found translation, magnification and rotation are transformed to the camera domain: pan, tilt, roll, and zoom. The following equations were used to accomplish this transformation, using initial camera parameter estimates (v_h, v_p, v_r) and measured differences in the image domain:

$$\begin{aligned}
 \text{Tilt: } v'_h &= v_h - \arctan \frac{\tan(\text{fov}_u \Delta U / w)}{\cos v_p}, \\
 \text{Pan: } v'_p &= v_p - \frac{\text{fov}_v \Delta V}{h} + \arctan \frac{\sin v_p (1 - \cos \Delta v_h)}{\cos \Delta v_h}, \\
 \text{Roll: } v'_r &= \phi + v_r, \\
 \text{Field of vision: } \text{fov}' &= \text{magnification} \cdot \text{fov}, \\
 \text{where } \phi &= -\alpha + \arctan[(\tan \text{fov}_u \Delta U / w) / \cos v_p],
 \end{aligned} \tag{1}$$

w, h are the image width, height, ΔU is the (measured) difference in horizontal and ΔV the vertical image position, and α is the (measured) rotation of image.

5.2. Mapping two images onto each other

The initial estimated camera parameters are used to produce an artificial image. The artificial image and the real image are best mapped onto each other with the use of line and special point features because they contain positional information. This means that the images need to contain lines otherwise the camera parameter determination module will fail.

The problem of mapping a set of lines and special points that were detected as described in Section 4.1 onto another set can be accomplished in many ways. First we weighted all the lines and special points in order to distinguish between highly likely matches and less likely matches. We used the Euclidean distance, the angular difference and the length as weight factors. Mapping the weighted sets was accomplished with the use of the iterative *simulated annealing* method [7].

The weight parameters and simulated annealing speed of convergence were optimized during the testing phase.

6. Change detection

The result of the camera parameter determination algorithms in Section 5 are two similar images. The comparison of 3D objects is now reduced to the matching of two 2D images, a real image and an artificial image. The purpose is to generate a weighted answer about all points in the images. Each point should be accounted for; does it match or does it not? Both region matching and line matching were used in order to accomplish this.

6.1. Region matching

The artificial image is segmented into regions of different objects. Each object has a known intensity histogram¹ distribution. The real image is segmented into regions as described in Section 4.2.

The overlapping regions of the real and artificial images are compared based on their respective histograms. These histograms cannot be compared directly because the labeling of the objects in the artificial image has not yet taken into account the *intensity shift* of the histogram. This means that although a histogram is characteristic for a certain object, parts of the object may be shaded or covered by a cast shadow. This means that although the shape of the histogram may roughly stay the same, the overall intensity will change.

For each region overlap, the intensity shift in the artificial image can be calculated. The histogram is corrected for this intensity shift, after which the histograms of the two regions can be compared directly. This produces an answer about the presence or absence of an object.

6.2. Line matching

Region matching produces results for all areas in the image where some sort of region can be defined. The borders between regions, or narrow objects which in themselves hardly cover any area, need to be matched in another way. Lines as features in the image are used for this. The matching of lines can be solved in exactly the same way as in the camera parameter determination module. This results in a matching set of lines.

7. Results and conclusions

The system was tested with a mock-up of the space station. The test consisted of two parts: the camera parameter determination module and the matching system.

¹ Testing was done with the use of a black and white camera. A color camera could induce the use of a color histogram.

Table 1
Quantitative camera parameter determination results

Parameters varied	Pan ^a	Tilt ^a	Roll	Fov	Range
1	✓				±17%
1		✓			±17%
1			✓		±15°
1				✓	±16%
3	✓	✓	✓		±15%
All	✓	✓	✓	✓	±14%

^a Pan and tilt are measured in percentages because the resulting image translation depends on pan and tilt relative to the field of vision.

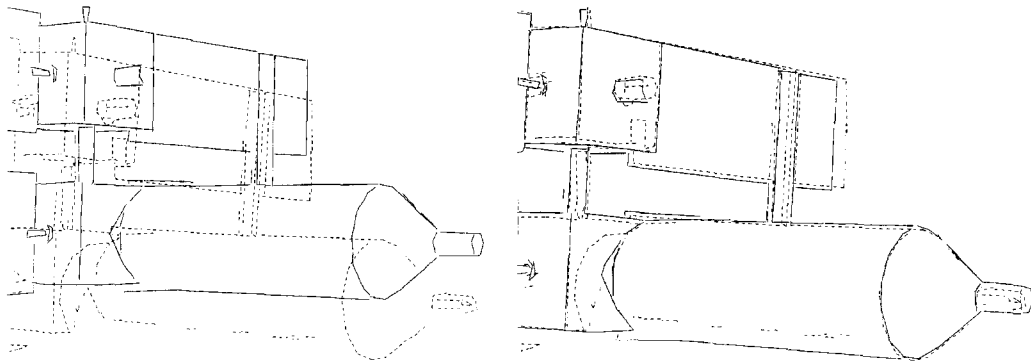


Fig. 3. Line drawing of initial camera parameter settings and final parameter settings after correct determination.

7.1. Camera parameter determination test results

Camera parameter determination is evaluated by the maximal amount of parameter fluctuation that can be detected by the system. The degrees of freedom (pan, tilt, roll, and field of vision) were varied separately and altogether. The tests were done by randomly varying one or more parameters between 0% and 100% of the field of vision.

Table 1 lists the range of allowed parameter fluctuation that always resulted in correct camera parameter determination. This means that an error percentage of camera parameters less than 14% always results in reliable camera parameter determination. For the ISS this is within bounds of possible camera parameter fluctuations (no more than 5%). Fig. 3 shows a typical camera parameter determination module input and result.²

The few tests when the camera parameter determination module did not perform as well occurred if the images did not contain many lines. Further research could investigate the relationship between the quality of the image contents and the performance of the camera parameter determination module. For example, the quality of the image contents can be represented by the amount of long straight lines in the images.

The camera parameter determination module was also tested for reliability if objects were added or deleted from the mock-up of the space station. The results are difficult to represent in a quantitative way because the module depends on image contents. In general the results show that the module performs well if less than 20% of the image was changed.³ For 30% image change the system always fails. For 10% image contents change, the module always

² Real camera parameter determination input consists of a camera image with an approximated rendered image but for reasons of clarity the second step of the camera parameter determination module is shown. The first step transforms the images to line drawings.

³ This means an object change, occupying less than 20% of the field of vision.

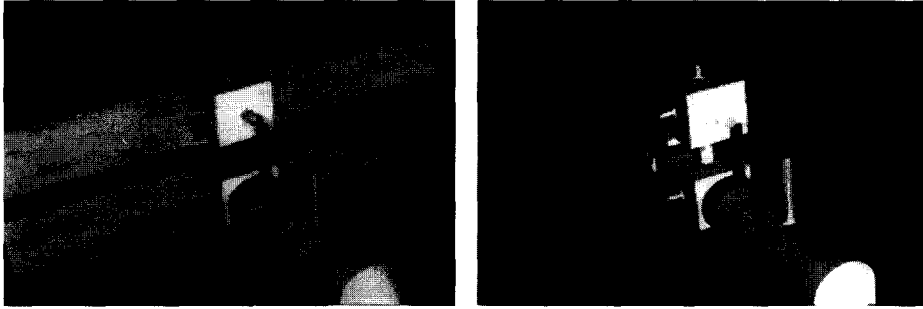


Fig. 4. ISS mock-up (real camera image) and Rayshade image model match without any changes.

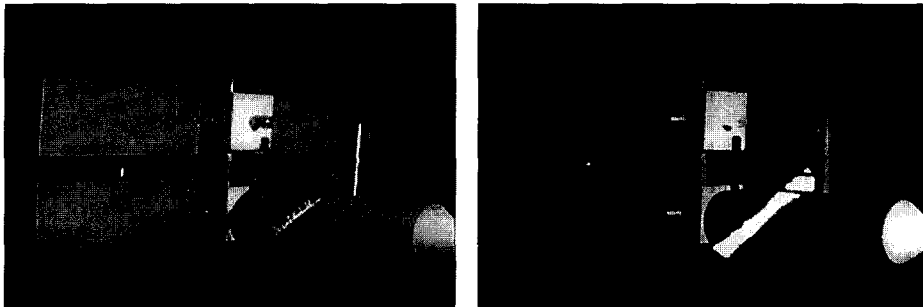


Fig. 5. ISS mock-up with an object addition. Object model shows white color where change was detected (red color in reality).

matches. These are rough estimates; realistic results can only be presented if they can be related in some way to the image contents.

If the camera parameter determination module fails for an object addition or removal, the matching module will try to match an image that cannot be matched. This means that an object change will not pass unnoticed, but when the change is too large, the system cannot qualify the exact location of the change.

Generally speaking, the camera parameter determination module results are very satisfactory.

7.2. Image matching test results

The matching system was tested for object removals and additions, but first the system was tested on how it performed on an unchanged model. The results are as expected: if Performer was used as a graphics engine, cast shadows are detected as object changes. If Rayshade is used as a graphics engine, cast shadows are correctly represented and matched. Reflections cause problems for both renderers but Rayshade shows the least amount of erroneous matches. An example of the test with an unchanged model can be seen in Fig. 4.

Both engines suffer from the fact that the ambient light level is unpredictable because of the use of automatic brightness and contrast settings on the camera. This means that from time to time an area that should have been matched is detected as changed. A less restrictive light intensity setting with the histogram matching results in too many matches when objects are really changed. It is expected that this problem does not occur in real space environment. If the MVS would have to operate on earth, ambient light will have to be modeled.

The sensitivity for addition and deletion of objects depended on the color of the new object in relation to the object (or background) in front of which it was placed (removed). The sensitivity of 3D changes depends of course on the position of the camera with respect to the 3D object. When highly contrasting objects were used, the system was very sensitive: objects as small as 2% of the field of vision could be detected. Less contrasting objects were

less easy to detect. A typical test result can be seen in Fig. 5 where an object was detected that takes approximately 15% of the field of vision.

7.3. Conclusions and recommendations

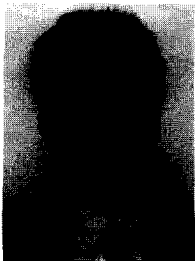
The main conclusion from the test results is that the MVS, as it is functioning now, meets the requirements for the ISS. The camera parameter determination is well within range and performs well with changes up till 20%. With high contrast objects changes as small as 2% can be detected.

Further improvements to the MVS still can be made by using color cameras and further refine the algorithms: incorporating curved line features and more complex texture matching algorithms.

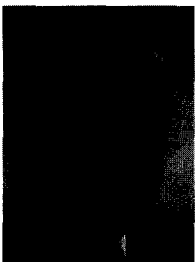
Based upon this system we will extend it in the future to determine the shape of added objects from multiple camera views, to be able to incorporate new objects also in the path planning of the robot.

References

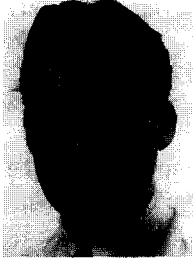
- [1] D.H. Ballard, C.M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [2] S. Castan, SDEF edge extraction algorithm, Khoros online manual, IRIT, 1997.
- [3] S. Fischler, J. Helman, M. Jones, J. Rohlf, A. Schaffer, C. Tanner, *Iris Performer Reference Pages*, Silicon Graphics, 1994.
- [4] F.C.A. Groen, R.J. Ekkers., R. de Vries, *Image processing with personal computers*, *Signal processing* 15 (1988) 279–291.
- [5] R.M. Haralick, L.G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, Reading, MA, 1992.
- [6] C.E. Kolb, *Rayshade User's Guide and Reference Manual*, 1992 (<ftp://graphics.stanford.edu/pub/rayshade/guide.ps.Z>).
- [7] G.S. Singh, K.R. Deshpande, On fast load partitioning by simulated annealing and heuristic algorithms for a general class of problems, *Advances in Engineering Software* (1993) 23–29.
- [8] K. Wall, P.E. Danielson, A fast sequential method for polygonal approximation of digitized curves, *Computer Graphics and Image Processing* 28 (1984) 220–227.



W.F.W. Haak studied Music at Tulane University, New Orleans and attained his bachelor degree in 1989. In 1997 he attained his Masters degree of computer science at the University of Amsterdam. The paper on “Automatic Modelverification of the International Space Station for Path Planning” is based on research at Fokker Space BV for a graduation thesis bearing the same title. Wouter Haak is currently employed by The Boston Consulting Group BV.



Franciscus C.A. Groen received his Ph.D. in 1977 from the Delft University of Technology. Till 1988 he was a staff member of the Pattern Recognition Group of the Department of Applied Physics of the Delft University of Technology. He is professor in the Special Applications of Computer Science at the University of Amsterdam. He stayed in 1984 as a Fullbright Research Scientist at the Robotics Institute of Carnegie Mellon University in Pittsburgh and in 1996 as visiting professor at the University of Utah in Salt Lake City. His research interest is in the field of intelligent autonomous systems and sensor data processing.



Edward Holweg was born in Maassluis on 6 January 1967. After graduating from high school in 1985, he studied electrical engineering at the Delft University of Technology. His M.Sc. thesis was performed at the Control Laboratory and is about the development of a real-time expert system, DICE. He graduated in September 1990. From October 1990 to 1995 he was employed at the Control Laboratory as research assistant and worked for the Teleman-18 project. The project was funded by the E.C., and the objective was the development of a dexterous gripper, able to operate in hazardous and unstructured environments, such as the hot spot of a nuclear power plant, or chemical polluted areas. In November 1996 he received his Ph.D. on the topic Autonomous Control in Dexterous Gripping, based on this work. From March 1996 to April 1998 he worked for Fokker Space and Fokker Control Systems in the field of space robotics and new business development, and finally as project manager medical robotics and mechatronics. Currently he is employed at SKF as projects manager of the mechatronics competence centre.