In: 14th IEEE International Workshop on Robot and Human Interactive Communication,August 13-15, 2005, Nashville, Tennessee USA

Human-robot cooperation: Safe Pick-and-Place Operations

Thorsten Gecks and Dominik Henrich

Lehrstuhl für Angewandte Informatik III (Robotik und Eingebettete Systeme)
Universität Bayreuth, D-95445 Bayreuth, Germany
E-Mail: {thorsten.gecks|dominik.henrich}@uni-bayreuth.de
http: //ai3.inf.uni-bayreuth.de/

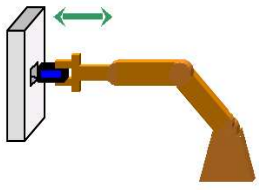
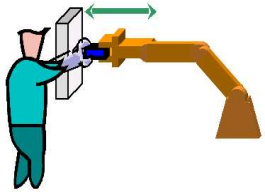
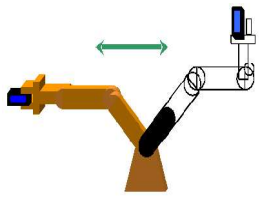
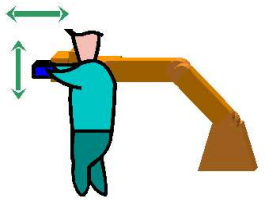
Abstract - **We present an industrial robot system whose workspace is supervised by several stationary cameras to ensure safe human-robot cooperations. All robot transfer motions are checked for collision by detecting obstacles using a difference image method. Whenever a collision is detected, the robot motion path is changed accordingly. The presented algorithm enables robots to perform safe pick-and-place operations exhibiting real time behaviour through efficient image processing.**

Index Terms - human-robot cooperation, industrial robot, workspace supervision, difference image, reference image update, solid modelling

I. Introduction

Humans need a reliable and safe interaction with robots in order to accept and trust the robotic partner at work and even at home. In order to achieve this, our research aims at securing interaction between humans and robots to prevent severe injuries to humans and (costly) collisions with environment obstacles.

TABLE 1:
CLASSIFICATION OF ROBOT OPERATION MODES

	Free Motions	***Guided Motions***
Fine Motions		
Gross Motions		

We currently limit interaction between robots and humans within an industrial environment to free robot movements and force-guided motions. Based on this decision, we distinguish four modes of operation as presented in Table 1. The free transfer movement mode is already implemented as described in [4]. The system is currently being extended to the three remaining modes. This paper presents a first approach to provide safety for free fine motions of the robot. These occur in e.g. pick-and-place tasks.

To safeguard these operation modes, the system needs to detect dynamic obstacles like humans and determine appropriate measures to avoid collisions with the detected obstacles (i.e. path planning), while being able to get in contact with objects to provide useful capabilities, like pick-and-place operations.

To achieve this goal, the acquisition of the current state of the environment is necessary. Cameras are convenient sensors for this task as they are widely available and cost efficient with respect to several criteria such as resolution and update rate. A comparison with other sensor hardware can be found in [4].

In the following sections, we will first give an overview of existing approaches in robot safety systems (Section II). After a short introduction to our safety system, we will describe its application in pick-and-place tasks and the corresponding algorithms in detail (Section III). Subsequently, we will illustrate our results with images of an example pick operation (Section IV) and draw a conclusion (Section V).

II. State of the art

In the past, many approaches have been discussed for sensor-based collision avoidance. However, most of them use sensors for providing local information only. For example, in [12] and [6] capacitance sensors were used as artificial skin. In [9], algorithms for whole-arm collision avoidance for robots with artificial skins were presented. In [15], a wrist-mounted laser scanner was used. [11] presented an approach for image-based path planning in configuration space; however, the use of a wrist-mounted sensor is assumed and the approach requires that the image of the scene in the target configuration is known. With only local sensor information available, only configurations close to the cur-

rent robot configuration can be examined, and thus only local path planning is possible.

There are several techniques to acquire global information about the environment. One widely used technique in computer graphics is to acquire the physical extent of objects by fusing multiple images of a scene with the back projection method, for example in [5]. However, the focus in [5] was the precise reconstruction of an object in 3D space, including texture information, and this is not necessary for our problem, because the object can be coarsely reconstructed and there is no need for knowing the object texture, as only the physical extents of obstacles are concerned for safety issues. Furthermore, with this technique only single objects would be reconstructed, while in our problem multiple obstacles exist in the scene.

[10] and [1] presented a system for obstacle detection with stationary CCD cameras. These cameras were used to establish multiple passive light barriers, a concept related to back-projection. Evenly distributed points in the robot workspace are mapped to a monitored pixel in each camera. If the features of the pixel differ from the given background values beyond a certain threshold within each camera, the system assumes the corresponding point in space being occupied by an obstacle. If such a point exists within the robots path, the current robot movement would be suspended. Path planning based on the obstacle information was not implemented.

The MEPHISTO system [14] allows for robot-human coexistence in the field of mobile robots. It combines laser scanners mounted on the robots and a global monitoring system consisting of color cameras surveying the floor upon which robots and humans move. The images obtained by the cameras are compared to a reference image that is continuously updated. The difference image is mapped on the floor in the form of a polygonal region, which allows fast collision detection. The system provides the observed mobile robots with a path planning service in a 3-dimensional configuration space (2D-position and rotation around axis perpendicular to the ground plane).

Our goal is to develop a system capable of global path planning and obstacle detection with industrial robots. It should be able to perform basic tasks like pick-and-place operations. Therefore we employ cameras as global sensors, because local sensors fail to provide safe object transporting capabilities. For gripped objects above a certain size it is nearly impossible to measure distances from the surface of the object to the environment, because it is not practical to place short range distance sensors on the object. But if they are placed on the robots body, the object would possibly obstruct the line of sight from the local sensor to the environment. This results in possible collisions of the object with the environment, as not all distances from the object to the environment are known.

Comparing to [10] and the back projection method, we use a difference image method by combining several cameras to detect obstacles within the workspace. Here, the difference image based obstacle detection is extended to support pick-and-place operations. Based on the difference images, the system plans alternative paths around detected obstacles. This concept seems to be a promising technique for the realization of safe robot workspaces.

III. System Concept

We now briefly present the concept of our prototype system of a camera supervised workspace, beginning with a hardware description and software overview and develop the system data flow sequentially (Section A). Afterwards we will describe how the system allows safe pick-and-place tasks (Sections B to D).

A. System Overview

The workspace of the robot is monitored by several stationary cameras, each one monitoring the entire workspace shared by humans and the robot. Additionally, the current robot position is acquired by angle encoders in the joints of the robot. The system computes a path around obstacles such as humans. To detect the obstacles within the workspace, the image processing subsystem applies a difference image method. It detects humans and other obstacles via comparison of the current images to reference workspace images [2]. Generally the *reference images* are composed of static environment objects only, not containing the robot and human operators.

These reference images are generated in a preceeding setup step. The reference image creation is done automatically, it only requires a workspace free from obstacles. The robot is driven into various positions and a picture is captured for each one. Then, a median value is calculated for every single pixel in all captured images for each camera. The robot arm is erased from the image, because every pixel is supposed to be part of the robot body only in one image for all robot positions (except the robot base, which is immovable for the shown robot type) [4].

For calculating a difference image of the workspace, the currently captured gray scale images are evenly subdivided into non-overlapping tiles on a grid, such that each tile contains several pixels. This subdivision also applies to the corresponding reference image. Each tile corresponds to a pixel in the difference image, which thus is of lower resolution. For each tile, features are calculated based on the given pixel values (Fig. 1 a, b). These features are then compared to the corresponding features of the reference images (Fig. 1 c, d). The classification yields two classes: The tile is classified as *Background* if no significant changes to the reference image exist and as *Foreground* if significant changes exist. Several classification methods and automatic classification parameter optimization strategies can be applied here [8]. The classification result of the tile determines the intensity of the corresponding difference image pixel (Fig. 1 e).

The prerequisite for the collision test is to eliminate the robot from the current set of foreground pixels. Therefore, the system computes a robot model with the current robot configuration given and projects it into the camera views by the use of a calibrated camera model (Fig. 1 f, g, h, i). These

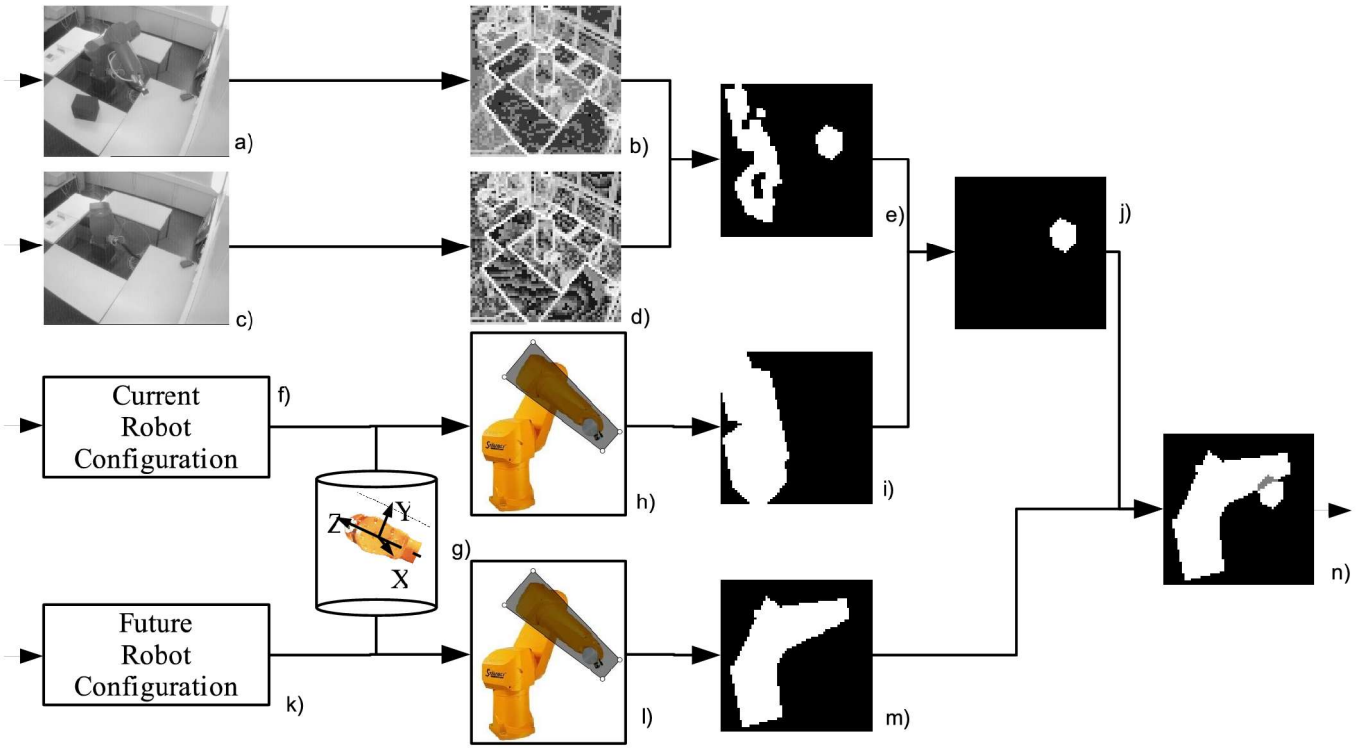


Fig. 1: Flow chart of the collision detection system with simplified computation steps (no iterations shown)

robot views are used to mask the robot in its current position and detect possible objects within the robot projection area using epipolar lines and information from other cameras [3]. Concurrently, only foreground pixels belonging to (possible) obstacles remain for consideration in the path planning process (Fig. 1 j). This image is called *cleaned workspace image*.

The path planning takes place in a discrete subspace of the robot configuration space. Path planning algorithms operating on this discrete space check future robot configurations for collisions with obstacles. The future robot configurations are therefore projected into the camera views with the help of the robot model and checked for intersection with the set of foreground pixels from (possible obstacles) from the cleaned workspace image (Fig. 1 k, g, l, m, n). Based on the fused intersection information from all cameras, the current robot position can be determined as collision-free or not.

B. Providing pick-and-place operations

The assumption of the classical difference image method using a static reference image renders the system useless for many applications. Typical industrial robot tasks involve manipulations of environment and work pieces in a way that changes the robot cell. These changes usually appear as foreground pixels since they create a difference in the current image compared to the reference image. They clutter the workspace over time and would drive the robot into immobility. Nevertheless, these changes are part of the application process, and therefore should not be detected as an obstacle. To keep the difference image method as an efficient image processing algorithm, we need to make certain extensions to enable the adaptation of workcell modifications which are part of the process.

In the following, we want to analyze the environmental changes within a workcell in regard to the realization of the safety system. At first, changes as such need a time base and in the case of our safety system, this time base is discrete; it is based on the frame rate of the deployed camera system. Secondly, changes of the physical world are only visible to the safety system if they occur within the internal representations of the environment, of which there exist two at the moment, namely a model-based CAD-representation and a sensor-based image representation. Based on the discrete time base, we want to distinguish the changes of both representations into three types (none, discrete and continuous), as presented in Table 2, and motivate these types in the following.

Table 2 ignores the environment changes due to only the robot moving. Only environment changes due to the robot moving obstacles are regarded. Otherwise, the moving robot would induce continuous changes in both environment representations (based on the definition as given in the following) and thus would render the first two types superfluous.

Regarding the image-based environment representation the changes refer to pixel color or greyscale value changes. Cases for change category *none* include transfer motions of the robot or pick-and-place tasks with objects being too

small to be detected by the camera system (for example in electronics assembly).

Discrete changes refer to pixel values that change from one image frame to the subsequent and then stay stable for a certain number of frames. Typical occurrences of this type of changes are pick-and-place operations of objects of significant size, where for example an object placed on a tray changes the pixel values, if its view is different from the background. Discrete changes provide an opportunity for reference image update algorithms, which sustain the usability of the simple difference image algorithm for the subsequent frames. The algorithm described in this paper performs such a reference image update.

TABLE 2:
TYPES OF REPRESENTATION CHANGE AND UPDATE APPROACHES FOR DIFFERENT ENVIRONMENT MODELS (BOTH REPRESENTATIONS EXCLUDE THE ROBOT PART OF THE ENVIRONMENT)

Update approaches		Type of representation change		
		None	Discrete	Continuous
Environment Representation	Model-based (CAD)	Static Environment model	Discrete model adaptation	Continuous model adaptation
	Sensor-based (Image)	Difference image method	Reference Image Update	For Further research

Continuous pixel value changes occur permanently from image frame to frame. A typical workcell situation that induces this type of change is conveyor belts. This prevents the use of a reference image update algorithm, because not only the reference image has to be updated from frame to frame, but also the most recent reference image would be useless in the subsequent frame as a base for comparison. So this change type needs special treatment, which belongs to further research and is not covered by this paper.

Concerning the model-based representation, discrete and continuous changes of model parameters are distinguished as described for the sensor-based representation above based on the discrete time. They can occur also in the example tasks mentioned, depending on the respective physical environment change is part of the model-based or image-based representation.

C. *Reference Image Update Algorithm*

Pick-and-place tasks impose the problem of discrete image changes. The approach described here realizes a reference image update (a summary of reference image update algorithms is given in [13]).

The reference image update algorithm deals with a pixel set U of difference image pixels that define the area of changed pixel values (Remember, that a difference image pixel is correlated to a distinct tile within a grayscale image, as described in Section A). The classification of the pixels within the grayscale image tile determines the value of the difference image pixel. Consequently, a difference image pixel belonging to the update area pixel set U defines the tile of the reference image that needs to be replaced with the corresponding tile in the current camera image to produce an updated reference image.

The presented reference image update algorithm for robot pick-and-place operations relies on several informations: the point in time for the start of the update process, the position and shape of the initial update area described by the pixel set U, the robot model, and obstacle information.

The time information t_0 is supplied by the robot program. In detail, this means that the program sends out a signal, when the robot picks or places an object. This signal contains information about the type of object as well.

The actual shape of the update area is provided by an object model. This object model exists, because the robot model needs to extended by the object model for save transfer movements with the respective object gripped [7].

The shape of the area described by the pixel set U is determined by the robot's tool coordinate system at the time the object change occured. This requires rigid objects, of which shapes are unchanged and well-defined by the tool coordinate system.

With these preconditions, it is possible to determine the current pixel set, that need to be integrated into the reference image. Formulated in pseudo-code, the update process works as follows:

$U := computeObjectPixels(t_0)$
repeat
  $R := computeRobotModelPixels(t_0)$
  $E := U \setminus R$
  $updateReferenceImageTiles(E)$
  $U := U \setminus E$
until $U = \emptyset$

First of all, an image of the object model at the robot position at time t_0 is calculated, which determines the pixel set U. In each iteration step of the system, the robot model provides the current pixel set R defining the robot's shape. This set determines the currently occluded update area and thus by substraction the set of actually to be updated pixels E results. This set determines the updated reference image pixels and, by set substraction, the new update area pixel set U for the next iteration describing the part of the original set U, that was not updated so far. The iteration continues until the pixel set U is empty.

Fig. 2 explains two different situations schematically: the update process in case of a pick operation (Fig. 2a) and a place operation (Fig. 2b). The algorithm used is the same for both cases, as the robot model is adapted to the respective situation. As depicted in Fig. 2a, the current robot model in case of a pick task is quite more extended compared to the place task in Fig. 2b. It thus usually generates a smaller number updated pixels in the first steps of the update process.

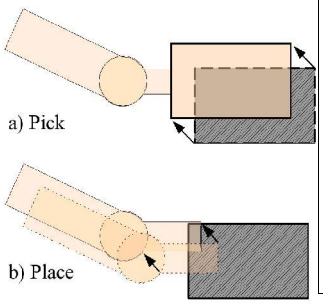


Fig. 2: Schematic update procedure (the remaining update pixel set E is hatched)

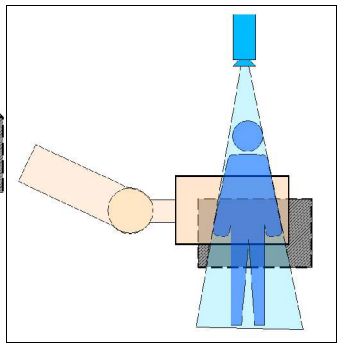


Fig. 3: Obstacle detection in front of the update area in the camera view with the help of obstacle information from another camera and epipolar lines (The pixel set E is hatched)

D. *Occlusion through obstacles*

The update process must prevent a foreground object (obstacle) from becoming integrated into the reference image. In the event of such an accidental integration, two cases can be distinguished: The object stays at the position it had been during update process or it moves away. In the first case, the object generates no foreground pixels in the difference image, which can result in robot-object collisions. In the second case the removed object generates foreground pixels, as there occurs an image difference. This in turn creates another practical problem: the space seems cluttered with obstacles no longer existing. This can cause robot immobility.

The major concern is thus to have the update area obstacle-free upon reference image update operations. The maximum possible update area comprises the objects shape. From this area we have to substract the area occluded by the robot (as described in Section C) and detected obstacles.

$U := computeObjectPixels(t_0)$
repeat
  $R := computeRobotModelPixels(t_0)$
  $E := U \setminus R$
  $O := computePossibleObstaclePixels(t_0)$
  $E := E \setminus O$
  $updateReferenceImageTiles(E)$
  $U := U \setminus E$
until $U = \emptyset$

A problem is, that no obstacle can be directly observed based on image differences in the update area, as image differences already occur in this area because of the pick/place process. Based on an epipolar line algorithm described in [3], we can detect the set of possible obstacle pixels O within the update area of one camera based on obstacle information from other cameras views (Fig. 3). Treating these possible obstacles as real and removing their pixels from the update area pixel set, we can safely update the remaining pixels of the update area, which results into the refined update algorithm described above.

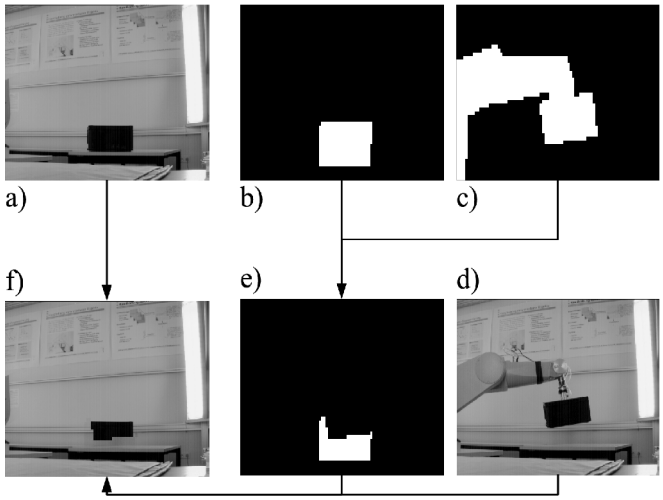


Fig. 4: Flow chart of a step of the reference image update algorithm with images of an example scene where the robot performs a pick operation

IV. Example Pick Operation

A. *Update sequence*

Fig. 4 illustrates a data flow diagram for an update step during a pick operation. The sequence is taken from an example run at our laboratory with the robot picking and placing a plastic box.

Fig. 4a represents the reference image to be updated. Fig. 4c shows the robot model image (including the currently picked object) and Fig. 4b the update area derived from the object model image at the pickup position. Subtracting the robot model image from the update area, the remaining update area (Fig. 4e) results, representing the pixels to be updated. The current reference image (Fig. 4a) is then partially replaced by the current camera image (Fig. 4d), as defined by the remaining update area, to result into the new reference image (Fig. 4f).

B. *Prototype system performance*

Detailed figures for the system are given in [7]. The supervised workspace spans about 2 meters in each dimension, as indicated in Figure 5. It can be scaled by changing the number of cameras. The robot executes a simple pick-and-place task with a box. Human operators may walk through the workspace causing the robot to stop or to plan an alternative path to its target. The system is performing quite well depending on a parameter that determines how many cameras have to detect a collision-free path before the robot moves. If the number of cameras is too small, the robot will sometimes try a move through an obstacle. On the other hand, if the number is too high, the robot often stops in the vicinity of obstacles in spite of a narrow, but free path to the target. This last solution is safe, but less efficient in terms of

process cycle time. The parameter is clearly dependent on the overall number of cameras and their positioning, an issue to be addressed in future research.

Fig. 5: Prototype system. Work cell with robot and human worker carrying a box. The cameras are indicated by icons (the bottom right one is hidden). The supervised workspace is indicated with a dashed transparent cube.

C. *Robustness*

The robustness of the system depends on the robustness of the difference image method, as the follow-up data processing assumes only a binary valued inputs (foreground/background). The robustness of the difference image method was examined in experiments done[8]. The implemented algorithm achieves up to 97 % detection of typical obstacles. With objects being very similar to the background the algorithm of course fails and this results in human-robot collisions. Typically the obstacle does not vanish in all cameras attached to the system as the background differs among cameras, so that a threshold parameter determining how many cameras have to detect the obstacle at least can provide more safety but less mobility of the arm and vice-versa.

V. Conclusions

We presented the implemented supervision system that enables safe human and robot coexistence in a shared workspace. The system provides safe pick-and-place capabilities that can be part of standard industrial processes. The implemented reference image update algorithm exhibits robust and usable behaviour and allows the use of a computationally inexpensive difference image algorithm to provide real time behaviour. Further research needs to be done to cope with continuously changing backgrounds, so as to qualify the system for further types of industrial processes.

References

[1] Ameling W. (Ed.): „Flexible Handhabungsgeräte im Maschinenbau“; Ergebnisse aus dem Sonderforschungsbereich 208, VCH Publishing, 1996

[2] Ebert D., Henrich D.: „Safe Human-Robot-Cooperation:Problem Analysis,System Concept and Fast Sensor Fusion" In: IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems, pp. 239-244, Baden-Baden, Germany, August 20 - 22, 2001

[3] Ebert D., Henrich D.: „Safe Human-Robot-Cooperation: Image-based collision detection for Industrial Robots" In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1826-1831, Lausanne, October 2 - 4, 2002

[4] Ebert D.: “Bildbasierte Erzeugung kollisionsfreier Transferbewegungen für Industrieroboter“ PhD Thesis, Informatics Faculty, University of Kaiserslautern, Germany, 2003

[5] Gerald E.: “Automatic Shape Reconstruction of Rigid 3-D Objects from Multiple Calibrated Images”, In: Eusipco 2000 Proceedings, Tampere, Finland , 2000.

[6] Feddema J.T., Novak J.L.: “Whole Arm Obstacle Avoidance for Teleoperated Robots”. In: IEEE Robotics and Automation Proceedings, pp.3303 – 3309, 1994.

[7] Gecks T., Henrich D.: „SIMERO: Camera Supervised Workspace for Service Robots“, 2nd Workshop on Advances in Service Robotics, Fraunhofer IPA, Stuttgart, Germany, 20-21 May 2004

[8] Heinzen F.: „SIMERO – Robuste und Schnelle Erzeugung von Silhouetten aus Grauwertbildern“, Diploma Thesis, Informatics Faculty, University of Kaiserslautern, Germany, 2003

[9] Lumelsky V., Cheung E.: “Real-Time Collision Avoidance in Teleoperated Whole-Sensitive Robot Arm Manipulators”. In: IEEE Transactions on Systems, Man and Cybernetics, Vol.23 No.1, pp.194-203,1993.

[10] Meisel A.; Föhr R.; Ameling W.:“3D-Kollisionsschutzsensor auf der Basis von CCD-Kameras“, In SENSOR 91, pp. 157-170, Nürnberg, May 1991

[11] Noborio H., Nishino Y.: “Image-based Path-Planning Algorithm on the Joint Space”. In: IEEE International Conference on Robotics and Automation, pp. 1180-1187,Seoul, 2001.

[12] Novak J.L., Feddema J.T.: ”A Capacitance-Based Proximity Sensor for Whole Arm Obstacle Avoidance”. In: IEEE Proceedings of the Intl. Conf. on Robotics and Automation, pp. 1307-1314, 1992.

[13] Radke R. J., Andra S., Al-Kofahi O., Roysam B.: „Image Change Detection Algorithms: A Systematic Survey“, IEEE Transactions on Image Processing, 2004

[14] Steinhaus P., Ehrenmann M., Dillmann R.: MEPHISTO: A Modular and Existensible Path Planning System Using Observation. ICVS 1999 361-375

[15] Yu Y., Gupta K.: „Sensor-Based Roadmaps for Motion-Planning for Articulated Robots in Unknown Environment: Some Experiments with an Eye-in-hand System“. In: IEEE International Conference on Intelligent Robots and Systems, pp.1707-1714, 1999.