In: German Workshop on Robotics – GWR 2009, June 9-10, Braunschweig/Germany

3D Collision Detection for Industrial Robots and Unknown Obstacles using Multiple Depth Images

Markus Fischer and Dominik Henrich

Lehrstuhl für Angewandte Informatik III, Universität Bayreuth, Germany
{markus.fischer, dominik.henrich}@uni-bayreuth.de, http://www.ai3.uni-bayreuth.de

Abstract In current industrial applications without sensor surveillance, the robot workcell needs to be rather static. If the environment of the robot changes in an unplanned manner, e. g. a human enters the workcell and crosses the trajectory, a collision could result. Current research aims at relaxing the separation of robot and human workspaces. We present the first approach that uses multiple 3D depth images for fast collision detection of multiple unknown objects. The depth sensors are placed around the workcell to observe a common surveilled 3D space. The acquired depth images are used to calculate a conservative approximation of all detected obstacles within the surveilled space. Using a robot model and a segment of its future trajectory, these configurations can be checked for collisions with all detected obstacles. If no collision is detected, the minimum distance to any obstacle may be used to limit the maximum velocity. The approach is applicable to a variety of other applications, such as surveillance of tool engines or museum displays.

Introduction

In current industrial applications, human and robot workspaces are completely separate and it is forbidden for them to share a common workspace. When a human enters the robot workcell, additional safety systems must ensure the safety of the human. There are many applications for coexistence of robot and human. Robots would require less space if no physical barriers are needed to isolate them and a technician could perform maintenance of other machines in the workcell. The first commercial product using image processing sensors is available [Pilz06]. This product checks for intrusion in virtual safety zones around robots and dangerous equipment.

As the main objective of such surveillance, collisions between the robot and any obstacle must be avoided, as these could lead to severe injuries if the robot

collides with a human. The second objective is velocity control based on the proximity of the robot to any obstacle. By limiting its current maximum velocity, the robot can always be stopped before it encounters an obstacle.

The approach introduced here is the first that aims to use multiple depth sensors for online collision detection. Since depth sensors already provide volumetric data, no correspondence problem has to be solved, as is the case with raw data from stereo cameras. The sensors are placed around the workcell and observe a common space called the *surveilled space*. In each collision detection cycle, the sensors acquire new, synchronised depth images. An approximate and conservative representation of all objects within the surveilled space is computed using these images. With a geometric (CAD) model of the robot and the current robot configuration, it is possible to identify the robot as one of the objects, and the remaining objects are classified as obstacles for the robot. For a given segment of the future trajectory, the robot model may then be adjusted and checked for collisions with the obstacles. One way to avoid the potential collision is to decelerate and stop the robot [Som05]. It may then proceed with its trajectory if no collision is detected in succeeding collision detection cycles. Another way to avoid the potential collision is to calculate a new trajectory for the robot [Gecks07]. As long as no collision is detected, the proximity can be used to limit the maximum velocity [Kuhn06, Kuhn07].

In summary, the contribution of this paper is a fast reconstruction of the entire robot environment, including multiple unknown objects, of multiple depth images for use in an online surveillance system.

State of the art

This work is related to reconstruction in computer graphics, to human detection in computer vision, and to surveillance in safety systems.

In computer graphics, the visual hull concept [Matusik01] uses object silhouettes in various colour images. The silhouettes define cones containing the object and the object volume is defined by the intersection of these cones. In [Li02], a stereo algorithm is used to improve the volumetric approximation of the visual hull. The presented system calculates 4 frames per second *fps* for a scene that contains a single object.

A mesh-based CAD model of an object based on multiple depth images provided by laser scans from different perspectives is presented in [Reed99]. Merging these images results in a precise CAD mesh model. Constructing the mesh and merging each perspective at a resolution of 110x128 *pixels* requires about 6 *min*.

These algorithms target reconstruction of single objects in empty space and are not suitable for fast fusion of multiple depth images containing multiple objects to obtain polyhedral representations.

In computer vision, several approaches for human detection are presented [Hu04]. Most of those approaches use models for object recognition, but object models limit the capability to detect arbitrary unknown objects.

In safety systems, existing strategies provide 1½D, 2½D or 3D surveillance, depending on the type and number of sensors used.

In [Som05] a single laser range finder acquires 1½D distance information within a plane just above the floor of the workcell. Any dynamic obstacles detected are assumed to be standing humans and are approximated by a vertical cylinder. The smallest distance between this cylinder and the robot limits the maximum robot velocity. However, other obstacles, humans in a stooping posture or humans behind the detected individual may not be correctly approximated.

In [Thiemermann03, Thiemermann05] a tri-ocular camera system acquires colour images of the shared workspace from above. The human's hands and neck are detected based on the characteristic colour and texture features of the skin. The minimum distance between human body parts and the robot is used to limit the maximum robot velocity. This system was extended with a tri-ocular stereo vision system acquiring 2½D information from above. It remains unclear whether this method can cope with larger work pieces or obstacles other than exposed human body parts.

[Pilz06] calculates 2½D data about the surveilled space from a single stereo image and checks for intrusions into predefined safety zones. Entry into such a safety zone leads to deceleration or stoppage of the robot. During active surveillance, these zones are static and cannot be changed.

[Winkler07] presents an approach for 2½D surveillance of a robot workspace with a single time-of-flight camera. A robot model is calculated and suppressed in the depth image and a safety zone for each robot link is defined and checked for intersection with the depth image. A collision detection cycle is about 200 *ms*, with additional time necessary for acquiring the depth images.

All of the above systems use multiple colour cameras or a single depth sensor, but none of them involve full 3D surveillance by means of depth sensors.

This work extends now the SIMERO system [Ebert03, Henrich08], which provides global surveillance of a robot workcell with multiple colour cameras and collision detection in 3D. By means of pixel classification [Gecks06], dynamic unknown objects are distinguished from the workspace background. Since no depth information can be obtained from colour images, only deviations from reference images can be used to detect dynamic unknown objects.

Problem Description

In this section, the theoretical background of the introduced surveillance approach is described.

The *visible volume V* of a sensor is defined as part of the space within the sensor's aperture angles. Here, multiple depth sensors observe a common volume of the three-dimensional workspace (Fig. 1a). The intersection of their *V*s is called the *surveilled space S*. All collision detection results obtained are respective to *S*.

Each *image pixel* (x,y) describes a straight line $l_{x,y}$ from the *focal point f* of the sensor into *V*. This $l_{x,y}$ is provided by the sensor calibration. In every *time step t*, the depth sensor measures the distance to the nearest obstacle along $l_{x,y}$. The *point* $p_{x,y}(t)$ is defined as the first intersection of $l_{x,y}$ with this obstacle's surface. For each *t*, the depth image is a matrix of all distance values of the (x,y). Adjacent pixels in both image directions ($l_{x,y}$, $l_{x+1,y}$, $l_{x,y+1}$, $l_{x+1,y+1}$) define a *sub-volume* $V_{x,y}$ of *V*. The *mesh* $m_{x,y}(t)$ is a triangulated surface with $p_{x,y}(t)$ as vertices within $V_{x,y}$. Assuming that the angle between adjacent rays is small enough, none of the surface within $V_{x,y}$ will differ significantly from the real object's surface. $m_{x,y}(t)$ divides $V_{x,y}$ into two parts. All points in $V_{x,y}$ sharing a side with *f* are empty space $E_{x,y}(t)$, all other points in $V_{x,y}$ are occluded space $O_{x,y}(t)$. The union $m(t)$ of all $m_{x,y}(t)$ represents the measured surface within *V* and divides *V* into $E(t)$ and $O(t)$ (Fig. 1c). *V* is clipped to the volume of *S* (Fig. 1d). Adjacent portions of $O(t)$ within *S* are classified as detected objects $o_j(t)$ with $O(t) = o_0(t) + \ldots + o_l(t)$ (Fig. 1f).

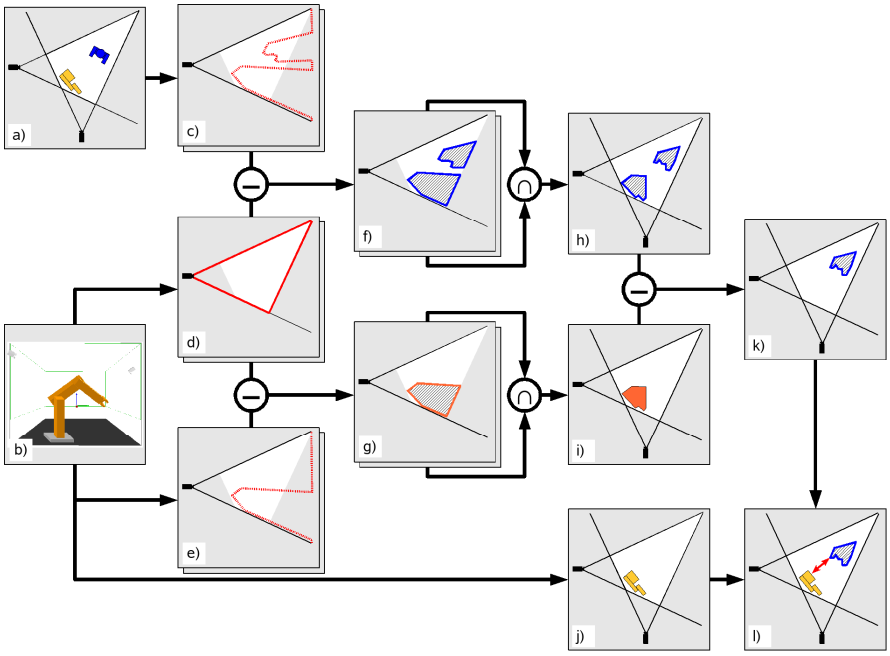

Fig. 1 Overview of the complete surveillance cycle viewed from above. Using the current depth images (c) and data about the robot (b), the minimum distance between the robot and any detected unknown object can be calculated (l).

These definitions must be extended for a multi-sensor system. The index i of the current sensor is added to the parameters of a set as a superscript, e.g. V^i as V of the i^{th} sensor. The space surveilled by multiple depth sensors is determined by intersecting the V^i:

$$S = \bigcap_i V^i = ([\bigcup_i E^i(t)] \cap [\bigcap_i V^i]) \cup [\bigcap_i O^i(t)] \quad (1)$$

This is a lossless representation of the complete content of S. While the union of all $E^i(t)$ leads to a conservative approximation of empty space, the intersection of all $O^i(t)$ produces a conservative approximation of present objects (Fig. 1h).

Known objects should not be checked for collision with the *safety zones* $Z(t) = z_0(t) + \ldots + z_l(t)$, but are also detected in the depth images. A model database provides geometric data for known objects, e. g. a robot, and safety zones, e. g. an expanded robot model, at any time (Fig. 1b). Virtual depth images of an empty space containing only known objects (Fig. 1e) produce virtual detected objects $k_m(t)$ with $K(t) = k_0(t) + \ldots + k_l(t)$ (Fig. 1g). Their intersection (Fig. 1i) is a subset of $O(t)$ from real depth images and results in the *detected unknown obstacles* $O(t) \backslash K(t)$ within S (Fig. 1k). For each t, the $Z(t)$ are defined (Fig. 1j) and the intersection of $Z(t)$ with $O(t) \backslash K(t)$ results in the *detected collisions* (Fig. 1l):

$$C(t) = Z(t) \cap (O(t) / K(t)) \quad (2)$$

Fast Collision Detection Algorithm

Initially, the algorithm contains the following assumptions: (a) The algorithm is initialised with a user-determined number of depth sensors. (b) All of these sensors are calibrated. (c) The CAD model of the robot and its complete trajectory are known. (d) The number of configurations to be checked for collisions is user-determined. (e) S may include multiple unknown objects and additional objects may enter or leave S at any time. (f) The position, geometry and orientation of these objects may change over time.

The presented algorithm implements the collision detection defined in the previous section and consists of three computational steps:

A. Calculate the $o_j(t)$ within S by the fusion of all current depth images.
B. Calculate the $k_m(t)$ within S to suppress them in the collision detection.
C. Calculate the $z_n(t)$ and their minimum distance to any unknown obstacle.

The first two steps must be done once per surveillance cycle. The last step must be done once for each future robot configuration to be checked for collision.

A. Fusion of Depth Image Data

The first step is to calculate the objects in each depth image. The algorithm uses the bounding box of S within the workspace as the clipping volume for V^i. This is a conservative approximation of the portion of V^i within S. Adjacent portions of the $m^i(t)$ within this bounding box are segmented. The $o^i_j(t)$ are the result of a convex hull computation [Barber96] for all segmented parts of $m^i(t)$. The convex hull is a conservative approximation of these detected objects. In computational geometry, efficient algorithms for convex hulls are presented. Since real sensor values include error pixels with incorrect distance values, one characteristic of the convex hull is very useful; if a single point is measured behind the real surface, it is "absorbed", since it lies within the convex hull. If a single point is measured above the real surface, the convex hull is extended by this point. This is a conservative approximation of the convex hull of the real surface.

To get the space occupied by all $o_j(t)$, the $o^i_j(t)$ of all depth images have to be intersected. Since this is computationally expensive and most intersections of $o^i_j(t)$ from different real objects results in empty volumes, only the intersection of their bounding boxes is calculated initially and is then iterated for all cameras. The result of this computation is a list of bounding boxes for all possible objects. The detected objects within the non-empty bounding boxes are then intersected, resulting in the $o_j(t)$.

A non-empty intersection of $o^i_j(t)$ from different real objects can produce a virtual object with no correspondence in reality (Fig. 2). This virtual object is denoted a *pseudo-object*. Since only depth information is used to calculate the occupied space, this object is indistinguishable from a real object in S. No depth sensor has information about this portion of S. The pseudo-object may contain no or one or more real objects. For a conservative approximation in S, pseudo-objects [Meisel94] have to be treated as real objects.

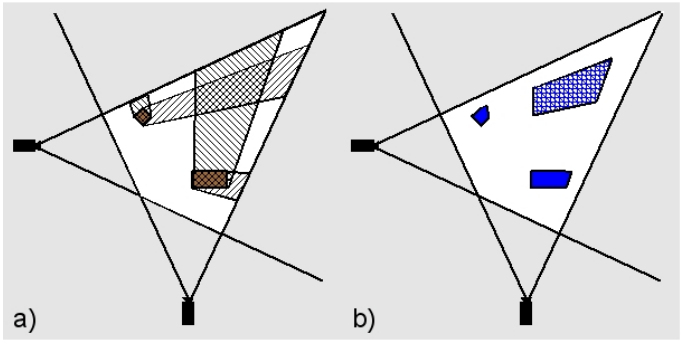

Fig. 2 Illustration of a pseudo-object calculation. S (white area) contains two real objects. Both depth sensors detect two objects $o^i_j(t)$ (a). The intersection of the $o^i_j(t)$ (b) results in real objects (blue) and a pseudo-object (blue hatched) that does not necessarily represent a real object.

B. Space Occupation of Known Objects

Since the number of sensors is limited, they cannot measure the entire surface of all known objects. Given the limited number of perspectives, the known objects must be approximated and some areas of S adjacent to known objects cannot be seen by any sensor and therefore produce obstacles during collision detection. Thus, virtual depth images containing only the $o^i_j(t)$ of known objects are generated and their intersections $k_m(t)$ are used instead of the real known object geometry. In the current implementation, this is done for the robot model and the current robot configuration. Additional known dynamic objects in S such as conveyor belts could be included in the same way, too.

C. Safety Zone Calculation and Collision Detection

The polyhedrons calculated in step A and B are used to calculate the minimum distance between the $z_n(t)$ and the detected unknown objects within S. A minimum distance of zero indicates a collision.

For the robot configuration to be checked for collisions, the volume occupied by the robot model is calculated. If objects other than the robot itself have to be checked for collisions with detected unknown objects, additional $z_n(t)$ can be defined. As with the robot model, these $z_n(t)$ may change their geometry in each new collision detection algorithm cycle, but in the current implementation, only the robot itself is used as a safety zone.

The detected unknown objects are the subset of detected objects excluding known objects. However, the difference operator results in non-convex polyhedrons. An intersection volume of $z_n(t)$ and $o_j(t)$ outside the $k_m(t)$ indicates a detected collision $C(t) \neq \emptyset$.

The collision detection step must iterate over all $z_n(t)$ and $o_j(t)$. The loop can be stopped if a collision is detected or continued with the next iteration if the distance [Cameron97] is greater than zero. If the distance between $z_n(t)$ and $o_j(t)$ is zero, there is an intersection and its volume is calculated by intersecting convex polyhedrons. For all $k_m(t)$, this intersection volume is checked for complete inclusion. Since the intersection volume and all $k_m(t)$ are convex, it is sufficient to check all vertices of the intersection volume for inclusion in the $k_m(t)$. There are three possible results: All vertices are included, none of them are included, or there are vertices both inside and outside of $k_m(t)$. If all vertices are included in $k_m(t)$, the volume is within the known objects and is assumed not to be an unknown obstacle and no collision is detected. Otherwise, the intersection volume has to be a detected collision, the minimum distance is set to zero and the loop is stopped. The minimum distance and the indices of $z_n(t)$ and $o_j(t)$ are then reported to the control program.

Experimental Results

The presented algorithm was tested in both a simulation system and an experimental setup.

Simulation results

The simulation system is used with a virtual robot workcell. Its geometry is user-determined and was set during testing to the size of the robot workcell used for the prototype experimental setup. The simulated workcell includes a stationary robot model and additional defined objects such as humans or tables, which are all represented as polyhedrons. Up to six unknown objects were used during experimentation. A pinhole model is used to simulate the depth sensors and their number, position, orientation, resolution and aperture angles are user-determined. Up to 8 depth sensors were simulated in an experiment.

There is no structural difference between simulated and real depth images. The only differences are due to noise in measurements made with real sensors. With these simulated depth images, the presented surveillance algorithm can be applied.

The system was implemented in C++, compiled with gcc version 4.1.0 and optimised with the –O3 flag running on SUSE Linux 11 (i586). A standard CPU (AMD Sempron 3000+, ~ 2GHz, 512 KB Cache) and 512 MB RAM were used.

The average computation time for a sequence of 1500 frames containing the simulated workcell and an increasing number of objects within *S* and an increasing number of implemented depth sensors was calculated (Fig. 3).

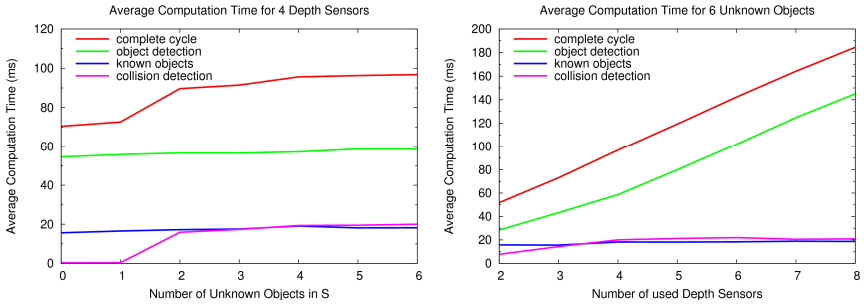

Fig. 3 Diagrams showing the average computation time for calculating a sequence of 1500 collision detection cycles. The algorithm was tested for an increasing number of unknown objects in *S* (left) and for an increasing number of depth sensors (right).

As shown in Fig. 3 left, the number of objects within *S* has little influence on the surveillance cycle time. The object detection step calculation time directly depends on the number of objects detected by each depth sensor. However, most in-

tersection-bounding boxes produced by $o_j^i(t)$ from different real objects in S are empty and are thus removed within the calculation. The intersection of bounding boxes itself is not expensive, leading to fast computation for an arbitrary number of objects. The collision detection computation time also directly depends on the number of detected objects. The performance for increasing numbers of objects is important, since robot workspaces generally contain multiple detected objects.

The number of depth sensors implemented is significant for use in real applications. As the number of perspectives available increases, the frequency of occlusions within S leading to pseudo-objects or small minimum distances between robot and detected objects will decrease. Since the fusion of depth images directly depends on the number of depth sensors, calculation time for object detection significantly increases proportionally to the number of depth sensors (Fig. 3) but the number of depth sensors does not influence the other computational steps.

Experimental results

The prototype system is implemented as a master-slave architecture, with all data processing steps done on the master. Each depth sensor has an own slave, which only acquires the current depth images and transfers them to the master. This is done via a local area network.

The prototype setup is built in a workcell containing a Stäubli RX130 robot (Fig. 4, left). The CAD model of the robot is known and a direct connection between the robot control and the master is established. At any time, the current robot configuration can be requested from the robot controller and a velocity for the current trajectory segment can be sent to the controller.

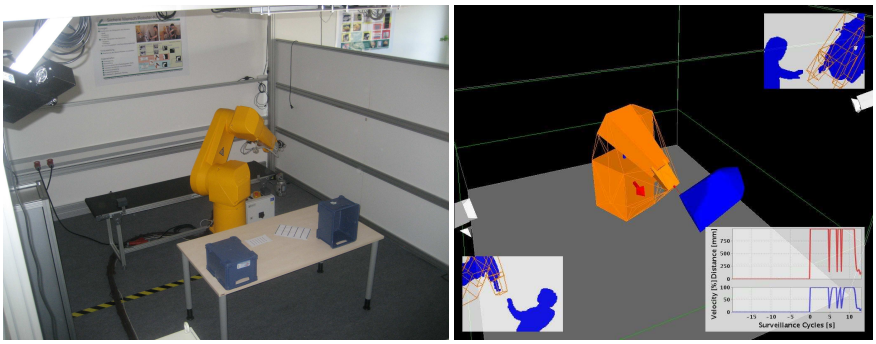

Fig. 4 Left: Prototype setup in a workcell containing an industrial robot and PMD cameras placed in the upper corners of the workcell. Right: Visualisation of the minimum distance calculation between robot (orange) and detected unknown objects (blue). The small images display the results of the object detection in the depth images. The fused objects are shown as 3D representations. The minimum distance (red line) is used for velocity control (diagram).

Two PMD cameras [Kraft04] are used in the current configuration of the prototype setup. The model Vision 19k used has a resolution of 120x160 *pixels* and an aperture angle of 40°. Since PMD cameras are active sensors, each of them must use a different modulation frequency. Multiple PMD cameras using a single modulation frequency would cause interferences within the measured depth images. The shutter time is user-controlled and was set within an interval from 25 to 50 *ms*. Sensor noise increases with decreasing shutter time. Up to 15 *fps* can be provided by the camera to its slave using Firewire.

To obtain lateral calibration of the PMD cameras, a standard algorithm was used [Tsai86]. For depth calibration, constant offsets dependent on the modulation frequencies were calculated for the measured correspondences. A mean deviation of two pixels in lateral coordinates and 25 mm in depth values was the result.

Due to the variation in the measured values, the quality of the depth images needed to be improved with filters. In the prototype system, averaging filters were applied to the raw depth images and morphological filters (erosion, dilation, open, close) were used on the segmentation images. By adjusting filter parameters, the quality of the depth measurement and object detection was increased significantly. The best results were achieved for an 3x3 averaging filter and an 3x3 Open filter.

The detected objects were calculated for these depth images. The robot model (consisting of one cube per robot link) for the current robot configuration was generated and used in the known object calculation. In the 3D representation (Fig. 4, right), the model itself is displayed as an orange volume and the calculated known object as an orange mesh. The robot model generated was used as a safety zone for the collision detection step by calculating the minimum distance between the robot and the nearest detected unknown objects. The minimum distance measured was used to limit the maximum velocity of the robot (Fig. 4, right). A logarithmic mapping determined the velocity limitation for the current distance. The maximum acceleration since the last algorithm cycle is limited, too.

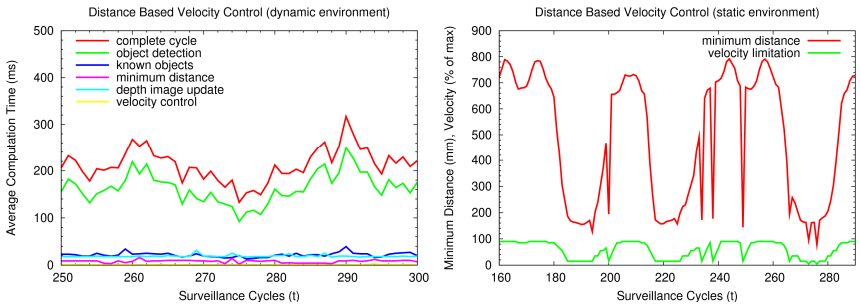

Fig. 5 Diagrams showing a sequence of logged surveillance cycle data from experiments in a static and dynamic robot environment with two depth cameras. The calculation times for each computational step and the complete cycle time are displayed (left). The minimum distance measured is used for simple velocity control (right).

The diagrams in Fig. 5 show logged sequences of surveillance cycle data from different experiments. The robot program loop was a circular movement between both ends of the conveyor belt. In the static environment, some boxes were placed on the table in front of the robot (Fig. 4, left). In the dynamic environment, a human entered *S*, left it and also placed boxes on the table or removed them.

Of all computational steps in the surveillance cycle, object detection consumes the majority of the algorithm cycle time. All other steps are almost constant. The measured distances in the right-hand diagram show that the repeated accuracy of measurements is within an interval of centimetres. It also shows that because of measurement errors, single frames lead to minimum distances much smaller than the real distances due to appearance of a small pseudo-object near the robot base.

Conclusions

This paper introduced the first approach for online collision detection of industrial robots with multiple depth cameras. The main contribution is fast, 3D reconstruction of robot environments containing multiple unknown objects. Previous surveillance and collision detection approaches use multiple colour cameras or only one depth sensor. Using an integrated robot model, a collision detection algorithm is applied to the space occupied by detected unknown objects within the surveilled area. Minimum distances between the robot and any detected unknown object may be used for velocity reduction or trajectory adjustment.

The main advantage over image-based surveillance systems like the current SIMERO system or SafetyEye is the high independence from external light sources. Dynamic lighting conditions or changes in object texture within the surveilled space cause fewer problems. The latter system also does not provide dynamic safety zones and require complete separation of humans and robots.

The algorithm introduced is usable with an arbitrary number of depth sensors. The current version should be used with a small number of sensors for applications with stringent time limitations, since the cycle time is directly dependent on the number of sensors used. The algorithm can be adapted to a variety of other surveillance and security applications, depending on a redefinition of the safety zones applied. For instance, in museum surveillance this can be a static safety zone around the displays, resulting in an intrusion or proximity alert.

Positioning of the depth sensors is important for the quality of results, because with few sensors, an object located between sensor and robot can lead to computed distances much smaller than actual distances. With optimised positioning, the robot and unknown objects can be separated in at least one depth image.

Further research will address automatic positioning of the depth sensors around the surveilled space. The quality of approximation is strongly related to an optimal number and position for variable geometry of robot workspaces.

References

[Barber96] C.B. Barber, D.P. Dobkin and H.T. Huhdanpaa: “The Quickhull algorithm for convex hulls”, ACM Transactions on Mathematical Software, 22(4): 469-483, Dec 1996, http://www.qhull.org

[Cameron97] S. Cameron: “Enhancing GJK: Computing Minimum and Penetration Distances between Convex Polyhedra”, IEEE International Conference on Robotics and Automation, April 1997

[Ebert03] D. Ebert: “Bildbasierte Erzeugung kollisionsfreier Transferbewegungen für Industrieroboter”, Schriftenreihe Informatik 2003, Band 12, ISBN 3-936890-23-4

[Gecks06] T. Gecks and D. Henrich: “Multi-Camera Collision Detection allowing for Object Occlusions”, 37th International Symposium on Robotics (ISR 2006) / 4th German Conference on Robotics (ROBOTIK 2006)

[Gecks07] T. Gecks and D. Henrich: “Path Planning and Execution in Fast-Changing Environments with Known and Unknown Objects”, IEEE International Conference on Intelligent Robots and Systems, 2007

[Henrich08] D. Henrich and T. Gecks: “Multi-camera collision detection between known and unknown objects”, IEEE International Conference on Distributed Smart Cameras, 2008

[Hu04] W. Hu, T. Tan, L. Wang and S. Maybank: “A survey on visual surveillance of object motion and behaviors”, IEEE Transactions on Systems, Man and Cybernetics, 34(3), 334-352, 2004

[Ingensand05] H. Ingensand and T. Kahlmann: „Systematic investigation of properties of PMD-sensors“, 1st Range Imaging Research Day 2005

[Kuhn06] S. Kuhn and D. Henrich: “Modelling intuitive Behaviour for Safe Human/Robot Coexistence and Cooperation”, IEEE International Conference on Robotics and Automation, 2006

[Kuhn07] S. Kuhn and D. Henrich: “Fast Vision-Based Minimum Distance Determination Between Known and Unknown Objects”, IEEE International Conference on Intelligent Robots and Systems, 2007

[Kraft04] H. Kraft et al.: „3D-Camera of High 3D-Frame Rate, Depth Resolution and Background Light Elimination Based on Improved PMD (Photonic Mixer Device)-Technologies”, OPTO 2004, AMA Fachverband, http://www.pmdtec.com/

[Li02] M. Li, H. Schirmacher, M. Magnor and H.-P. Seidel: “Combining Stereo and Visual Hull Information for On-Line Reconstruction and Rendering of Dynamic Scenes”, IEEE Workshop on MMSP, pp9-12, 2002

[Matusik01] W. Matusik, C. Buehler and L. McMillan: “Polyhedral Visual Hulls for Real-Time Rendering”, Proceedings of the 12th Eurographics Workshop on Rendering, pp116-126, 2001

[Meisel94] A. Meisel: “3D-Bildverarbeitung für feste und bewegte Kameras”, Vieweg Verlag, Reihe Fortschritte der Robotik Nr. 21, 1994

[Pilz06] Patent DE 10 2006 057 605 A1 “Verfahren und Vorrichtung zum Überwachen eines dreidimensionalen Raumbereichs”, Pilz GmbH & Co. KG, 2006, http://www.safetyeye.com/

[Reed99] M. K. Reed and P. K. Allen: “3-D Modeling from Range Imagery: An Incremental Method with a Planning Component”, Image and Vision Computing 17(2), pp99-111, 1999

[Som05] F. Som: “Sichere Steuerungstechnik für den OTS-Einsatz von Robotern”, 4.Workshop für OTS-Systeme in der Robotik, IPA 2005

[Thiemermann03] S. Thiemermann: “team@work - Mensch/Roboter-Kooperation in der Montage”, 2.Workshop für OTS-Systeme in der Robotik, IPA 2003

[Thiemermann05] S. Thiemermann: “Direkte Mensch-Roboter-Kooperation in der Kleinteilmontage mit einem SCARA-Roboter”, IPA-IAO-Bericht, 2005, ISBN 978-3-936947-50-2

[Tsai86] R. Y. Tsai: “An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision”, IEEE Conference on Computer Vision and Pattern Recognition, 1986

[Winkler07] B. Winkler: “Safe Space Sharing Human-Robot Cooperation Using a 3D Time-of-Flight Camera”, International Robots and Vision Show, 2007