Collision-Free Path Planning in Robot Cells Using Virtual 3D Collision Sensors

Tomislav Reichenbach & Zdenko Kovacic

Source: Cutting Edge Robotics, ISBN 3-86611-038-3, pp. 784 , ARS / plV, July 2005

1. Introduction

In industrial robotic systems, it is crucial to avoid collision among the manipulators or with other objects in the workspace. The problem is more complicated for manipulators than for mobile robots, since not only the effector end must move without collision to the desired location, but the links of the arm must also avoid collisions. The problem has been addressed in various ways in literature (Lozano-Perez, 1986) and (Barraquand, Langlois & Latombe, 1992). These methods can generally be divided into two methodologies: global and local.

A global typical technique in collision avoidance consists of exploring a uniform grid in configuration-space (C-space) or configuration time space (CT space) using a best-first search algorithm guided by goal-oriented potential field (Barraquand & Latombe, 1991) and (Hwang, & Ahuja, 1992). Two main problems exist, the obstacles must be mapped into the configuration space (a very complex operation) and a path through the configuration space must be found for the point representing the robot arm. Usage of this method in dynamic complex environments, with more robots sharing the same space, is too time demanding to be accomplished in real time.

A new idea proposed in this work is to detect possible collisions among robotic system entities in a virtual world. Moreover, the idea is to use a collision detection engine, which is extensively used in computer graphics and simulation, to test for possible collisions in full 3D environment with objects consisting of thousands of polygons and modeled as similar to the real ones as close as possible. A modification of manipulator kinematics is proposed in order to control directly a colliding point, hence enforcing fastest collision avoidance. This approach has some advantages of both global and local planning methodologies, as the complete environment is considered in planning and fast collision avoidance response suitable for on-line application in dynamic environments is achieved. Motion planning for models with fairly complex geometry has been proposed in some former papers (Cameron & Qin, 1997) with modified Gilbert, Johnson and Keerthi (GJK) algorithm applied for distance computations and collision detection, but it differs in the nature and use of the collision detection algorithm.

A software simulation package for virtual modeling of complex kinematic configurations has been developed (C++ and OpenGL). Models are created in CAD software and then imported into virtual environments described in the XML. Virtual models provide very inexpensive and convenient way for design, analysis, control, dynamic simulation and visualization of complex robotic systems. Given that virtual models are accurate, collisions

in a virtual environment should occur in the same way as in the real world. The 3D models used in this work are polygonal structured, i.e. polygons form a closed manifold, hierarchical non-convex models (Lin & Gottschalk, 1998) undergoing series of rigid-body transformations. Polygons are made entirely of triangles, as hardware accelerated rendering of the triangles is commonly available in the graphic hardware. All the parameters (link positions, link lengths) for a direct kinematics solution are extracted from a 3D description.

This chapter is organized as follows. First, a brief introduction to kinematic models is given, then a method for collision detection among objects in virtual environment is explained along with an algorithm for determination and generation of new kinematic parameters for the colliding objects. A modification of manipulator kinematics is proposed in order to control directly a colliding point, hence enforcing fastest collision avoidance. Following is the description of collision avoidance strategy and the results of simulations of collision free trajectory planning in the experimental Flexible Manufacturing Systems (FMS).

2. Kinematic Model

The endpoint position and orientation of a manipulator may be expressed as a nonlinear function of the constants such as physical link parameters, and by joint angles or displacements that are variables. To represent the spatial transformations between the joints, several symbolic notations have been proposed. The most widely used notation is the classic Denavit-Hartenberg (D-H) notation (Denavit & Hartenberg, 1955) or its Paul (Paul, 1981) version. These notations however are only valid for the kinematic chains with one branch (i.e. serial kinematic chains), and can lead to ambiguities when dealing with kinematic chains with more then one branch (parallel kinematic chains). The Sheth-Uicker (S-U) (Sheth & Uicker, 1971) notation extends D-H notation for multiple loop kinematic chains in the general case, but is much more complicated because it introduces additional coordinate systems. Another notation presented by Kleinfinger (Khalil & Kleinfinger, 1986), is usable for all serial, treelike or closed loop kinematic chains, and has fewer parameters then the S-U notation, however, since all manipulators employed in this chapter are serial kinematic chains, only D-H notation, albeit in a somewhat modified form, is used.

3. Kinematic Parameters

Six parameters are needed to describe a complete relation between two independent coordinate systems - frames. Like most of the other kinematic notations, D-H notation requires following some rules when designating axes and orientations to the link frames.

3.1. Modified D-H Notation

Assuming that robot links have only one dimension, the length, the classic D-H notation describes the complete relation between two adjacent link frames with only four kinematic parameters $[a_i \quad d_i \quad \alpha_i \quad \theta_i]$. The meaning of these parameters is the following: **rotation** around a z^{i-1} axis with an angle θ_i, **translation** along a z^{i-1} axis with a displacement d_i, **translation** along a $x^i = x^{i-1}$ axis with a displacement a_i, **rotation** around a x^{i-1} axis with an angle α_i.

For two coordinate systems having an arbitrary position and orientation in the 3D space,

the conversion between them can be described with a matrix $\mathbf{M}_i$ given in the [4x4] homogenous matrix form,

$$\mathbf{M}_i = \mathbf{T}_i \cdot \mathbf{R}_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{12} & r_{22} & r_{13} & p_1 \\ r_{13} & r_{23} & r_{13} & p_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where

$$\mathbf{T}_i = \begin{bmatrix} 1 & 0 & 0 & p_1 \\ 0 & 1 & 0 & p_1 \\ 0 & 0 & 1 & p_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R}_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{12} & r_{22} & r_{13} & 0 \\ r_{13} & r_{23} & r_{13} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To convert from matrix $\mathbf{M}_i$, parameter b_i representing a link length along the y^{i-1} axis of the i^{th} coordinate system, has been added. Now, there are 5 independent parameters [a_i b_i d_i α_i θ_i] used in modified D-H notation. Only one of these parameters is a variable, joint position (θ_i if the joint is rotational or d_i if the joint is prismatic) and the other 4 (or 3 in the original D-H notation) are fixed parameters determined only by a manipulator construction. In this way, the D-H transformation matrix between two adjacent link frames has the following form:

$${}^{i-1}\mathbf{T}_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i - b_i \sin\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\cos\alpha_i \sin\alpha_i & a_i \sin\theta_i + b_i \cos\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

To get D-H parameters [a_i b_i d_i α_i θ_i] for the i^{th} link from a joint frame transformation matrix ${}^{i-1}\mathbf{T}_i$ written in the form as in eq. (2), a correct solution has to be chosen from a solution set $S = \{\alpha_i^1, \alpha_i^2, \alpha_i^3, \theta_i^1, \theta_i^2, \theta_i^3\}$. Because of the ambiguity of solutions of trigonometric equations, several possible solutions are available:

$$\begin{aligned} \alpha_i^1 &= atan2({}^{i-1}\mathbf{T}_i(3,2), {}^{i-1}\mathbf{T}_i(3,3)) \\ \alpha_i^2 &= atan2(-{}^{i-1}\mathbf{T}_i(2,3), {}^{i-1}\mathbf{T}_i(2,2)) \\ \alpha_i^3 &= atan2({}^{i-1}\mathbf{T}_i(1,3), -{}^{i-1}\mathbf{T}_i(1,2)) \\ \theta_i^1 &= atan2({}^{i-1}\mathbf{T}_i(2,1), {}^{i-1}\mathbf{T}_i(1,1)) \\ \theta_i^2 &= atan2(-{}^{i-1}\mathbf{T}_i(1,2), {}^{i-1}\mathbf{T}_i(2,2)) \\ \theta_i^3 &= atan2({}^{i-1}\mathbf{T}_i(1,3), -{}^{i-1}\mathbf{T}_i(2,3)) \end{aligned} \tag{3}$$

4. Direct Kinematics

A transformation from the n^{th} local frame to the global frame is defined as:

$$\begin{aligned} {}^0\mathbf{T}_n &= \prod_{k=0}^{n} \left(\mathbf{M}_k \cdot \mathbf{M}_{jk} \right) \\ \mathbf{M}_k &= \mathbf{T}_k \cdot \mathbf{R}_k \end{aligned} \tag{4}$$

Where:

- $\mathbf{T}_k$ is k^{th} joint translation matrix.
- $\mathbf{R}_k$ is k^{th} joint rotation matrix.
- $\mathbf{M}_{jk}$ is k^{th} joint transformation matrix.

Matrix $\mathbf{M}_{jk}$ is defined as:

$$\mathbf{M}_{jk} = \begin{bmatrix} \cos\beta_k \cos\gamma_k & -\sin\gamma_k & \sin\beta_k & l_1 \\ \sin\gamma_k & \cos\alpha_k \cos\gamma_k & -\sin\alpha_k & l_2 \\ -\sin\beta_k & \sin\alpha_k & \cos\alpha_k \cos\beta_k & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where α_k, β_k and γ_k are the rotation angles around the respective x_k, y_k, z_k axes, and l_1, l_2, l_3 are translations along the respective x_k, y_k, z_k axes of the k^{th} coordinate system. If the system is using D-H notation, then all joint rotations and translations are done around and along the local z_k-axis, so the eq. (5) takes the following form:

$$\mathbf{M}_{jk} = \mathbf{M}_{qk} = \begin{bmatrix} cos\gamma_k & -\sin\gamma_k & 0 & 0 \\ \sin\gamma_k & \cos\gamma_k & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos q_k & -\sin q_k & 0 & 0 \\ \sin q_k & \cos q_k & 0 & 0 \\ 0 & 0 & 1 & q_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

where q_k is a joint variable defined as $q_k = (1-\xi_k)\cdot d_k + \xi_k \cdot \theta_k$. If the joint is rotational then $\xi_k = 1$, else it is prismatic and $\xi_k = 0$. Due to high complexity of 3D models, there are more coordinate systems than joints. All coordinate systems that do not undergo rigid-body motion are considered static with matrix $\mathbf{M}_{qk}$ equal to the unity matrix. A direct kinematic solution, a tool position and an orientation, are easily extracted from eq. (4).

5. Collision Detection

Collision detection is a part of *interference detection*, which can be divided into three portions: collision detection that detects whether objects are in collision, collision determination that finds the exact collision point, and finally, collision avoidance that determines what actions should be taken in response to the collision. There are numerous approaches to a collision detection problem (Jimenez, Thomas & Torras, 2001), which can be mainly grouped into; space-time volume intersection, swept volume interference, multiple interference detection, and trajectory parameterization. A collision detection algorithm used in this work belongs to a multiple interference detection category. This implies that the algorithm reduces a general collision detection problem to multiple calls to the static interference tests focused on detecting intersections among simple geometrical entities; triangles and oriented bounding boxes (OBB) belonging to the objects being tested. As the algorithm is static, i.e. collision detection occurs only at discrete times, it is fast enough and effective from the computational point of view, thus it can provide real-time collision detection in very complex (high polygon count) virtual environments.

5.1. Oriented Bounding Boxes (OBB)

An oriented bounding box (OBB) can be described by the center point of the box $\mathbf{b}^c$, and a center rotation vector $\mathbf{b}^r$ (or three normalized vectors $\mathbf{b}^u$, $\mathbf{b}^v$, $\mathbf{b}^w$ that represent the face

normals). To express the OBB dimensions the most compact representation is with the half-lengths h_u^B, h_v^B, h_w^B, which represent dimensions along the respective u, v, w axis.

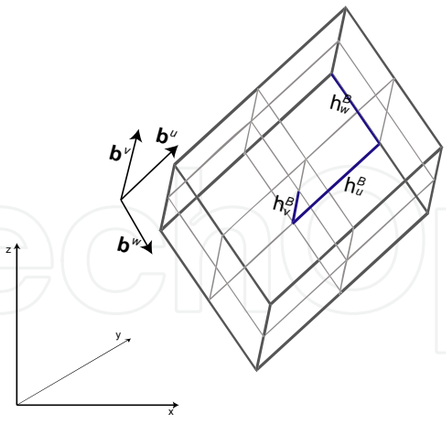


Figure 1. Oriented bounding boxes

An intersection test between two OBBs A and B is accomplished with a fast routine presented in (Gottschalk, Lin & Manocha, 1996). This routine is based on separating axis theorem and is about an order of magnitude faster than methods that use linear programming or closest features. According to the separating axis theorem, it is sufficient to find one axis that separates objects A and B to determine they do not overlap. The total number of tested axes is fifteen, three from the faces of A, three from the faces of B and nine from combination of edges from both OBBs (3x3). The potential separating axes that should be orthogonal to the faces of A and B so the simplest way is to use the normals to the faces $\mathbf{a}^u$, $\mathbf{a}^v$, $\mathbf{a}^w$ and $\mathbf{b}^u$, $\mathbf{b}^v$ and $\mathbf{b}^w$. The remaining nine potential axes are formed by one edge from each OBB, $c^{ij} = a^i \times b^j, \forall i \in \{u, v, w\}$ and $\forall j \in \{u, v, w\}$. The OBBs are then projected onto the axes to see if both projections overlap on all axes, i.e. if the OBB A and B overlap. The maximum number of operations is reached when the two OBBs overlap, and is around 180 (without the transform of B into A's coordinate system) (Gottschalk, 2000). However, in most cases without overlap, a separating axis is found earlier.

5.2. Triangle/triangle Intersection

After the intersection between the OBBs is determined, an exact collision point is found with triangle/triangle intersection test (see Fig. 2). Often not only the information whether two triangles intersect but also their exact intersection point is needed. Several methods for triangle/triangle intersection test are available, with two of the fastest being the *interval overlap method* developed by (Möller, 1997) and the triangle/triangle intersection method found in ERIT package (Held, 1997).
The ERIT's method of detecting whether two triangles T_1 and T_2 intersect, can be briefly summarized as:

1. Compute $\pi_2 : \mathbf{n}_2 \cdot \mathbf{x}$, the plane in which T_2 lies.
2. Trivially reject if all points of T_1 are on the same side of π_2
3. Compute the intersection between π_2 and T_1
4. If this line segment intersects or is totally contained in T_2, then T_1 and T_2 intersect; otherwise, they do not.

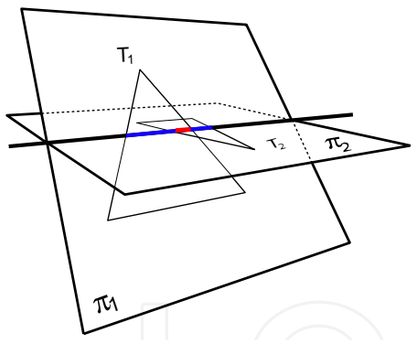


Figure 2. Triangle/triangle intersection

In some occasions, it is not necessary to determine the exact collision point, but only to check if objects are in collision. If the object is distant it can be approximated with just one OBB (see Fig. 3) and consequently collision detection will be faster. The balance between fast collision detection and exact collision point determination is the case shown in Fig. 4. The 3rd hierarchy level OBBs are used for objects that are near each other (see Fig. 5). Normally, a manipulator will not contain higher levels of OBBs hierarchies (4th, 5th, etc...), since they would not provide computational advantage over a triangle/triangle intersection test. Binary trees are often used for the hierarchy representation and different strategies for hierarchy building are available, e.g. K-DOPTtree, OBBtree (Gottschalk, Lin & Manocha, 1996). The OBBtree approach is the most similar to the one used here. A depth of hierarchical tree, and a decision when the triangle/triangle test will be used instead of OBB overlap check, can be conditioned by available computational time and the complexity of the model.

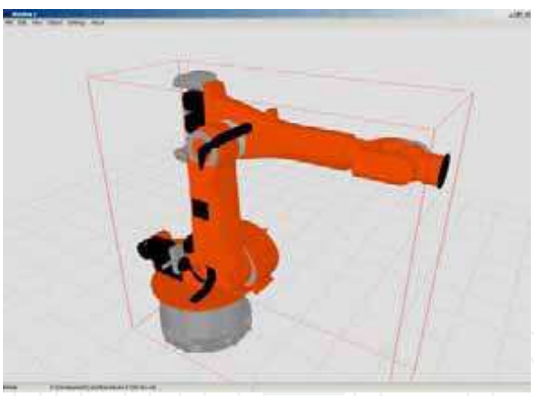


Figure 3. 1^{st} level of OBB hierarchy

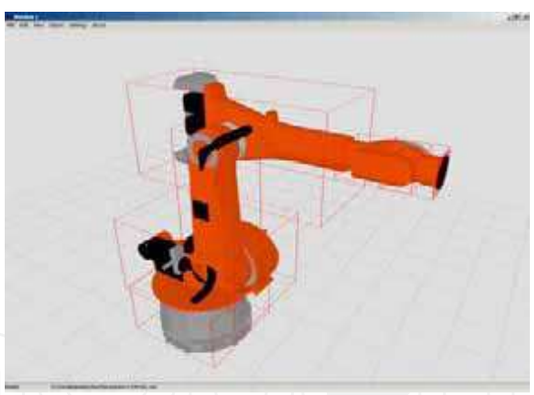


Figure 4. 2^{nd} level of OBB hierarchy

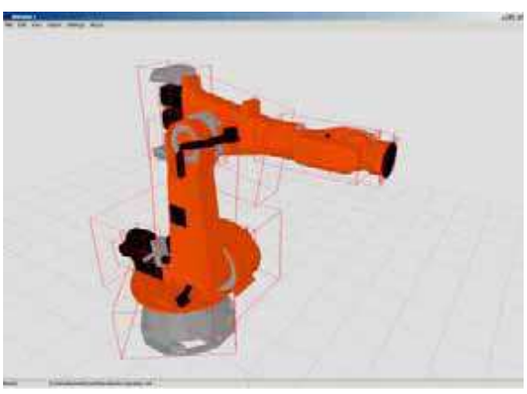


Figure 5. 3^{rd} level of OBB hierarchy

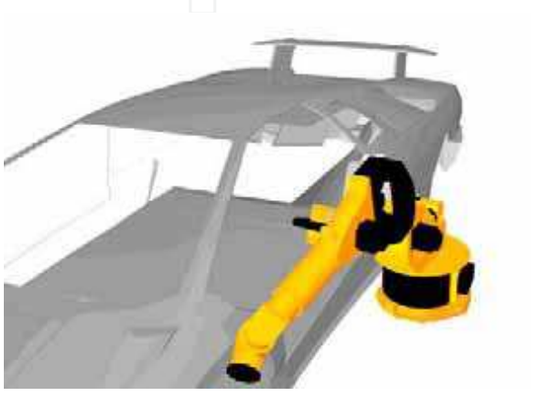


Figure 6. Manipulator prior to the collision

The next logical step is to use information about the intersection point (i.e. collision) and try to prevent the collision by changing the path of one or both colliding objects. Assuming that at least one colliding object is a robot, one must know kinematic parameters of the robot to be able to prevent collisions and plan collision-free robot trajectories. For that purpose regular kinematic parameters associated with positions and orientations of all robot joints and end effectors are not sufficient for proper collision-free trajectory planning. Namely, these parameters do not describe all points on the robot surface that could collide with the environment. While in practice, determination of kinematic parameters for an arbitrary collision point on the real robot surface is a very difficult goal, in the virtual environment this may be resolved in an elegant way by using a kinematic model of the robot derived from its virtual 3D model (Reichenbach & Kovačić, 2003). Once the triangle/triangle intersection test has established the exact collision point, the determination of a link in the hierarchy where collision will take place is straightforward. If more than one link is in the collision, the link that is higher in the object hierarchy is preferred. A new D-H kinematic model is generated from eq. (3) with the collision point serving now as a tool frame origin (see Figs. 6 and 7), so inverse kinematics for the collision point may be calculated.

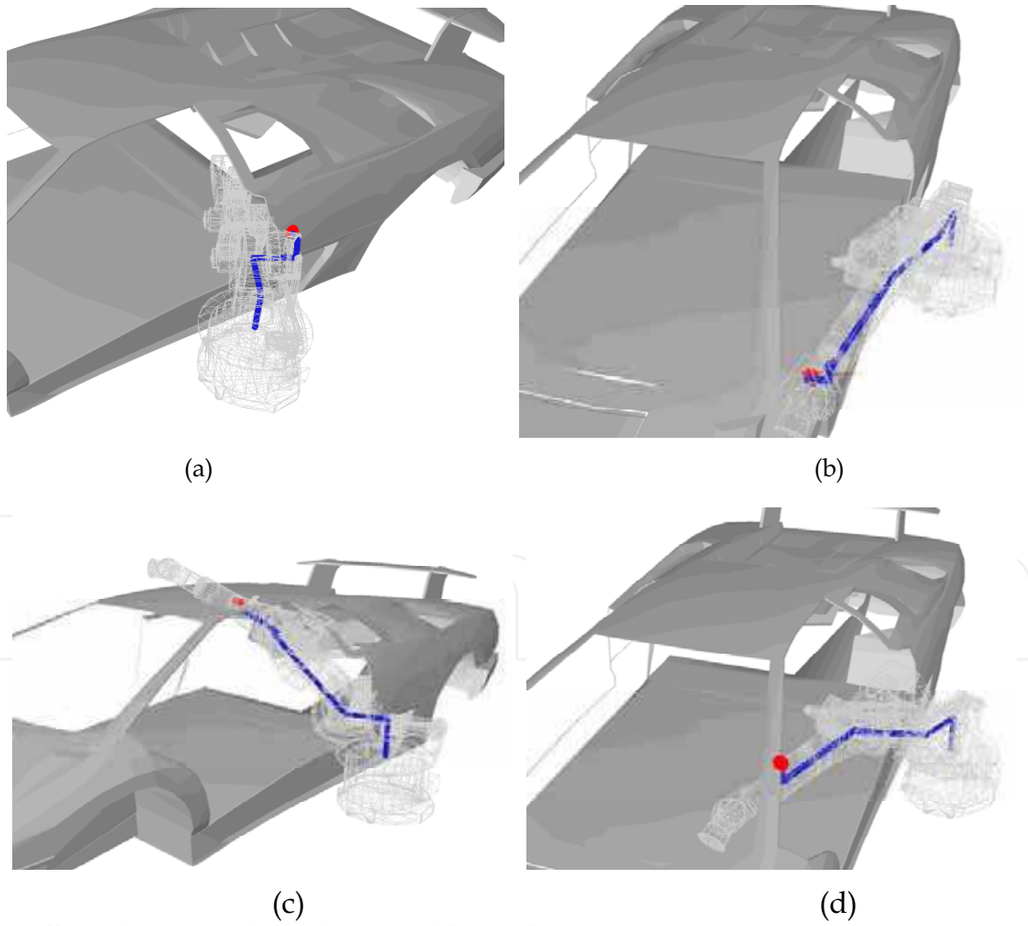


Figure 7. Different kinematic chains determined by a collision point

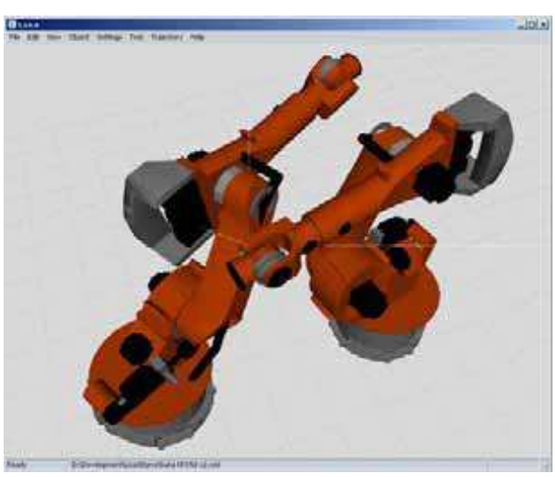
The number of collision checks in one step is:

$$N_{col} = \binom{N_{dyn}}{2} + N_{dyn} \cdot N_{stat} + N_{link} \quad , \tag{7}$$

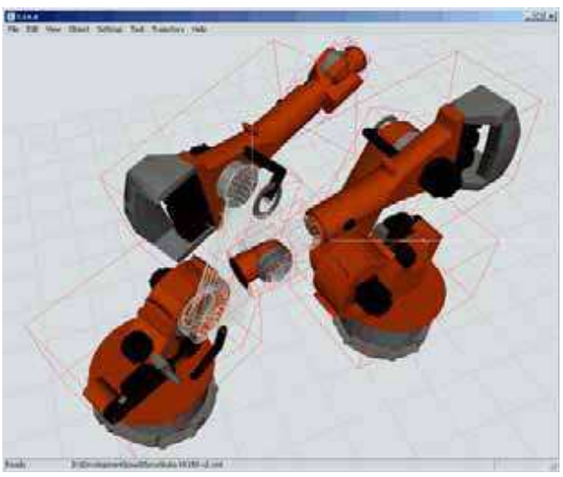
where N_{dyn} is a number of dynamic objects, N_{stat} is a number of static objects and N_{link} is number of links in a manipulator. Further reduction in the number of collision checks can be achieved by using a *sweep and prune* technique, which exploits the temporal coherence, normally found in virtual environments (Klosowski, 1998) and (Möller & Rundberg, 1999). Temporal coherence (or frame-to-frame coherence) means that objects undergo small changes in their position and orientation between two consecutive frames.

6. Collision Avoidance

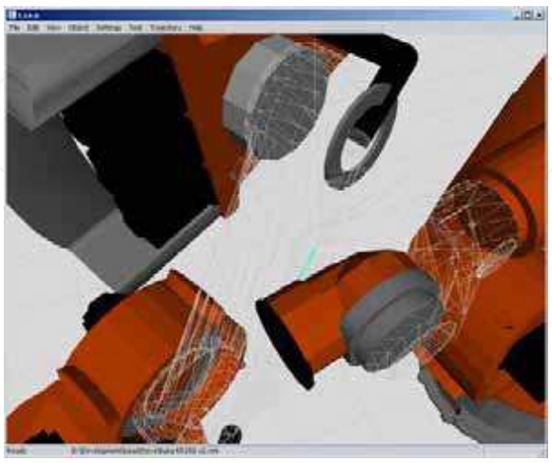
Oriented bounding boxes (OBB) are used to determine the distance and the collision between different objects at the first hierarchical stage. As one moves down the generated OBB hierarchy-tree a search for the collision point is narrowed, thus finally allowing the exact collision point determination with triangle/triangle intersection test at the final overlapping OBB nodes (see Fig. 8). How deep is the hierarchical tree, or when the triangle/triangle test will be used instead of OBB overlap check, can be specified depending on the computational time available and the complexity of the model.



(a) Prior to the collision



(b) OBBs and triangles of the colliding link



(c) Close-up

Figure 8. Two KukaKr150 robots in collision

Required trajectory of a manipulator is checked against possible collisions and if a collision is detected in some imminent manipulator position, the manipulator link is

moved away from the possible collision point. Newly found collision-free points are then inserted into the previous trajectory and on-line re-planning of the trajectory is made. A complete check for all points in the trajectory and a trajectory re-planning are considered as one iteration of a collision avoidance algorithm. Iterative collision avoidance actions are taken until trajectories become collision-free.

Different collision avoidance strategies are proposed and tested. One strategy iteratively moves the first collision point for a predetermined displacement value while direction of the displacement is calculated based on collision point. A value of the displacement is predetermined according to the sparsity of objects in an environment, with larger movements possible in sparser environments. The algorithm proceeds to the next iteration until there are no collisions in the trajectory of the manipulator. Another possible strategy is to move all collision points simultaneously in one iteration for a predetermined displacement value. A strategy, that moves the middle[1] collision point for a minimum distance required to evade the collision, is presented in section 7. The avoidance movement is made in a direction of the normal to the colliding surface (determined by a collision detection algorithm) and the value of displacement is calculated as the distance from the collision surface to the end of the link along the kinematic chain. In addition, the amount of displacement may be incremented by the projection of the colliding link OBB to the direction of collision avoidance movement and the direction of general movement between the points in the trajectory (see Fig. 9).

Inverse kinematics calculus at this point is done with modified manipulator kinematics, with the collision point serving as a tool position, and only links higher in a hierarchy from the collision link are moved. In the environments with dynamic objects it is possible to estimate the time interval when collision is likely to occur, by observing how far the objects are and how rapidly they move. Collision tests are then focused on this interval. A trajectory planning is done on-line, according to the algorithm proposed by (Ho & Cook, 1982) and (Taylor, 1979) taking into account maximum possible joint velocities and accelerations.

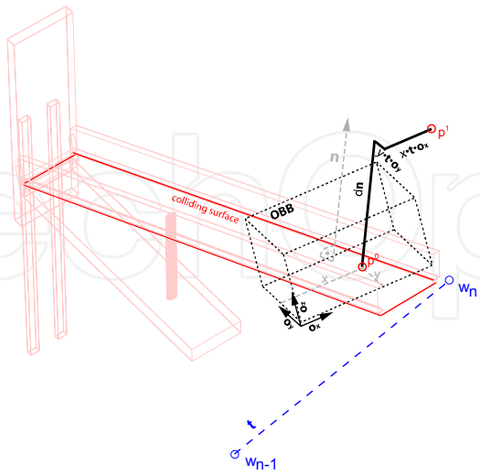


Figure 9. A displacement calculation

When a trajectory planning is made, a volume swept by robot is checked against possible collisions. While the swept volume collision checks are made, an off-line trajectory planner may normally operate with a deeper hierarchical OBB level, due to a different amount of the computational time available, than an on-line trajectory planner. During an on-line

[1] Point in the middle of a continuous collision stretch is referred as the middle point

search for a collision-free path, a progressive hierarchical OBB level approach is used. Objects that are considered far from each other are tested only in the first level OBB tree hierarchy. As objects are approaching to each other, deeper hierarchical OBB trees are used to check against collisions. Further improvement of the search for a collision-free path is made by reducing the number of checks for the moving objects. The projection of object face normals to a relative velocity vector of the object must be positive, similarly to what is proposed in (Kheddar, Coquillart & Redon 2002), otherwise the object is not checked for collisions[2].

7. FMS Control Application

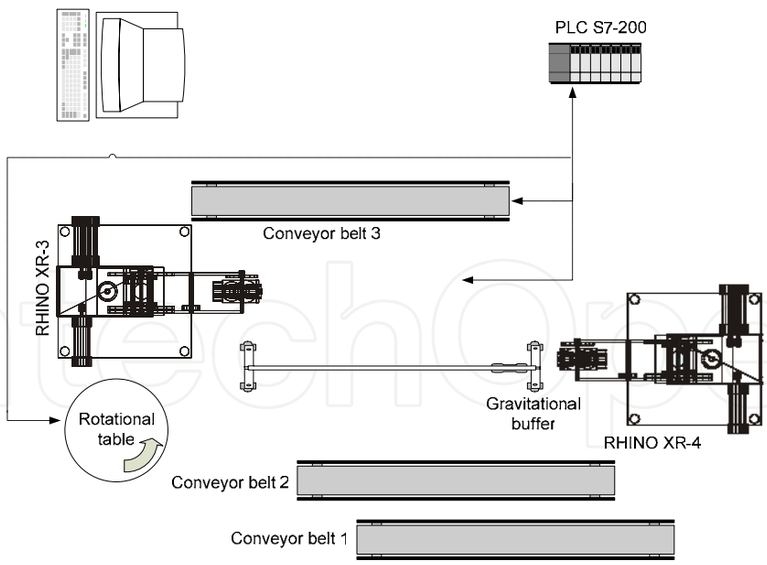
A controlled environment is static in the sense that all positions, dimensions and velocities of objects are known, but objects can be dynamic, i.e. they can undergo rigid-body transformations. In a constantly changing and partially unpredictable environment on-line trajectory planning must be used. In the FMS, trajectories are currently planned off-line, which causes serious limitations in reassignments to new tasks and results in time-consuming coordination rules. The FMS resources are required to move from a job before it is completed (pre-emption property), and the process should not have to hold the resources already allocated to it until it has all resources required to perform a task (hold while waiting property).

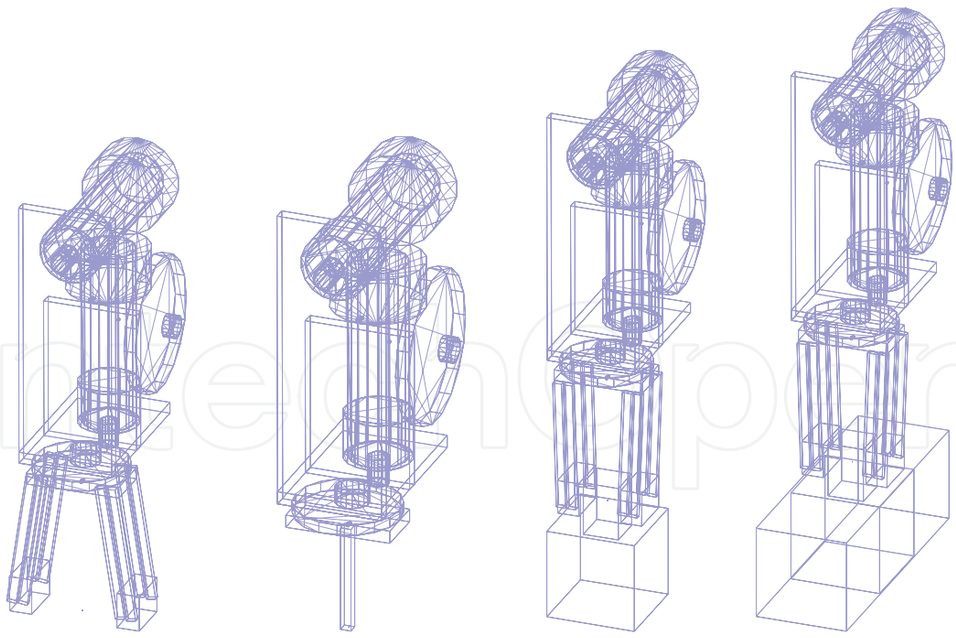
A target FMS (see Figs. 10 and 11) contains two educational robots Rhino XR-3 and Rhino XR-4. There could be the case where the Rhino XR-4 robot is requested to move from a conveyer belt pickup/release place to one x-y table position. During this task, the Rhino XR-4 robot can also hold some object in its gripper. In the regarded FMS testbed, one of the obstacles is the gravitational buffer. In case of manipulator holding objects in its gripper, other FMS elements could also become obstacles, as it is shown in section 7.4 where x-y table is the first obstacle RhinoXR4 robot could collide with during its movement. The collision can be prevented by using the collision avoidance strategy in which the distance from the collision point and the extremes of a colliding object is calculated and directly used for the collision avoidance maneuver.

The idea is to use virtual models as a part of supervisory control of the target FMS. All sensor data are processed in the way that the virtual objects can be moved exactly as the real ones, while at the same time testing for possible collisions and employing the collision avoidance strategies when necessary. The virtual supervisory control provides collision-free trajectories generation having only the trajectory start position and the trajectory end position, thus eliminating the need to specify additional way points in the trajectory planning.

The results of several different simulation experiments and their comparison are presented in the following subsections. In addition, the influence of end-effector construction and the influence of the conveyed object is discussed (section 7.5). Different end-effectors, or different objects moved by the manipulator, produce different trajectories. With a growing complexity in the construction of the end-effector of the manipulator, the collision avoidance maneuver computational time and planned trajectory complexity increase.

[2] The approach is based on the following premise, that the objects that are moving away from each other are not checked for collisions





(a) Type-I (b) Type-II (c) Type-III (d) Type-IV

Figure 12. Tool configurations

7.1 Planning with End-Effector Type-I

In the first experiment a collision avoidance trajectory planning is done with the end-effector type-I attached to a manipulator (see Fig. 12a). Joint values of the Rhino XR-4 robot are shown in Fig. 16. In the first iteration of collision avoidance trajectory planning, collisions are continuous along the trajectory stretch where RhinoXR-4 robot is moving near the gravitational buffer (see Figs. 16a and 17a). Actual manipulator positions can be observed in Fig. 13 (complete system) and in Fig. 18a. In order to depart from the collision point the manipulator is moved in a direction pointed by the normal to the colliding surface of the gravitational buffer (detected in the collision stage).

A movement amount is calculated as a distance between a collision point and a tool tip, in order that this point becomes collision-free in the next iteration of trajectory planning. A newly inserted way point, determined by the direction of the normal and the calculated movement amount, has significantly decreased a number of collision points. Joint values of the manipulator in the 2nd trajectory planning iteration are shown in Fig. 16b.

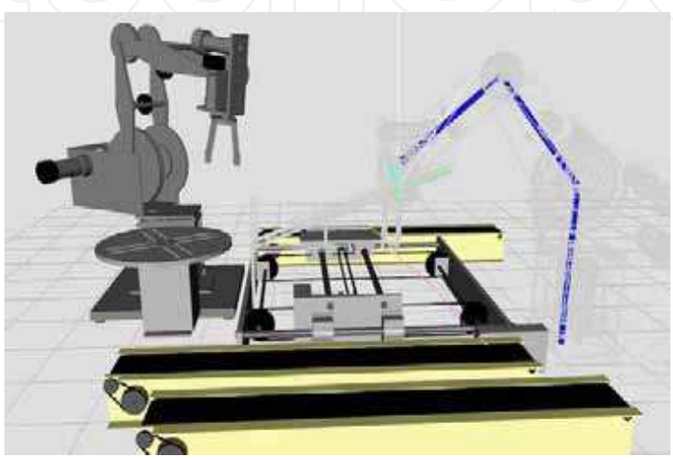


Figure 13. RhinoXR4 robot in collision with gravitational buffer 1st iteration

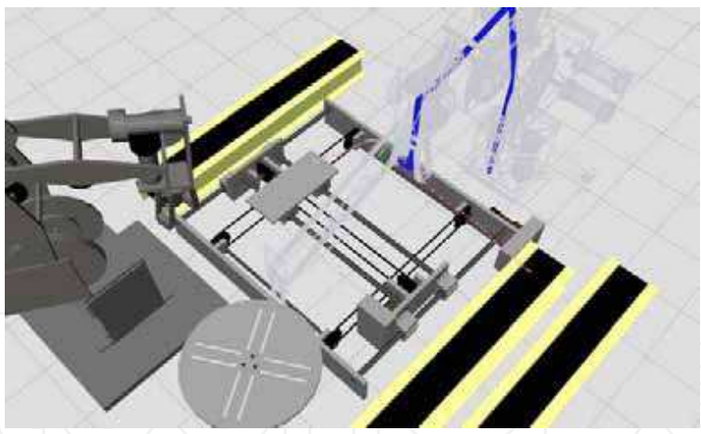


Figure 14. RhinoXR4 robot in collision with gravitational buffer 2nd iteration

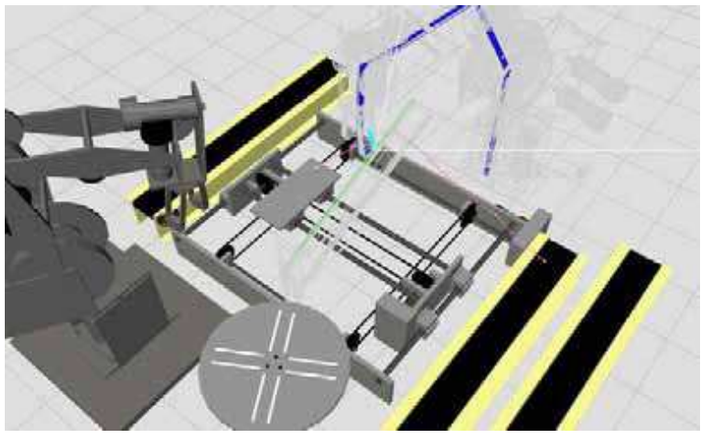
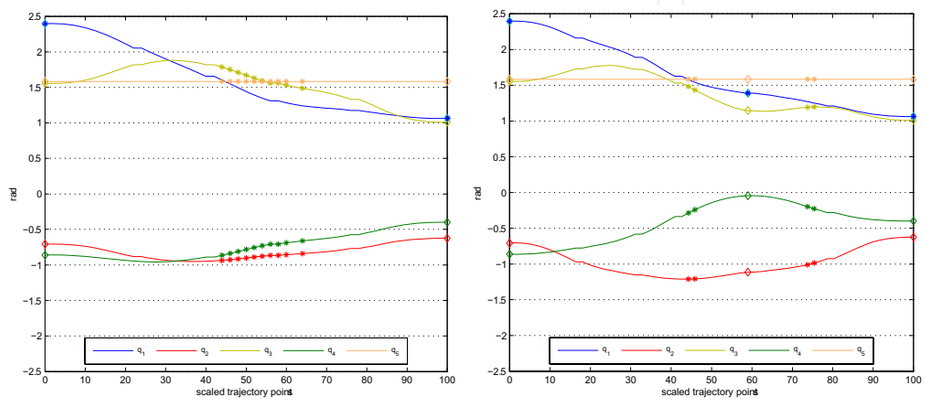
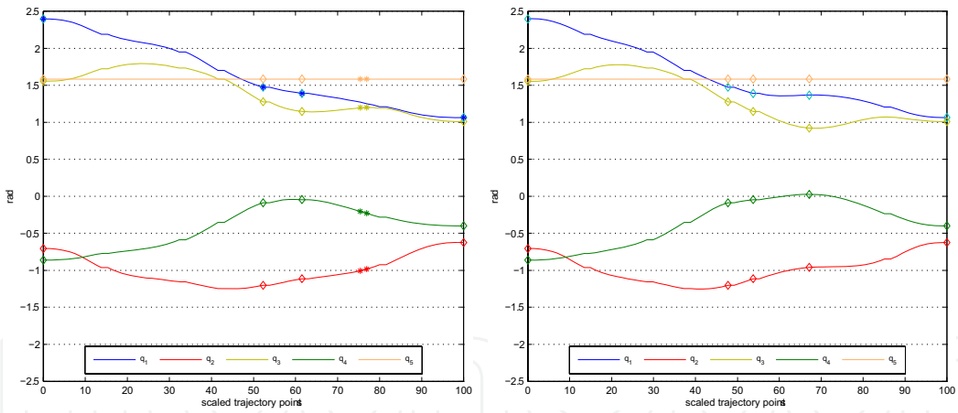


Figure 15. RhinoXR4 robot in collision with gravitational buffer 3rd iteration

Now, there are only two small trajectory stretches where the collision is still present. Subsequent two iterations (see Figs. 16c and 16d) insert two additional way points and eliminate all collision points from the trajectory. It may be valuable to observe that the 3rd and the 4th iteration are necessary because of the position of the end-effector and its construction. In Figs. 14 and 15, one can notice that only the pincers of the end-effector (gripper tool) are colliding, but otherwise manipulator is outside the collision area. The manipulator positions throughout the iterations of collision-free trajectory planning are shown in Fig. 18. A complete trajectory evolution in the Euclidian space R^3 can be seen in Fig. 25a.

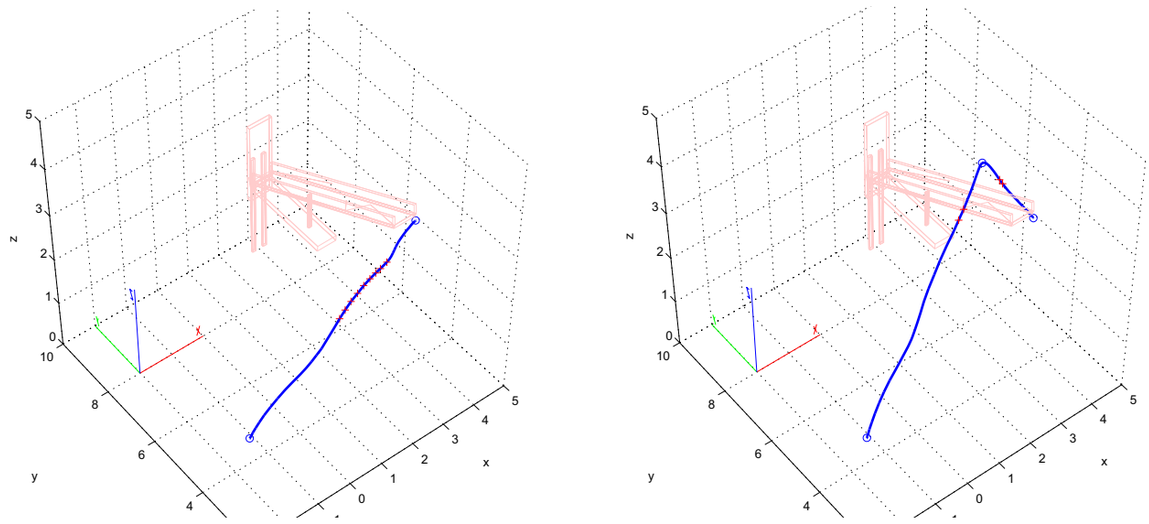


(a) 1st iteration of trajectory planning (b) 2nd iteration of trajectory planning

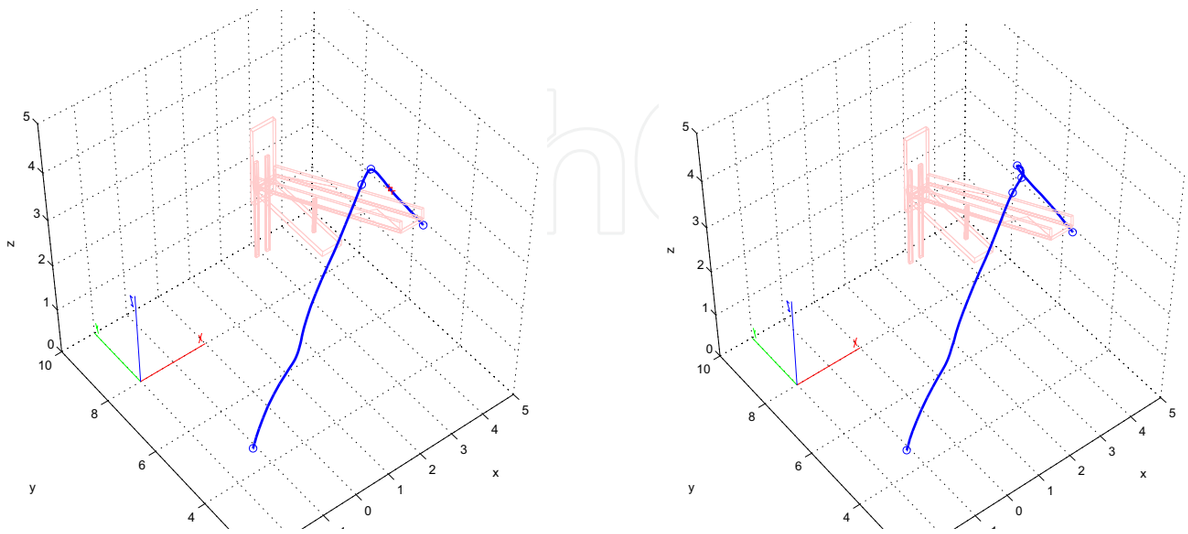


(c) 3rd iteration of trajectory planning (d) 4th iteration of trajectory planning

Figure 16. Joint values throughout the iteration in collision-free trajectory planning

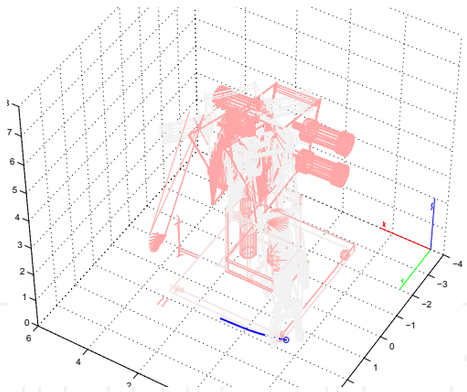


(a) 1st iteration of trajectory planning (b) 2nd iteration of trajectory planning



(c) 3rd iteration of trajectory planning (d) 4th iteration of trajectory planning

Figure 17. End-effector positions throughout the iteration in collision-free trajectory planning



Cutting Edge Robotics

Edited by Vedran Kordic, Aleksandar Lazinica and Munir Merdan

ISBN 3-86611-038-3

Hard cover, 784 pages

Publisher Pro Literatur Verlag, Germany

Published online 01, July, 2005

Published in print edition July, 2005

This book is the result of inspirations and contributions from many researchers worldwide. It presents a collection of wide range research results of robotics scientific community. Various aspects of current research in robotics area are explored and discussed. The book begins with researches in robot modelling & design, in which different approaches in kinematical, dynamical and other design issues of mobile robots are discussed. Second chapter deals with various sensor systems, but the major part of the chapter is devoted to robotic vision systems. Chapter III is devoted to robot navigation and presents different navigation architectures. The chapter IV is devoted to research on adaptive and learning systems in mobile robots area. The chapter V speaks about different application areas of multi-robot systems. Other emerging field is discussed in chapter VI - the human- robot interaction. Chapter VII gives a great tutorial on legged robot systems and one research overview on design of a humanoid robot.The different examples of service robots are showed in chapter VIII. Chapter IX is oriented to industrial robots, i.e. robot manipulators. Different mechatronic systems oriented on robotics are explored in the last chapter of the book.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen