DERIVATION OF KINEMATIC PARAMETERS FROM A 3D ROBOT MODEL USED FOR COLLISION-FREE PATH PLANNING

Tomislav Reichenbach, Zdenko Kovačić

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, CROATIA
E-mail: tomislav.reichenbach@fer.hr, zdenko.kovacic@fer.hr
URL: http://flrcg.rasip.fer.hr

Abstract: This paper presents a method of deriving a kinematics model from a 3D robot model positioned in the virtual environment. This method is applicable to non-convex hierarchical polygons models constructed of triangles and allows determination of kinematic parameters for each point (triangle) on the model including those on the surface of the model. Real-time collision detection algorithm, based on the usage of oriented bounding boxes and the triangle/triangle intersection is used for determining the exact collision point. Using the collision point as a new end of a kinematic chain, new kinematic parameters derived from colliding triangles are calculated. A collision-free robot motion is then enforced by planning collision-free trajectories for these critical points or regions.

1. Introduction

Many techniques used in robotics such as spline-interpolation, collision detection and representation of 3D objects in the virtual reality environment have been actually adopted from computational geometry.

Owing to these methods, new approaches to design, analysis, control, dynamics simulation and visualization of robotic systems and flexible manufacturing systems (FMS) have been enabled. Instead of building real systems, a designer first builds new layouts and configurations in the virtual environment and refines them without actual production of physical prototypes. In this sense, virtual models of robotic systems and FMS can be viewed as a set of virtual robots, machine tools, rotary tables, belt conveyers and other elements put within the virtual environment. One such virtual robotic work cell for palletization is shown in Fig. 1.

Virtual reality 3D models are usually generated using programming languages such as Virtual Reality Modeling Language (VRML) or by using various popular CAD programs (e.g. AutoCAD, Catia or 3D-Studio). So obtained virtual models differ in format and way of creation, but often, they can be converted into concurrent formats and vice versa. In this paper we have adopted a 3D-Studio format (*.3ds) of all considered virtual models.

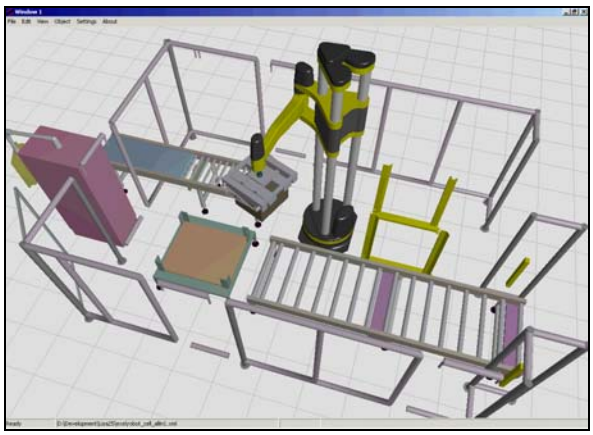

Figure 1. Virtual robotic work cell for palletization (Courtesy of Euroimpianti S.p.a, Italy)

One of interesting aspects in analysis of robotic systems is detection of possible collisions among entities in a virtual world. Good examples are robotic systems with two or more robots sharing their workspace during task execution. Collisions may also occur during pick and place or palletization operations.

Providing that virtual models are accurate, collisions in a virtual environment should occur in the same way as in the real world. By using programs which contain 3D virtual reality models and accompanying dynamic simulators, design, analysis, control, dynamic simulation and visualization of complex robotic systems becomes a fully feasible task (e.g. programs eMPower, Grasp2000, RobotStudio, Flexman [8], [9] or Leonardo [10]).

According to the 3D model taxonomy, 3D models used in this paper are polygonal structured, i.e. polygons form a closed manifold, hierarchical non-convex models undergoing a series of rigid-body transformations [3]. Polygons are made entirely of triangles as hardware accelerated rendering of the

triangles is commonly available in the graphic rendering pipeline.

There are numerous approaches to a collision detection problem which can be mainly grouped into space-time volume intersection, swept volume interference, multiple interference detection and trajectory parameterization [1].

A collision detection algorithm used in this paper belongs to multiple interference detection category, which reduces a general collision detection problem to multiple calls to static interference tests focused on detecting intersections between simple geometrical entities, triangles and oriented bounding boxes (OBB) belonging to 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 to provide real-time collision detection in very complex (high polygon count) virtual environments.

A next logical step is to use information about a location of intersection (i.e. collision) and try to prevent collision by changing the path of one or both elements in collision. Assuming that at least one of the elements tested against collision is a robot, one must know kinematic parameters of the robot to be able to plan correct collision-free robot trajectories. Usually, kinematic parameters of real robots are determined according to the Denavit-Hartenberg (D-H) convention [2], while their concrete values are obtained by very precise measurements.

When collision of such a robot is considered, then 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 which could collide with the environment.

While practically, determination of kinematic parameters for an arbitrary point on the real robot surface is not an accomplishable goal, in the virtual environment this may be resolved in an elegant way by using a kinematics model of the robot derived from the virtual 3D model of the robot.

This paper is organized in the following way. First we describe an algorithm for determination of kinematic parameters for the objects in the virtual environment that have their own kinematic structures. The assumption is made that the geometric structure of all objects is known, either generated from the stereovision or from a 3D modeling. For geometry interpretation triangle meshes are used and all the parameters (joint positions, link lengths etc.) are extrapolated from a 3D description. Then, based on the kinematic parameters obtained for any point (triangle) on the robot surface, inverse kinematics solution is found. This solution will be further used for planning collision-free trajectories referred to the critical point (the point or the area that is actually colliding) in accordance with a fuzzy logic-based collision avoidance strategy.

2. Derivation of kinematic parameters from a 3D robot model

Each virtual object is composed of an arbitrary number of links that form a parent-child hierarchy. There is no limit on the number of child links for a parent, so complex kinematic configurations can be formed out of serial and/or parallel kinematic chains. In this paper we will consider only serial kinematic configurations as most of robotic arms can be represented with such a configuration. Frames (coordinate systems) are assigned to the links sequentially and are either static or dynamic. Dynamic local frames may undergo rigid-body transformations during a simulation in a virtual environment.

A local *k*-th coordinate frame is defined with its center vector and a rotation around a vector. If c_{k_x}, c_{k_y} and c_{k_z} are values at the x, y and z axes of the *k*-th coordinate system, the center vector is defined as $\mathbf{p}_{k_{center}} = \begin{bmatrix} c_{k_x} & c_{k_y} & c_{k_z} \end{bmatrix}$. Similarly, θ_k is a rotation angle around vector $\mathbf{p}_{k_{rotation}} = \begin{bmatrix} r_{k_x} & r_{k_y} & r_{k_z} \end{bmatrix}$.

Respective homogenous transformation matrices for $\mathbf{p}_{kcenter}$ and $\mathbf{p}_{k_{rotation}}$ can be given as,

$$\mathbf{T}_{k_C} = \begin{bmatrix} 1 & 0 & 0 & c_{k_x} \\ 0 & 1 & 0 & c_{k_y} \\ 0 & 0 & 1 & c_{k_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

and

$$\mathbf{T}_{k_R} = \begin{bmatrix} x^2(1-\cos\theta_k)+\cos\theta_k & xy(1-\cos\theta_k)-z\sin\theta_k & xz(1-\cos\theta_k)+y\sin\theta_k & 0 \\ xy(1-\cos\theta_k)+z\sin\theta_k & y^2(1-\cos\theta_k)+\cos\theta_k & yz(1-\cos\theta_k)-x\sin\theta_k & 0 \\ xz(1-\cos\theta_k)-y\sin\theta_k & yz(1-\cos\theta_k)+x\sin\theta_k & z^2(1-\cos\theta_k)+\cos\theta_k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Note that c_{k_x}, c_{k_y}, c_{k_z}, r_{k_x}, r_{k_y}, r_{k_z} and θ_k are fixed (geometric) parameters that depend only on a manipulator geometric 3D structure. Transformation for the *i*-th joint dynamic frame is defined as:

$$\mathbf{T}_{i_q} = \begin{bmatrix} \cos\beta_i \cdot \cos\gamma_i & -\sin\gamma_i & \sin\beta_i & c_{i_x} \\ \sin\gamma_i & \cos\alpha_i \cdot \cos\gamma_i & -\sin\alpha_i & c_{i_y} \\ -\sin\beta_i & \sin\alpha_i & \cos\alpha_i \cdot \cos\alpha_i & c_{i_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where α_i, β_i and γ_i are roll-pitch-yaw (RPY) angles denoting rotations around the x, y and actual z-axis with angles γ_i, β_i and α_i, respectively. A vector $\mathbf{p}_{i_c} = \begin{bmatrix} c_{i_x} & c_{i_y} & c_{i_z} \end{bmatrix}$ is a translation vector of the origin of the i-th joint local frame.

Finally, transformation from the n-th local frame to the global frame can be expressed as follows:

$${}^0\mathbf{T}_n = \prod_{k=0}^{n} \left(\mathbf{T}_{k_C} \cdot \mathbf{T}_{k_R} \cdot \mathbf{T}_{k_q} \right) = \prod_{k=0}^{n} \mathbf{T}_{k_l} \quad , \quad (4)$$

where k is index of all local frames. If index i is involved with dynamic local frames, matrix $\mathbf{T}_{k_q}$ is defined as $\mathbf{T}_{i_q}$ (see (3)), otherwise it assumes the form of the unity matrix.

The solution of a direct kinematics problem (D-H parameters) may be now derived from (4).

Parameter b, representing the link length in y-axis direction has been added to D-H parameters to generalize the solution. Accordingly the order of transformations between a link frame *i-1* and a link frame i is a rotation around axis $\mathbf{z}^{i-1}$ with angle $\boldsymbol{\theta}_i$, translation along the $\mathbf{z}^{i-1}$ with displacement d_i, translation along the $\mathbf{x}^i = \mathbf{x}^{i-1}$ with displacement a_i, translation along the $\mathbf{y}^{i-1}$ with displacement b_i and rotation along $\mathbf{x}^{i-1}$ with angle α_i.

Because of the ambiguity of the trigonometric solutions, to get the DH parameters $\begin{bmatrix} a & b & d & \alpha & \theta \end{bmatrix}$ for the i-th link from the joint frame transformation matrix ${}^{i-1}\mathbf{T}_i$ a correct solution has to be chosen from a set of possible solutions $S = \left\{ \alpha^1, \alpha^2, \alpha^3, \theta^1, \theta^2, \theta^3 \right\}$:

$$\alpha_i^1 = \operatorname{atan2}\left({}^{i-1}\mathbf{T}_i(3,2), {}^{i-1}\mathbf{T}_i(3,3) \right) \quad (5)$$

$$\alpha_i^2 = \operatorname{atan2}\left(-{}^{i-1}\mathbf{T}_i(2,3), {}^{i-1}\mathbf{T}_i(2,2) \right) \quad (6)$$

$$\alpha_i^3 = \operatorname{atan2}\left({}^{i-1}\mathbf{T}_i(1,3), -{}^{i-1}\mathbf{T}_i(1,2) \right) \quad (7)$$

$$\theta_i^1 = \operatorname{atan2}\left({}^{i-1}\mathbf{T}_i(2,1), {}^{i-1}\mathbf{T}_i(1,1) \right) \quad (8)$$

$$\theta_i^2 = \operatorname{atan2}\left(-{}^{i-1}\mathbf{T}_i(1,2), {}^{i-1}\mathbf{T}_i(2,2) \right) \quad (9)$$

$$\theta_i^3 = \operatorname{atan2}\left({}^{i-1}\mathbf{T}_i(1,3), -{}^{i-1}\mathbf{T}_i(2,3) \right) \quad (10)$$

If there is no solution from the set S for equations (5)-(10), matrix ${}^{i-1}\mathbf{T}_i$ can be decomposed by inserting a transformation of an auxiliary frame $\mathbf{T}_{i^*}$:

$${}^{i-1}\mathbf{T}_i = {}^{i-1}\mathbf{T}_{i^*} \cdot {}^{i^*}\mathbf{T}_i \quad (11)$$

The auxiliary frame should be chosen so that satisfies ${}^{i^*}\mathbf{T}_i(3,1) = 0$.

The 3D model of the robot Kuka Kr150 has been used to illustrate derivation of D-H parameters from the model. There are seven local frames - six joint frames and a base frame (Fig. 2). A created kinematic chain is shown in Fig. 3 and the corresponding D-H parameters are displayed in Table 1.

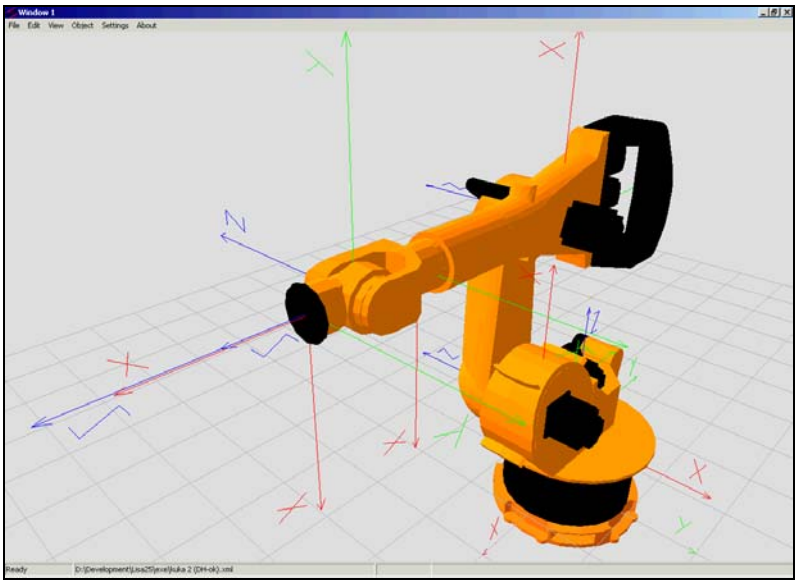

Figure 2. A 3D model of the robot Kuka Kr150 with highlighted joint coordinate frames

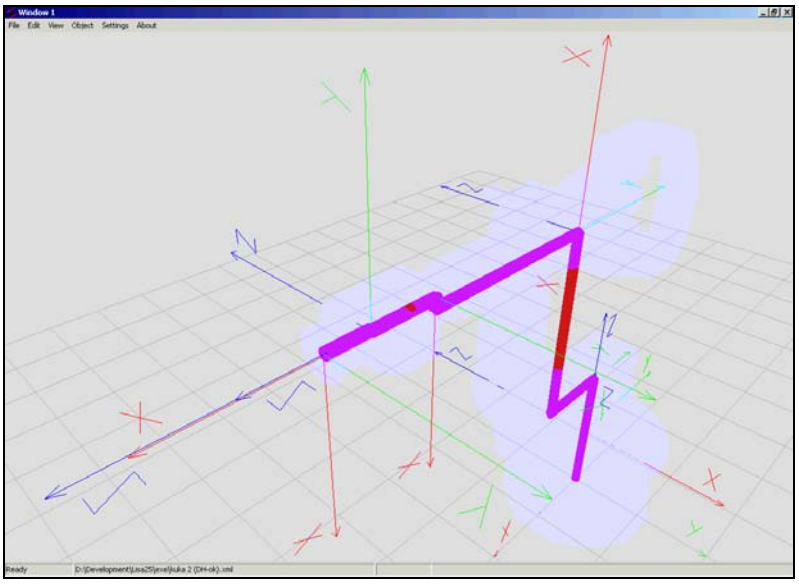

Figure 3. A created kinematic chain for the robot Kuka Kr150

Table 1. D-H parameters of the robot Kuka Kr150

a	b	d	α	ϑ
0.000	0.000	0.000	0.000	0.000
0.000	0.000	1.919	0.000	0.000
2.132	0.050	1.892	1.571	0.000
-0.057	5.051	0.050	0.000	0.000
4.592	0.257	0.000	-1.571	0.000
1.510	0.005	-0.028	0.000	0.000
0.005	0.964	0.029	1.571	1.571

3. Collision detection

Oriented bounding boxes (OBB) are used to determine the distance and collision between different objects at the first hierarchical stage. As it has been proved in [5], overlaps between OBBs are rapidly determined by performing 15 simple axis projection tests. By descending down the generated OBB tree (hierarchy) a search for the collision point becomes narrower, which finally allows the exact collision point determination with triangle/triangle intersection test performed on the final overlapping OBB nodes [4]. How descent will go far down the OBB tree or when the triangle/triangle test will be used instead of OBB overlap check can be specified depending on the available computational time and the complexity of the 3D model.

Once the triangle/triangle intersection test has given the exact collision point, it has also given the colliding link to which it belongs. For more than one link in collision, the preferred one has the highest level in the object hierarchy. A new D-H kinematic model is generated from equations (1)-(3) with the given collision point serving as the origin of the last "joint" local frame. Based on this model, inverse kinematics for this collision point can be calculated.

4. Collision avoidance strategy

Although collision detection is a very important part of a space-time analysis of studied robotic systems, collision avoidance is even more important goal when real-time operating conditions are concerned. In the adopted concept, collision avoidance is done in two stages, first a collision-free path is calculated with respect to the known static objects (it allows the usage of an off-line trajectory planner), and then an on-line trajectory planner is used to generate a collision-free path with respect to the dynamic objects. In such a concept unknown static objects are treated as they were dynamic objects.

Regarding the precision of collision checks, an off-line trajectory planner may operate with a deeper level-of-detail (LOD) than an on-line trajectory planner, due to a different amount of the computational time available. During an on-line search for a collision-free path, a progressive LOD approach is used. Objects that are considered far from each other are tested only in the first level of the OBB tree hierarchy. As objects are approaching, LOD increases and deeper level OBB in the hierarchical tree are checked against collision (see Fig. 4).

Further improvement of the collision-free path search is made by reducing a number of checks.

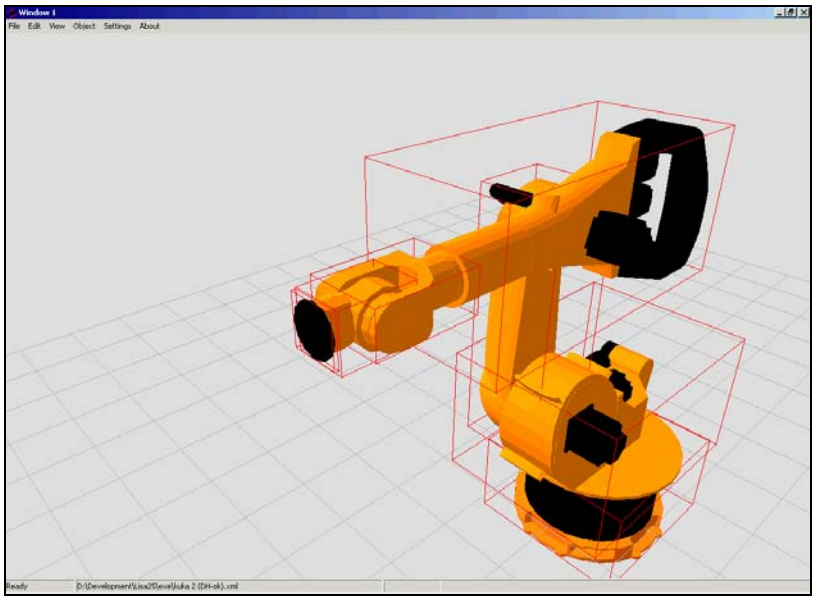

Figure 4. Second level oriented bounding boxes (OBB) for the robot Kuka Kr150

Normal vectors projection of an object face to a relative velocity vector of the object must be positive, as proposed in [6], or more recently in [7], otherwise the object's face is not checked against collisions.

With the knowledge how far the objects are and how rapidly they move, a time interval when collision can possibly take place can be determined. An exact amount of a trajectory correction, magnitude and direction, is determined by a fuzzy logic based collision avoidance strategy using object proximity and object velocity as its inputs.

In the vicinity of a studied robot a fuzzy potential field is used to determine possible collisions with the environment. Input membership functions for the object proximity are "very near", "near", "medium" and "far", while "stop", "slow", "medium" and "fast" represent the input membership functions for the object velocity. While calculating an exact amount of a trajectory correction, magnitude and direction, the fastest evasion and minimal joint movement criterion is favored.

Volume sweep checks are made and possible collision points are used to get new kinematic parameters from the 3D model (Fig. 5). Updated trajectory points are processed in the on-line trajectory planner and searches for the fastest possible collision-free path around the object are made. When a continuous path planning has been used as an off-line trajectory planning method, return to a previous position is enforced, if blocking object has moved. In case such a movement is not possible or a point-to-point path planning has been used, then a return to a reference trajectory is achieved with a minimum motion.

5. Conclusions

There are many robotic applications where collisions of robots with the environment may occur – palletization, pick and place or assembly operations are very good examples.

Regarding collision prevention, 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 which may collide with the environment.

While determination of kinematic parameters for an arbitrary point on the real robot surface is practically non-accomplishable goal, in the virtual environment this may be resolved in an elegant way by using a kinematics model of the robot derived from the virtual 3D model of the robot. For this purpose, a method for generation of a kinematics model from the virtual 3D model is described and illustrated for the industrial robot Kuka Kr150.

By using so obtained kinematic parameters, a new kinematic chain is generated with the collision point serving as the end of the chain. Thus any point of the robot including all points on the robot surface can be controlled and its trajectory can be planned.

A collision detection method used in this paper belongs to the multiple interference detection category,

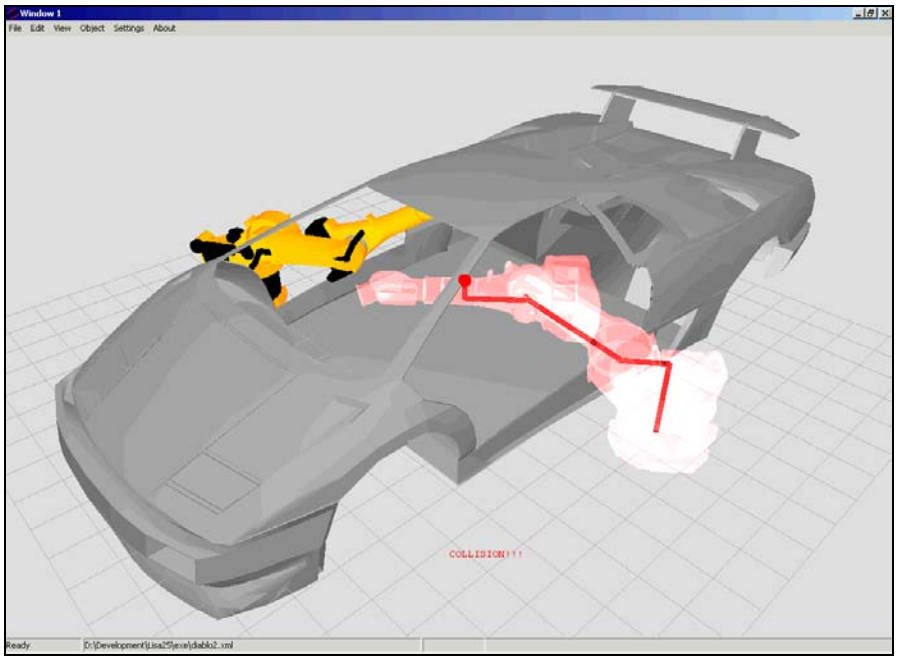

Figure 5 . Collision with a static object - a kinematic chain is generated with the collision point serving as the end of the chain.

which reduces a general collision detection problem to multiple calls to static interference tests focused on detecting intersections between simple geometrical entities, triangles and oriented bounding boxes (OBB).

The more precise the 3D model is, the better collision detection and collision avoidance will be. But, for real time applications, fuzzy logic based collision avoidance strategy may successfully resolve the ambiguity of space-time relations at the minimal cost of the collision avoidance precision.

Future work will focus on getting experimental results in the laboratory. Object movements and positions will be observed by the stereovision while on-line implementation of the developed algorithm will be used for real-time collision avoidance.

6. Acknowledgements

The work described in this paper was performed within the framework of the 036044 research project entitled "Control of robotized plants" which was supported by a grant from the Ministry of Science and Technology of the Republic of Croatia.

7. Literature

[1] P. Jimenez, F. Thomas, and C. Torras, "3D collision detection: A survey", Computer & Graphics, 2001, vol. 25, pp. 269--285.

[2] J. Denavit, R.S. Hartenberg; A Kinematic Notation for Lower Pair Mechanisms Based on Matrices, *J. Appl. Mech. ASME*, June 1955, pp. 215-221.

[3] M. Lin and S. Gottschalk ;Collision Detection between Geometric Models: A Survey, Proceedings of IMA Conference on Mathematics of Surfaces 1998.

[4] Real-Time Rendering; Tomas Möler and Eric Haines. A K Peters Ltd 1999

[5] OBB-Tree: A Hierarchical Structure for Rapid Interference Detection; S. Gottschalk, M. Lin, and D. Manocha. Proc. ACM SIGGRAPH, 1996.

[6] G. Vanecek, Jr. Back-face culling applied to collision detection of polyhedra. Journal of Visualization and Computer Animation, 5(1), 1994.

[7] S. Redon, A. Kheddar and S. Coquillart. Hierarchical Back-Face Culling for Collision Detection. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. October 2002.

[8] Kovačić Z.;Bogdan S.; Reichenbach T.; Smolić-Ročak N.; Birgmajer B. FlexMan - A Computer-integrated Tool for Design and Simulation of Flexible Manufacturing Systems, CD-ROM Proceedings of the 9th Mediterranean Conference on Control and Automation Control, TM2-B, Dubrovnik, 2001.

[9] Smolić-Ročak N.; Bogdan S.; Kovačić Z.; Reichenbach T.; Birgmajer B. Modeling and Simulation of FMS Dynamics by Using VRML", CD-ROM Proceedings of the b'02 IFAC World Congress July 21-26, 2002 Barcelona, Spain.

[10] Kovačić Z.; Bogdan S.; Petrinec K.; Reichenbach T.; Punčec M. LEONARDO - The Off-line Programming Tool for Robotized Plants, CD-ROM Proceedings of the 9th Mediterranean Conference on Control and Automation Control, WM2-C, Dubrovnik, 2001.