Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 20 (2014) 62 – 66

www.elsevier.com/locate/procedia

2nd International Conference on Ramp-Up Management 2014 (ICRM)

Efficient collision avoidance for industrial manipulators with overlapping workspaces

Philipp Ennen*, Daniel Ewert, Daniel Schilberg, Sabina Jeschke

Institute of Information Management in Mechanical Engineering (IMA)
and Center for Learning and Knowledge Management (ZLW)
and Assoc. Institute for Management Cybernetics e.V. (IfU)

* Corresponding author. *E-mail address:* philipp.ennen@ima-zlw-ifu.rwth-aachen.de

Abstract

"This paper introduces an efficient collision watchdog predicting impacts between fast moving industrial robots. The presented approach considers the manipulator states in a three dimensional space. Tailored bounding volumes allow fast collision detection and distance calculations. The watchdog makes use of the internal rotary sensors of each robot to build an integrated world representation. Based on this information it is able to monitor the non-predictable behavior of all involved robots."

Selection and peer-review under responsibility of the International Editorial Committee of the "2nd International Conference on Ramp-Up Management" in the person of the Conference Chair Prof. Dr. Robert Schmitt
Keywords: collision detection, industrial manipulators, overlapping workspaces

1. Introduction

In high-wage countries industrial production is often organized as an automated process. The setup of such a process is complex and cost-intensive, so automated processes amortize only in case of high-batch sizes. This contradicts the wishes of customers for tailored products which results in a large amount of manual work to achieve the desired flexibility [1]. Hence it becomes necessary to work on easily-adaptive and automated production systems.

An essential approach is given by self-optimizing production systems which are based on multiple axis industrial robots. Each robot within such a production system is autonomous regarding his motions, that is, it calculates its motions by itself. This may lead to a non-predictable behaviour of the ensemble of robots. If different manipulators are moving within the same workspace this possibly results in collisions.

With respect to non-predictable behaviour robot states cannot be determined in advance. On that account collision detection has to be calculated a posteriori. So the basic requirements for the detection is time efficiency and a prospective decision function. The main task of the algorithm is to provide impacts inside robot ensembles. Therefore a large number of collision tests are necessary by checking all independent moving components. Possible combinations can be calculated by $(n-1)!$ for n components. This equation is derived from following: If a robot ensemble consists of n independent components, the first component can collide with *n-1* others, the second one can collide with *n-2* additional others and the third one with *n-3*. Finally the series $(n-1)!$ results.

Nomenclature

n, m	bounding volumes/lines
r_n, r_m	radius of bounding volume *n, m*
P_n, P_{n+1}	position of origin joint *n*, joint *n+1*
μ_n	parameter of line *n*
d_{nm}	minimum distance between line n and m
t_r	response time
$\overrightarrow{v_n}$, $\overrightarrow{v_m}$	maximum velocities of bounding volumes n, m
τ	timestamp

Selection and peer-review under responsibility of the International Editorial Committee of the "2nd International Conference on Ramp-Up Management" in the person of the Conference Chair Prof. Dr. Robert Schmitt
doi:10.1016/j.procir.2014.05.032

ΔT	time between two timestamps
κ	numerical condition
S	safety reserve
P_{crit}	Point of maximum velocity in a bounding volume

In a nutshell: Applying an ensemble of two six axis robots with $\boldsymbol{n = 12}$ independently moving components, the collision detection have to take care of impacts between $\boldsymbol{k = 39.916.800}$ possible combinations of components. This slows down collision calculations. Additional robot states have to be computed by values of internal rotary sensors. Therefore joint positions have to be converted into Cartesian coordinates. This transformation also have an influence on performance. Consequently due to a high number of necessary tests, coordinate transformation and fast moving robots, the algorithm has to be very efficient in calculation and respecting movements following in near future.

2. State of the art

In recent years many algorithms for collision detection have been developed and optimized, especially within the fields of computer graphics. As a result the classifications into *discrete* and *continuous* methods has been established [2]. For this purpose most famous algorithms for both classifications are be presented.

The continuous method (a priori) makes use of predicted trajectories of physical objects. This way the point of collision can be detected in advance. Literature here distinguishes between different types: the algebraic equation solving approach [3], the swept volume approach [4], kinetic data structures approach [5], and adaptive bisection approach [6].

The algebraic equation solving method makes use of calculating the point in time of collision. Therefore equations describing relevant trajectories have to be explicitly solved, often by using numerical techniques. The swept volumes approach is based on calculated swept volumes of all moving objects. If those volumes interference with each other or the environment, a collision is detected. The kinetic data structure approach makes use of elementary conditions. These conditions, also called certificates [5], describe criteria for object movements. If a certificate fails a collision might occur. The adaptive bisection method is based on conservative state sampling. Here, the distance, velocity and direction of movement is interpreted as the estimated time of impact between objects. If a lower bound is reached, a collision is detected.

Discrete methods (a posteriori) sample object trajectories and repeatedly apply a static interference test. In case of convex polytopes the problem can be described as a linear programming problem as follows: Two convex polytopes only overlap if no separating plane exists. A sufficient criterion for a separating plane is all vertices of the first polytopes lying in one halfspace of the plane and those vertices of the second polytopes lying in the second halfspace. Approaches for calculating the separating plane can be found in [7] [8].

In case of non-convex polytopes a preprocessing step is necessary which decomposes the polytopes into primitive convex entities [9]. This increases the number of essential pairs of objects that need to be checked for contact. Bounding volumes (BV) are established to reduce the number of pairs of objects by approximating them. In a hierarchical tree model the root BV contains all primitives of a model, each children's BVs contains separated primitives enclosed by the parent and the leaf usual contains one single primitive. Common used BV primitives are spheres [10] [11], axis-aligned bounding boxes [12] [13] [14] and orientated bounding-boxes [15] [16] [17]. Another approach for collision detection and minimum separation distance calculation is based on the Minkowski sum of two objects A and B. Cameron and Culley [18] have shown the minimum separation distance is equal to the minimum distance of the origin of the Minkowski sum $A \otimes -B$ to the surface. An algorithm solving this is provided in [19], but it has to be considered that the Minkowski sum of two convex polytopes can have $O(n^2)$ features [20], making the algorithm inefficient for such cases.

3. Collision avoidance

The continuous methods presented in section 2 are based on the equations of movement of every component involved. However, most proprietary robotic systems do not provide this information. Therefore the collision detection must be based on discrete methods: A watchdog is monitoring the robotic movements and triggers an emergency stop if it becomes necessary. Basically, the following guiding principle applies: For industrial manipulators a fast and safe algorithm for collision detection and avoidance is more important than exact analytical results.

Therefore it is feasible to approximate the actual robot geometry with simplified structures which completely enclose the robot. In the present case this is realized as follows: For each axis a finite line $\overrightarrow{s_n}$ is spanned through the two adjacent joints P_n and P_{n+1}. Also, a radius r_n is introduced which describes the maximum distance between the line and the surface of the related robot component (see Figure 1). The

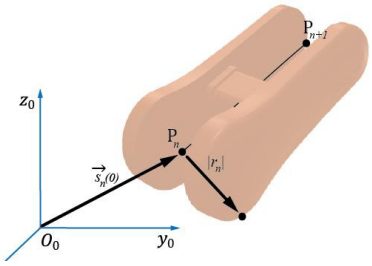

Figure 1 Description of a single robot component

describing equation for the line $\overrightarrow{s_n}$ can be found in (2).

$$\vec{\mathbf{s}}_n(\mu) = \mathbf{P}_n + \mu_n \cdot (\mathbf{P}_{n+1} - \mathbf{P}_n) \tag{2}$$

The point $\boldsymbol{P_n}$ represents the origin of a joint and is related to the world coordinate system O_o. Therefore a coordinate

transformation is required. This can be done using the Denavit-Hartenberg representation [21].

Once the robot components are described, following conclusion is allowed: a collision exists if and only if the minimal distance between robot lines $\overrightarrow{s_n}$ and $\overrightarrow{s_m}$ is smaller than the sum of the related radiuses r_n and r_m. For fast moving robots we have to consider the whole systems response time. Therefore equation (3) is introduced, with *n* and *m* as tested components, d_{nm} the minimal distance between those components, $\overrightarrow{\boldsymbol{v_n}}$ and $\overrightarrow{\boldsymbol{v_m}}$ as respective maximum velocities, t_r as response time, *S* as a safety reserve and τ as the timestamp.

$$d_{nm}(\tau) - 2t_r \cdot \left\| \vec{\mathbf{v}}_n(\tau-1,\tau) - \vec{\mathbf{v}}_m(\tau-1,\tau) \right\|_2 - S < r_n + r_m \quad (3)$$

Figure 2 shows the main signal flow with all functionalities affecting the response time of the system. These include sensors and the communication units, where signal flow is managed between all involved entities, the necessary coordinate transformations, and subsequently the actual collision detection. In consequence of an imminent collision, a suitable reaction is invoked and send to the actors.

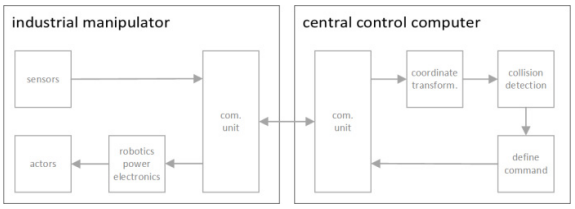

Figure 2 Signal flow

3.1. Maximum velocity calculation

Given a point $\boldsymbol{P_{n,crit}}$ at two timestamps τ and $\tau - 1$ with ΔT as relating time difference, the velocity in-between $\boldsymbol{P_{n,crit}}|_{\boldsymbol{\tau}}$ and $\boldsymbol{P_{n,crit}}|_{\boldsymbol{\tau-1}}$ can be linearly approximated as seen in equation (4).

$$\vec{\mathbf{v}}_n = \frac{\mathbf{P}_{n,crit}|_{\tau} - \mathbf{P}_{n,crit}|_{\tau-1}}{\Delta T} \quad (4)$$

For a safe estimation of imminent collision only maximum velocities are taken into account. Considering a linear relationship between velocity and distance to the point of initial movements the point $\boldsymbol{P_{n,crit}}$ have to be both located at maximum distance to the origin joint and inside the introduced bounding volume. So in case of the robot description model presented in Figure 1, the critical point is calculable with equation (5).

$$\mathbf{P}_{n,crit} = \mathbf{P}_{n+1} + r_n \cdot \frac{\mathbf{P}_{n+1} - \mathbf{P}_n}{\| \mathbf{P}_{n+1} - \mathbf{P}_n \|_2} \quad (5)$$

3.2. Minimum distance calculation

This section will address how to calculate the minimum distance $d_{nm}(\tau)$ (see equation (3)). The calculation procedure thereby depends on the relative position to each other. Therefore, configuration tests are needed which allow to select the proper procedure. Figure 3 shows the projection of three general configurations.

The minimum distance in case (A) corresponds to the shortest distance between the lines, in case (B) it is found as the shortest distance between a point and a line and in case (C) the minimum distance is equal to the shortest distance between two points.

For case (A) the procedure (5) is valid [22], provided that

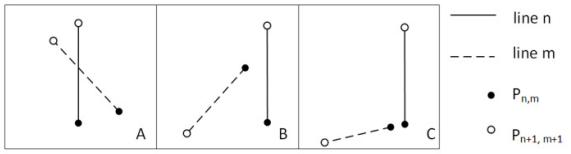

Figure 3 Different test configurations in general

$(\boldsymbol{P_{n+1}} - \boldsymbol{P_n}) \times (\boldsymbol{P_{m+1}} - \boldsymbol{P_m}) \neq \vec{0}$, in other words the two lines have to be skew. However, this limitation is insignificant because in case of *moving* robots exact parallel lines can only occur within one single timestamp. Hence the response time t_r in equation (3) is doubled.

$$d_{nm,A} = \frac{\left\| \det\left[\mathbf{P_n} - \mathbf{P_m} \quad \mathbf{P_{n+1}} - \mathbf{P_n} \quad \mathbf{P_{m+1}} - \mathbf{P_m} \right] \right\|_2}{\left\| (\mathbf{P_{n+1}} - \mathbf{P_n}) \times (\mathbf{P_{m+1}} - \mathbf{P_m}) \right\|_2} \quad (5)$$

The minimum distance in case (B) is calculated by (6) [22] .

$$d_{nm,B} = \frac{\left\| (\mathbf{P_n} - \mathbf{P_m}) \times (\mathbf{P_{n+1}} - \mathbf{P_n}) \right\|_2}{\left\| \mathbf{P_{n+1}} - \mathbf{P_n} \right\|_2} \quad (6)$$

And the minimum distance in case (C) is calculated by (7) [22].

$$d_{nm,C} = \left\| \mathbf{P_n} - \mathbf{P_m} \right\|_2 \quad (7)$$

3.3. Identifying test configuration

In total, there are nine possible configurations (shown in Figure 4), depending on their relative orientation. For correct distance calculation it is necessary to identify the case of configuration.

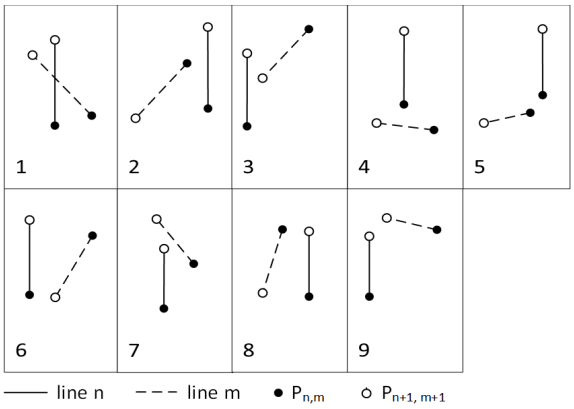

Figure 4 Overview of all test configurations

Assuming case (1) the value of both line-parameters $\mu_{n,m}$ are necessary, which describe the points of applied minimum distance. If they are in between [0,1], case (1) is acknowledged, if both are higher than 1, case (9) is detected. Figure 5 shows all existing combinations in a decision tree.

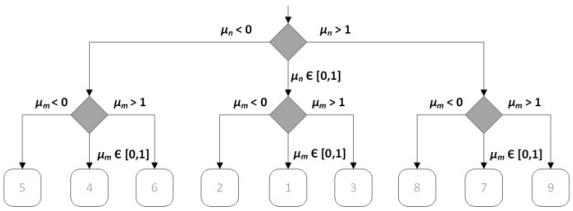

Figure 5 Decision tree for determining current test configuration

For calculating μ_n and μ_m a temporary plane is necessary. In case of μ_n the plane is spanned by $\boldsymbol{P_{m+1} - P_m}$ and $(\boldsymbol{P_{m+1} - P_m}) \times (\boldsymbol{P_{n+1} - P_n})$ (see equation (8)).

$$\vec{q} = \mathbf{P}_m + \alpha \cdot (\mathbf{P_{m+1}} - \mathbf{P_m}) + \beta \cdot (\mathbf{P_{m+1}} - \mathbf{P_m}) \times (\mathbf{P_{n+1}} - \mathbf{P_n}) \quad (8)$$

The intersection of the temporary plane with the line *n* gives μ_n. With $\overrightarrow{x_n} = (\alpha, \beta, \mu_n)^T$ the value of μ_n can be calculated by solving the system of equations in (9).

$$\vec{\mathbf{x}}_\mathbf{n} = \mathbf{A}^{-1}\left[\mathbf{P_n} - \mathbf{P_m}\right], \quad (9)$$

$$\text{with } \mathbf{A} = \left[\mathbf{P_{m+1}} - \mathbf{P_m} \quad (\mathbf{P_{m+1}} - \mathbf{P_m}) \times (\mathbf{P_{n+1}} - \mathbf{P_n}) \quad \mathbf{P_n} - \mathbf{P_{n+1}}\right]$$

3.4. Numerical effects

If both lines *n* and *m* are approaching a parallel configuration, the inverse matrix in equation (9) and the distance calculation (equation (5)) cannot be reliably determined. Therefore a watchdog for the numerical condition of the problem is necessary. Since the matrix ***A*** (see equation (9)) describes the whole geometrical situation, the condition can be calculated by solving equation (10) [23].

$$\kappa(A)_\infty = \left\|\mathbf{A}^{-1}\right\|_\infty \left\|\mathbf{A}\right\|_\infty \quad (10)$$

If the condition becomes infinity, ***n*** and ***m*** are exactly parallel, otherwise equation (5) and (9) can be solved. However, if they are approaching a parallel configuration, numerical errors during distance calculation can occur. Therefore the numerical condition should be a small value. If the value is exceeding an upper limit, the collision calculation has to be skipped. However, this does not affect the collision detection in practice: Experiments on two industrial robots (ABB IRB 120) have shown that extremely bad conditioned robot states can only occur within one timestamp (see Figure 6).

So in case of a doubled response time (see equation (3)) the skipped timestamp is considered by approximating current robot state in a previous timestamp.

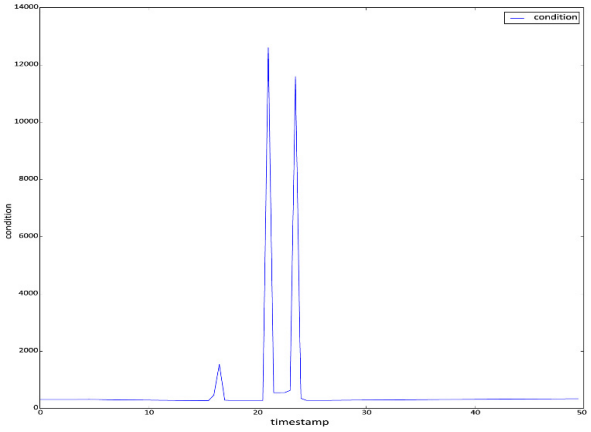

Figure 6 Numeric condition of moving robot components

3.5. Relevant test candidates

Not every combination of lines needs to be tested. Directly adjacent lines, linked with a joint, are overlapping all the time. So as a minimum condition, *n* and *m* are not allowed to be connected by a joint. Also depending on robot location there can be geometrical reasons which eliminate possible collisions. By considering relevant test candidates only, the time efficiency of the collision watchdog can be dramatically increased.

4. Evaluation

The presented approach was evaluated using two six axis robots (ABB IRB 120, see Figure 7). A socket communication based on TCP/IP is used for sending sensor information from the robots to a central control computer. Here the necessary transformations are calculated using the ROS framework [24].

A python based program is working as a watchdog and conducts collision detection. If the robots get into impact, the watchdog triggers an emergency stop. Considering the whole system a response time of t_r = 0.5 sec is measured. Therefore in this use case a forward looking collision detection is mandatory for safe movements.

Test runs were based on the decision tree in Figure 5. They were successfully performed for a complete statement coverage of this decision tree.

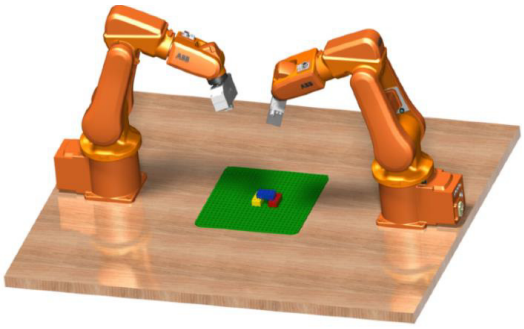

Figure 7 Robotics cell based on two ABB IRB 120

5. Summary

We presented a hybrid approach of a both continuous and discrete collision detection. Considering current velocities allows to approximate future collision states within following timestamp. By doing this the algorithm provides collision detection for fast moving robots in case of non-negligible response time. Relevant test candidates for collision are verified by checking them for interference.

For choosing the right calculation rule an analysis of current test configuration is necessary. Therefore a decision tree is compiled. Depending on the result, the appropriate distance calculation is performed. A concluding interference test considering response time, speed and current minimal distance gives a final statement about collision. This algorithm is successfully tested on a robot cell consisting of two industrial manipulators.

6. Outlook

An optimal approach for collision detection will prevent collisions by calculating collision-free trajectories a priori. Therefore the presented approach can be integrated into numerical calculation of inverse kinematics whereby impacts could be predicted in advance. However, for non-predictable behaviour this approach cannot be realised. For a safe algorithm the approximation of future robot states have to be credible and a good estimation.

Therefore adding constraints like robotics kinematic can result in more precise position approximations. Additional while the presented approach detects collisions, it will not resolve the collision state. On that account an evasive action is necessary. Considering the contact point of imminent collision the initial evasive direction should be normal to this collision point. Using this as a boundary condition a new path can be calculated.

References

[1] Brecher, C., Klocke, F., Schmitt, R., Schuh, G., 2007. Excellence in Production. Apprimus, Aachen.

[2] Redon, S., Kim, Y.J., Lin, M.C., Manocha, D., 2005. Fast continuous collision detection for articulated models. Journal of Computing and Information Science in Engineering 5 (2), 126–137.

[3] S. Redon, A. Kheddar, S.Coquillart, 2000. An Algebraic Solution to the Problem of Collision Detection for Rigid Polyhedral Objects. INRIA - Rocquencourt, San Francisco.

[4] Abdel-Malek, K., Yang, J., Blackmore, D., Joy, K., 2006. Swept Volumes: Fundation, Perspectives, and Applications. International Journal of Shape Modeling 12 (1), 87–127.

[5] J. Basch, J. Erickson, L.J. Guibas, J. Hershberger, L. Zhang, 2003. Kinetic Collision Detection Between Two Simple Polygons. SiRF Technology Inc.

[6] F. Schwarzer, M. Saha, J.-C. Latombe, 2005. Adaptive Dynamic Collision Checking for Single and Multiple Articulated Robots in Complex Environments.

[7] R.Seidel (Ed.), 1990. Linear programming and convex hulls made easy. ACM, 211-215.

[8] Chung, K., Wang, W., 1996. Quick elimination of non-interference polytopes in virtual environments, in: , Virtual Environments and Scientific Visualization'96. Springer, pp. 64–73.

[9] Chazelle, B., 1984. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. SIAM Journal on Computing 13 (3), 488–507.

[10] Hubbard, P.M., 1996. Approximating polyhedra with spheres for time-critical collision detection. ACM Transactions on Graphics (TOG) 15 (3), 179–210.

[11] S. Quinlan (Ed.), 1994. Efficient distance computation between non-convex objects. IEEE, 3324-3329.

[12] Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B., 1990. The R*-tree: an efficient and robust access method for points and rectangles. ACM.

[13] M.Held, J.T. Klosowski, J.S.B. Mitchell (Ed.), 1995. Evaluation of collision detection methods for virtual reality fly-throughs. Citeseer, 205-210.

[14] M.Ponamgi, D. Monacho, M.Lin (Ed.), 1995. Incremental algorithms for collision detection between solid models. ACM, 293-304.

[15] S.Gottschalk, M.Lin, D.Manocha (Ed.), 1996. OBBTree: A hierarchical structure for rapid interference detection. ACM, 171-180.

[16] G. Barequet, B. Chazelle, L. Guibas, J. Mitchell, A. Tal (Ed.), 1996. Boxtree: A hierarchical representation for surfaces in 3D. Wiley Online Library, 387-396.

[17] Gottschalk, S., 2000. Collision queries using oriented bounding boxes.

[18] S. Cameron, R.H. Culley (Ed.), 1986. Determining the minimum translational distance between two convex polyhedra. IEEE, 591-596.

[19] Gilbert, E.G., Johnson, D.W., Keerthi, S.S., 1988. A fast procedure for computing the distance between complex objects in three-dimensional space. Robotics and Automation, IEEE Journal of 4 (2), 193–203.

[20] Dobkin, D., Hershberger, J., Kirkpatrick, D., Suri, S., 1993. Computing the intersection-depth of polyhedra. Algorithmica 9 (6), 518–533.

[21] Niku, S.B., 2001. Introduction to robotics: analysis, systems, applications. Prentice Hall New Jersey.

[22] Bronshtein, I.N., Semendyayev, K.A., Musiol, G., Muehlig, H., 2007. Handbook of mathematics. Springer.

[23] Dahmen, W., Reusken, A., 2008. Numerik für Ingenieure und Naturwissenschaftler. Springer.

[24] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng (Ed.), 2009. ROS: an open-source Robot Operating System.