Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 11 (2017) 241 – 248

Procedia MANUFACTURING

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy

Automatic Path Planning of Industrial Robots Comparing Sampling-Based and Computational Intelligence Methods

Lars Larsen[a], Jonghwa Kim[b]*, Michael Kupke[a], Alfons Schuster[a]

[a]German Aerospace Center (DLR),Am Technologiezentrum 4, 86159 Augsburg, Germany
[b]University of Science & Technology (UST),217 Gajeong-ro, 34113 Daejon, Korea

Abstract

In times of industry 4.0 a production facility should be "smart". One result of that property could be that it is easier to reconfigure plants for different products which is, in times of a high rate of variant diversity, a very important point. Nowadays in typical robot based plants, a huge part of time from the commissioning process is needed for the programming of collision free paths. This mainly includes the teach-in or offline programming (OLP) and the optimization of the paths. To speed up this process significantly, an automatic and intelligent planning system is necessary. In this work we present a system which can plan paths for industrial robots. We compare widely used sampling-based methods like PRM or RRT with Computational Intelligence (CI) based methods like genetic algorithms.

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing

Keywords: Path Planning, Industrial Robots, Intelligent and Flexible Manufacturing, Genetic Algorithm

1. Introduction

Industry 4.0 robotic facilities should be flexible and easy to reconfigure. The term describes the change of production facilities to smart factories [1]. State of the art robotic cells are programmed for a special task which

* Corresponding author. Tel.: +82-42-865-2475; fax: +82-42-865-2329.
E-mail address: kim@ieee.org

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing
doi:10.1016/j.promfg.2017.07.237

clearly defines all necessary steps and can only react to small deviations in the process. Reconfiguration means downtime of the production and programing of the robots which can either be done on the real hardware or in an offline programming tool like DELMIA or RobCAD. In both cases this means the facility cannot be used. According to the complexity of the new process the programming can need quite a long time.

2. Related Work

In general there are not very many publications in the area of 3D path planning for industrial robots. Suh et al. [2] and Chen et al. [3] present a system for the automatic path planning of spray robots for a consistent application. However the method is very specialized and only applicable for the proposed scenario. Ting et al. [4] and Klanke et al. [5] show an approach using wave expansion method. Oh et al. [6] combine a support vector machine with a Rapidly Exploring Random Trees (RRT) algorithm for a 6-DOF industrial robot. Qin et al. [7] use a randomized parallel search algorithm for a PUMA 200 robot.

Compared to the small number of publications for realistic path planning problems with actual industrial robots there is a huge amount of publications for robots with a high degree of freedom (DOF). The primary goal was to find complete planners which means that the planner will find a path if one exists. Due to the high computation amount these kinds of planners only work for robots with a small DOF [8]. Today sampling-based planners are often used, whose advantage is that not the complete collision free room is constructed in advance. Instead the space is just examined at specific positions called samples for collisions. Sampling-based algorithms have been treated in a great number of publications like [9] [10] [11]. The most famous are Probabilistic Roadmaps (PRM) and RRT.

In general there are a manageable number of publications using Genetic Algorithms (GAs) for path planning. A popular use case of GA is in the area of mobile robots [12] [13]. In this field the planning is reduced to a 2D problem because the robots are not able to move in z-direction. Another application of GAs can be found in the area of 2D manipulators [14] [15]. In [16] [17] [18] GAs are used for path planning of industrial robots. Mostly the mapping of the robot in the simulation is done with a very reduced model which just represents the kinematic of the robots and not the real 3D structure because the collision detection can be very CPU-intensive.

3. Automatic Path Planning of Industrial Robots

3.1. Implementation

In [19], [20]and [21] the CoCo (**Co**llision-free **Co**operation) simulation environment has been introduced and stepwise improved. It has been developed using the C# programming language. For the visualization of 3D objects the Helix Toolkit [22] is used which has been combined with BEPUphysics [23] library for collision detection. To speed up the calculation internally the objects are represented as convex hull (see Fig. 2c). For sampling-based path planning the Open Motion Planning Library (OMPL) [24] has been integrated into CoCo. OMPL itself does not have a visualization or collision detection mechanisms which allows integrating the library into systems that provide these functionalities. In our case the CoCo collision detection callback method is registered in OMPL. OMPL consists of many state-of-the art sampling-based motion planning algorithms like Probabilistic Roadmaps (PRM), Rapidly Exploring Random Trees (RRT), RRTconnect, RRT*, SPArse Roadmap Spanner (SPARS) and many more. For the implementation of GA planners we used the AForge [25] library as basis which is also written in C# and provides a lot of functions for Computer Vision and Artificial Intelligence e.g. neural networks, genetic algorithms, machine learning, etc..

3.2. Path Planning

In general collision-free path planning means: (a) avoid collisions between obstacles and the robots, and (b) optimize the path according to predefined constraints like path length or smoothness. Given that the GAs work in Cartesian space also called SE(3) space we also used that space for the sampling-based planners to have a good comparability.

3.3. Sampling-based Planners

Single Query algorithms: The main characteristic of single query algorithms is that they search a path from a start to a goal configuration without previous knowledge. In general they are faster than multi query algorithms because no preprocessing phase is necessary. For another query in the same scene the algorithm completely has to recalculate even if the start and goal configuration just changed a little bit. The most famous representatives are RRT, Expansive Space Trees (ESTs), Kinematic Planning by Interior Cell Exploration (KPIECE), Potential Field Methods, Search Tree with Resolution Independent Density Estimation (STRIDE), Path-Directed Subdivision Trees(PDSTs) and Fast Marching Treed (FMTs).

Multi Query algorithms: These kinds of algorithms have a preprocessing phase where the configuration space is extensively examined for collision states. Built on that information a data structure is stored which can be used later for the planning phase. At the beginning these kinds of planners are slower than single query algorithms but for new calculation on the same scene the store data can be used. Thereby multi query algorithms are optimal for static environments. Famous representatives are PRMs, SPARS and SPARS2.

3.4. Evolutionary Algorithms

EAs were introduced by John Holland in 1975 [26] and are assigned to the class of stochastic and metaheuristic methods, because they normally don't find the optimal solution for a problem. EA's which are inspired by the nature, can be successfully applied to technical issues. There exist four main distinctions of evolutionary algorithms: genetic algorithms, evolutionary algorithms, genetic programming and evolutionary programming which mainly can be distinguished by the way the chromosomes are represented. EAs are very efficient in optimization problems. In this publication we use them for the path planning of 6-DOF industrial robots. An EA algorithm is working according to the steps shown in Fig. 1., which are explained in detail below.

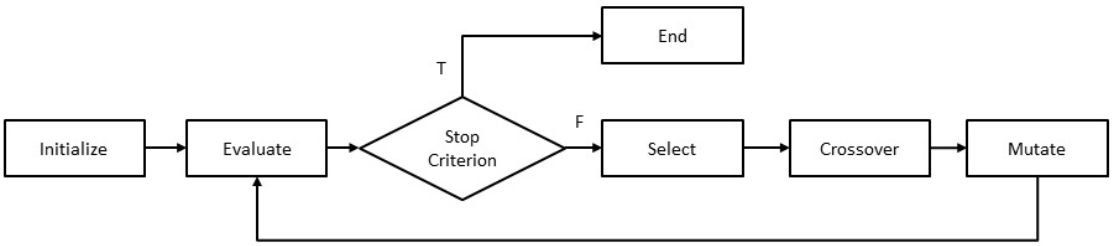

Fig. 1. Sequence of a the Evolutionary Algorithm

Chromosome representation: As already mentioned the success of the genetic algorithm is depending on the representation of the chromosome. In general there are two ways to represent each point of the path. Possibility one would be by the joint angles $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$ of the robot and possibility two by the coordinates x, y, z, a, b, c for each point. Additionally there is the possibility to represent the values as real values or as binary code or gray. In our implementation we used the representation of the points as real values. To represent the path each chromosome stores the points in an array with the following structure: $StartPoint, ViaPoint_1, ViaPoint_2$, …, $ViaPoint_n, GoalPoint$ (see Fig. 2a) at which a $ViaPoint$ persists of x, y and z coordinates.

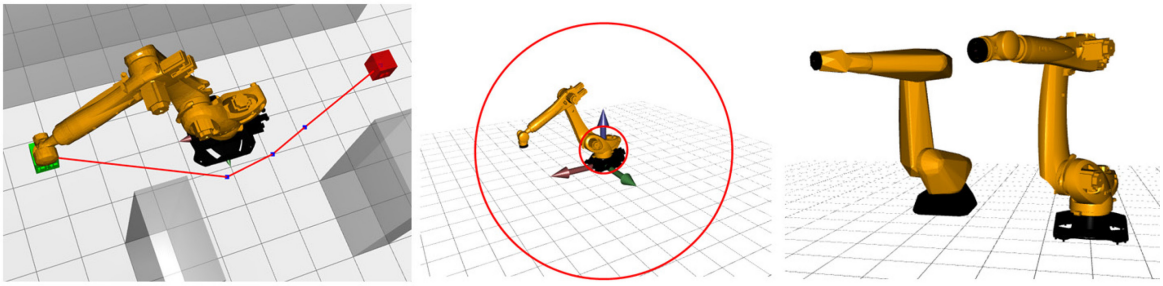

Fig. 2. (a) representation of a path – start point (green box), via points (blue), goal point (red box) (b) simplified workspace of the robot (c) normal and convex hull representation which is internally used for collision detection

Initialization: At the beginning the population is initialized randomly which means in our case that all $ViaPoints$ are initialized with random coordinates. To avoid absurd points which are located outside the workspace of the robot each generated coordinate is proved by the following equation which guarantees that the point is inside the range of the robot, not inside the body of the robot and above the ground.

$$\sqrt{x^2 + y^2 + z^2} > maximumStretchLength\ OR\ \sqrt{x^2 + y^2} < robotCenter\ OR\ z < 0\,,$$

whereas $maximumStretchLenght = 3340$ and $robotCenter = 500$ for the used robot type. For each ViaPoint of the chromosome new coordinates are generated until the equation is applied. Fig. 2b shows the approximation of the workspace. The origin of the coordinate system lies inside the foot of the robot.

During initialization the size of the population must be set which is an important part for the result of the algorithm. For a very large number of populations the algorithm will need a long time to finish, because the number of collisions checks rises with the number of chromosomes. However if the population size is too small the algorithm will run into in a local minimum very fast.

Selection: The selection operator chooses individuals for reproduction. According to their fitness better individuals are copied to the next population and bad individuals die. The fitness of an individual is determined by the objective function which is to be optimized for the individuals. There are different kinds of selection methods like Roulette Wheel-, Rank- or Steady-State Selection.

Crossover: The crossover is important to exchange material between different chromosomes in analogy to reproduction in nature. During crossover two parent chromosomes are crossed to generate a new child. In our implementation we used the single-point crossover operator which is working according to the sketch in Fig. 3. First a seed point is randomly chosen in the range of $ViaPoints$. Afterwards the values of two chromosomes are swapped at the seed point.

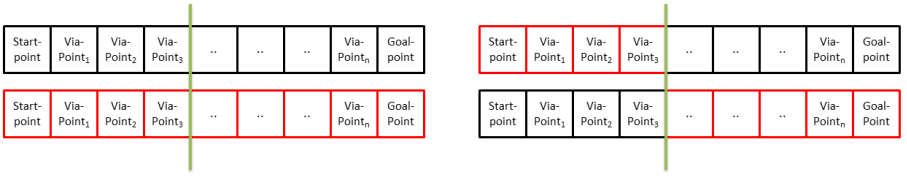

Fig. 3. Crossover operator (a) two chromosomes before crossover (b) two chromosomes after crossover

Mutation: Mutation describes the randomly change of values of the chromosome. It is analog to the biological mutation. During mutation one or more gene values are altered from its initial state. For a genetic algorithm it is

possible to assign the mutation probability. It should be set to a low value because otherwise the search of the algorithm will turn into a primitive random search.

In our implementation we randomly pick one of the $ViaPoints$ of the chromosome and modify its coordinate.

Evaluation: The fitness function is used to summarize the quality of an individual. The value of the fitness is calculated by the objective function. In our implementation the objective function minimizes the path length and guarantees collision avoidance between the robot and obstacle. It is calculated as follows:

$$Fitness = \omega_1 * PathLength + \omega_2 * CollisionCounter$$

The path length is calculated by the Euclidean distance between $Start, Via$ and $Goal$ points. For good comparability of $PathLength$ and $CollisionCounter$ it is important to normalize the values e.g. between 0 and 1. Additionally each addend is weighted by a factor to strengthen or weaken its influence. Due to performance reasons the $CollisionCounter$ value is calculated in two steps. First a ray intersection is calculated for each path segment of the chromosome with the Axis Aligned Bounding Boxes of all obstacles in the scene. For each chromosome the number of ray intersection is counted. If the value is above a threshold (in our case two) the chromosome gets a very small value for its fitness e.g. epsilon. If the number of ray intersection is smaller than two a convex hull collision check is performed. Therefore each path segment of the chromosome point list is interpolated according to a predefined distance. Afterwards the robots tool center point is set to each interpolation point and detected collisions are summed up. Choosing a small interpolation distance means a very accurate and slow collision check and a huge interpolation distance means an imprecise and fast check. The number of collisions for all path segments of a chromosome are counted. When the counted collisions exceed a threshold the fitness of the chromosome is set to epsilon. Given that the number of potential collisions rises with a smaller interpolation distance and declines with a higher interpolation distance the convex hull collision threshold is set according to the length of the interpolation distance between 20 and 100 at which 100 is chosen for a very small interpolation distance.

Termination: The whole process as shown in Fig. 1 is repeated until a termination condition is reached. Possible conditions are: a solution which reaches the minimal fitness or a fixed number of iterations is reached.

4. Experimental Results

The size of the scene is correlated to the maximal range of the KUKA KR210 R3100 robot of maximal 3095mm in all directions. Each planner was tested ten times. The maximal allowed time for one planner was limited to 20 minutes. The resolution of the planning for the OMPL planners was set to 0.01% of the scene size. The evaluation of all planners was done by the following parameters: solution found (yes/ no), planning time (smaller=better) and path length (smaller=better). The planning was done with constraints which means that the orientation of the Tool Center Point (TCP) was fixed during the whole movement. The paths have not been smoothened after planning. All calculations have been performed on an Intel Xeon E5620 with 2,4 GHz and 12GB RAM and a NVIDIA Quattro 4000 graphic card. Table 1 and Fig. 4 show the planning results.

For sampling-based planner the study showed that RRT*, PRM, RPM*, TRRT and RRTconnect have the best performance regarding path length from the various planners available in OMPL. Referring planning time, RRTconnect, RRT and TRRT performed best. The star-algorithms are so called optimizing planners which converge to a best path regarding an optimization criterion, in our case path length. It tries to optimize, until a stopping condition is met. It was found for all planners that in average planning in SE(3) space gives best results compared to Joint Space. Depending on the requirements the right planner can be chosen. Optimized planners could be extended by another stop criterion like no changes of the path length in the last x-iterations which could speed them up dramatically.

Table 1 Overview of planning results (SE(3), with constraints)

Planner	Success rate (%)	Calculation time (*s*) Avg / min /max	Euclidian path length (mm) Avg / min /max
PRM	100	1200 / 1200 / 1200	5956.7 / 5034.31 / 9346.53
RRTconnect	100	9.36 / 3.7 / 29.52	6195.55 / 5182 / 9901.62
RRT	100	15.62 / 2.66 / 46.82	6066.35 / 5043.03 / 11689.81
RRT*	100	1200 / 1200 / 1200	6464.78 / 5182.85 / 12027.31
TRRT	100	12.19 / 2.13 / 43.63	6462.78 / 5182.85 / 12027.31
PRM*	100	1200 / 1200 / 1200	12403.45 / 5020.68 / 41282.02
SPARS	70	1200 / 1200 / 1200	6989.14 / 5199.34 / 15768.85
STRIDE	100	130.14 / 3.07 / 346.26	7548.09 / 5152.4 / 15990.59
Genetic (1 ViaPoint)	100	91.4 / 88 / 97	4587.83 / 4414.75 / 5198.06
Genetic(2 Via Points)	100	118 / 109 / 130	4738 / 4543.69 / 4952.78
Genetic (3 Via Points)	100	134 / 122 / 150	5350 / 5124.64 / 5706.33
Genetic (4 Via Points)	100	142 / 123 / 175	6895.91 / 6175.60 / 7780.91

For the genetic algorithm there are a lot of parameters which can be changed and have direct influence to the result of the algorithm. For the results which can be seen in Table 1 the following settings were used: individual count = 50, number of iterations = 20, single mutation rate = 0.15 (determines the total amount of mutated chromosomes), one-point crossover rate = 0.75 (determines the amount of chromosomes which participate in crossover), random portion selection = 0.2 (defines the amount of random chromosomes in the new population), population count = 1, collision interpolation distance = 100mm, fitness weight ω_1= 0.3 and ω_2= 0.7. The parameters individual count, roulette wheel selection, number of iterations, collision interpolation distance and number of populations have a huge influence on the computing time of the algorithm, because the higher the values more collison checks have to be performed.

During the experiments additionally the following points were identified. Really crucial for the success of the genetic algorithm is the fitness function. Only if that function provides meaningful values which allow comparing different chromosomes, the algorithm will find a collision free solution. The range of mutation should be not to low and not too high. If the value is too small the mutation will not help very much to find new good chromosomes. If the value is to big the algorithm will have very high jumps also after many iterations which makes it very unstable. For our experiments a good value was 10% of the working space which is around 300mm. The more ViaPoints are chosen the longer the final path will get. This is due to the zig zag form of the found path. To add a smoothness criterion of the path to the fitness function could improve this problem. As stop criterion we used the maximal number of iterations reached which sometimes was too long, because a good path was already found after some iterations. Here a more intelligent stop criterion which e.g. considers the overall fitness of the population should be used.

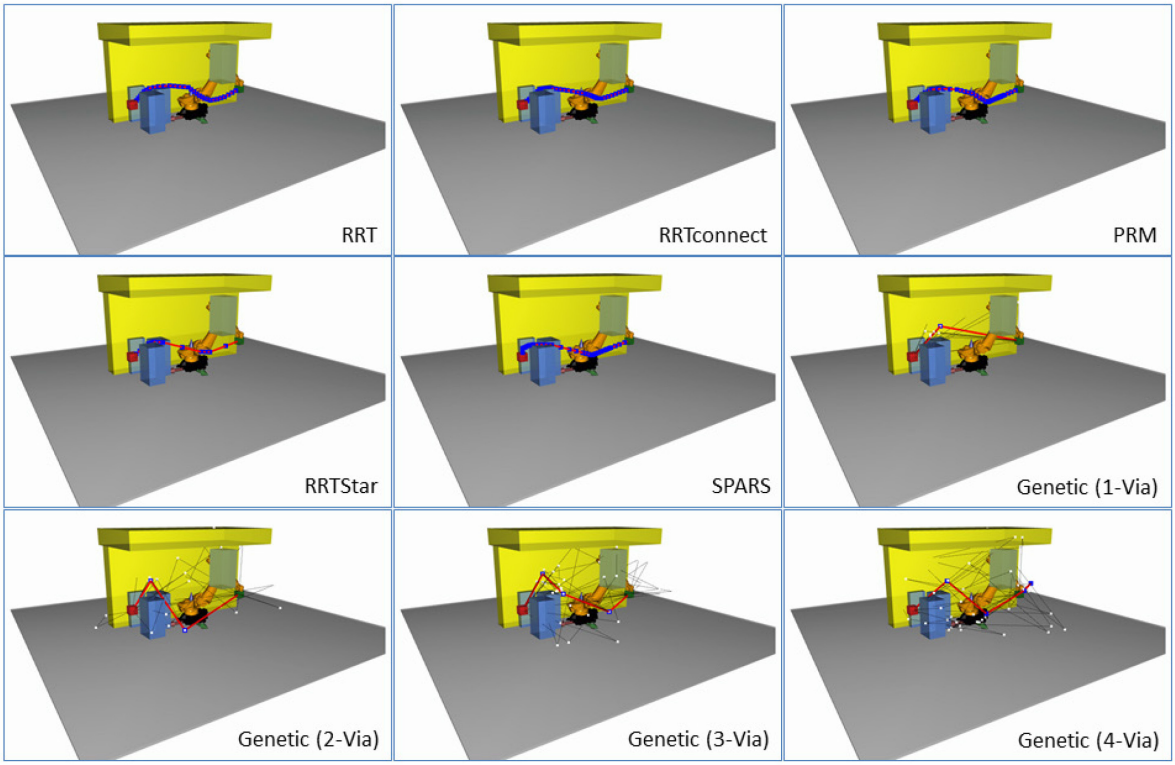

Fig. 4. Setting of the test scene and different planning results. Start point in the front in red and goal point in the back in green. The direct distance between start and goal is 4184mm. The blue and yellow boxes and the grey ground represent obstacles. The red line with blue boxes shows the found path. The screenshots of genetic planners additionally show the residual solution of the population with black lines and white boxes.

5. Conclusion

It could be shown that both sampling-based planners and genetic algorithm can be used for path planning applications of 6-DOF industrial robots. On the one hand RRT and TRRT are much faster than the genetic planner but on the other hand the paths found by genetic planner are shorter although e.g. RRT* had 1200s calculation time. In practice it seems that probabilistic planners are better to use them in real production scenarios, because the paths are much smoother

We already successfully connected the CoCo simulation framework with real robotic hardware to validate the paths where it arised that automatic planning can be really useful in industrial context. To connect the path planning system with the robot system, we used the RoboticAPI [27] which provides an object-oriented programing interface for industrial robots. All operations which require hard real-time such as motion planning are automatically translated into an intermediate language and executed by an external motion controller called Robot Control Core (RCC) [28]. The direct communication with the robotic hardware allows us to dynamically react on changes in the process and to replan which is an advantage compared to state of the art offline programming tools planning solutions.

References

[1] H. Lasi, P. Fettke und H.-G. Kemper, „Industry 4.0," *Business and Information Systems Engineering,* Bd. 4, Nr. 6, 2014.

[2] S.-W. Suh, I.-K. Woo und S.-K. Noh, „Development of an automatic trajectory planning system (ATPS) for spray painting robots," *Proceedings of the IEEE International Conference on Robotics and Automation,* 1999.

[3] H. Chen, T. Fuhlbrigge und X. Li, „Automated Industrial Robot Path Planning for Spray Painting Process: A Review," *4th IEEE Conference on Automation Science and Engineering,* 2008.

[4] Y. Ting, W. Lei und H. Jar, „A Path Planning Algorithm for Industrial Robots," *Computers & Industrial Engineering,* 2002.

[5] S. Klanke, D. Lebedev, R. Haschke, J. Steil und H. Ritter, „Dynamic Path Planning for a 7-DOF Robot Arm," *International Conference on Intelligent Robots and Systems,* 2006.

[6] K. Oh, J. Hwang, E. Kim und H. Lee, „Path Planning of a Robot Maipulator using Retrieval RRT Strategy," *International Journal of Electrical,* 2007.

[7] C. Qin und D. Henrich, „Path Planning for Industrial Robot arms - A Parallel Randomized Approach*," 1996.

[8] S. M. LaValle, Planning Algorithms, Cambridge: Cambridge University Press, 2006.

[9] S. M. LaValle, „Rapidly-Exploring Random Trees: A New Tool for Path Planning," 1998.

[10] L. E. Kavraki, P. Svestka, J.-C. Latombe und M. H. Overmars, „Probabilistic Roadmaps for Path Plan- ning in High-Dimensional Configuration Spaces," *Robotics and Automation,* 1996.

[11] D. Coleman, I. A. Sucan, M. Moll, K. Okada und N. Correll, „Experience-based planning with sparse road- map spanners," *International Conference on Robotics and Automation (ICRA),* 2015.

[12] N. Achour und M. Chaalal, „Mobile Robots Path PLanning using Genetic Algorithms," *IARIA,* 2011.

[13] A. Elshamli, H. A. Abdullah und S. Areibi, „Genetic Algorithm for Dynamic Path Planning," 2004.

[14] P. Sivakumar, K. Varghese und R. N. Babu, „Path Planning of Construction Manipulators using Genetic Algorithms," *Automation and Robotics in Construction,* 1999.

[15] L. E. N. Patrocinio Nunes, V. O. Gamara Rosado und F. J. Grandinetti, „Robotic Manipulator Path Planning by Genetic Algorithms," *International Congress of Mechanical Engineering,* 2007.

[16] F. J. Abu-Dakka, „Trajectory Planning for Industrial Robots Using Genetic Algorithms," Valencia, 2011.

[17] A. C. Nearchou und N. A. Aspragathos, „A Genetic Path Planning Algorithms for Redundant Articulated Robots," *Robotica,* Bd. 15, pp. 213-224, 1997.

[18] Z. S. Abo-Hammour, O. M. Alsmadi und S. I. Bataineh, „Continuous Genetic Algorithms for Collision-Free Cartesian Path Planning of Robot Manipulators," *Internal Journal of Advanced Robotic Systems,* 2011.

[19] L. Larsen, J. Kim und M. Kupke, „Intelligent path panning towards collision-free cooperating robots," SciTePress, Vienna, 2014.

[20] L. Larsen, V.-L. Pham, J. Kim und M. Kupke, *Collision-free path planning of industrial cooperating robots for aircraft fuselage production,* Seattle: IEEE Intl. Conf. on Robot. & Autom., 2015.

[21] A. Angerer, A. Hoffmann, L. Larsen, J. Kim, M. Kupke und W. Reif, „Planning and execution of collision-free multi-robot trajectories in industrial applications," *47th Symposium on Robotics - International Symposium on Robotics (ISR),* 2016.

[22] H. Toolkit, „https://github.com/helix-toolkit," 08 02 2017. [Online].

[23] BEPUphysics, 07 02 2017. [Online]. Available: http://www.bepuphysics.com/.

[24] I. A. Sucan, M. Moll und L. E. Kavraki, „The Open Motion Planning Library," *IEEE Robotics & Automation Magazine,* 2012.

[25] AForge, 08 02 2017. [Online]. Available: http://www.aforgenet.com/aforge/framework/.

[26] J. H. Holland, Adaptation in Natural and Artificial Systems, 1992.

[27] A. Angerer, A. Hoffmann, A. Schierl, M. Vistein und W. Reif, „Robotics API: Object-Oriented Software Development for Industrial Robots," *Journal of Software Engineering for Robotics,* 2013.

[28] M. Vistein, A. Angerer, A. Hoffmann, A. Schierl und W. Reif, „Flexible and continuous execution of realtime critical robotic tasks," *International Journal of Mechatronics and Automation,* 2014.