Trajectory Optimization Methods for Robotic Cells Considering Energy Efficiency and Collisions

Elias Knöchelmann[1], Dwayne Steinke[1], Joel Greenyer[2], Svenja Spindeldreier[1], and Tobias Ortmaier[1]

[1] Institute of Mechatronic Systems, Leibniz Universität Hannover, Hannover, Germany
elias.knoechelmann@imes.uni-hannover.de,
[2] Software Engineering Group, Leibniz Universität Hannover, Hannover, Germany

Abstract. In production robots are moved at maximum speed whenever possible in order to achieve the shortest overall cycle time. This can lead to individual waiting times, especially in interlinked production processes. These waiting times offer opportunities for optimization. Due to high energy prices and political efforts, energy efficiency has become the focus of trajectory optimization in recent years. Robot cells with a common intermediate circuit offer the possibility of energy exchange across individual axes or robots. By adapting the robot trajectories, the total power consumption of a robotic cell on the grid side can be significantly reduced. This paper focuses on trajectory optimization, whereby a detailed collision detection of individual robots is included within the analysis. It is shown that with collision detection energy optimization for cramped robot cells becomes possible and the losses in efficiency compared to the optimization without it are minute.

1 Introduction

Reducing CO_2 emissions through higher energy efficiency is one of the main goals of our time. Operating costs of factories can also be reduced by lowering the energy consumption of its components. As automation advances, the number of robots in production is constantly increasing [22]. This has also been recognised by the EU. An energy roadmap [7] has been issued in 2014 with the goal of increasing energy efficiency by 27 % until 2030. The aim of this work is to reduce the energy consumption of industrial robot cells by adapting robot movements. The method is evaluated using the robot cell shown in Fig. 1.

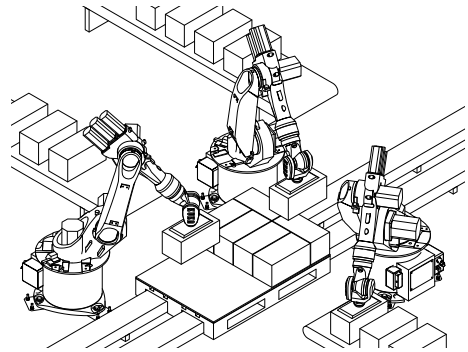


Fig. 1: Investigated robot cell packing a pallet, with several potential collisions.

The reduction of the energy consumption of robots can either be hardware-based or software-based [5]. Examples for hardware changes are e.g. extension of the

T. Schüppstuhl et al. (Hrsg.), *Annals of Scientific Society for Assembly, Handling and Industrial Robotics*, https://doi.org/10.1007/978-3-662-61755-7_21

DC link to several robots [27] or the installation of energy storage devices [13]. With these two methods recuperated braking energy from the motors is stored or used for other robots. This means that recuperated energy for overvoltage protection does not have to be converted into heat [27]. However, these methods have the disadvantage that additional adaptation costs are incurred per robot and that systems for installation have longer downtimes.

In the case of software solutions, costs are only incurred during development, and the transfer to several systems is negligible. Software solutions are a geometrically or time sequence change to a robot trajectory [5]. These solutions are a part of trajectory optimizations and usually refers to one robot. In [30] a robot trajectory was decomposed into viapoints and the position of these were optimized. In [15] the energy absorption of a trajectory was optimized based on maximizing the energy exchange in the DC link. Another possibility is a different time sequence of robot trajectories for a given task. In most cases several robots are involved. One method in this field is, instead of waiting after fast movements, the robots movement may be slowed down. In [9] different robot trajectories were investigated and it was found that slowing down trajectories almost always saves energy. However, movements can also be shifted in time so that braking movements and acceleration, from different robots/axis, directly overlap over each other. Of course this only has advantages if robots can exchange energy.

In industry this is not possible at the moment, but by implementing a DC factory, an energy exchange at factory level becomes possible [27]. In this paper it is assumed that the robots have such a connection. The method of sliding over each other can also be combined with the slowing down of fast movements. In [14] such an optimization for a multi-axis system was presented. Losses from overvoltage protection are reduced for one cyclic trajectory per axis. A method for scheduled energy optimal trajectories was presented in [29] and great energy savings were shown for a single robot. When considering geometric path changes or when changing the time sequence of several robot movements, it is important that no collisions occur. [28] took collisions into account, but only for a restricted area and [24] used simple bounding boxes. In [2] an iterative collision detection for a robot was presented, which guarantees collision freedom for the next pose. However, to the best knowledge of the authors, for larger linked movements of several adjacent robots with a detailed collisions detection, as shown in Fig. 1, no uniform method for trajectory optimization was published.

An automated methodology to exploit braking energy recuperation though scheduling and slowing down trajectories was proposed in [12]. The proposed methodology accomplishes this in two levels. In the first level, the sequential control optimization (SCO), the overall discrete control of the system is optimized to maximize the times in which the acceleration and deceleration phases of individual robots beneficially overlap. Usually, the order of robot movements can be arranged in multiple ways, and some control strategies offer more potential for braking energy recuperation than others. Then, in the second level, the detailed trajectory optimization (DTO), for each set of braking and acceleration phases that an optimized control strategy brought to overlap, the trajectories of the involved axes must be aligned in detail. For example, the braking phase of one axis can be shifted, stretched, or compressed to match the energy demand for ac-

celerating another axis. In this paper, we focus on the DTO, a method for the overlying SCO was already published in [12].

The SCO calculates the acceleration and deceleration phase of robots that should overlap to maximize the energy efficiency gain. These time points and related robot paths are given to the DTO. Here the robot trajectories are improved in an model-based offline optimization. The two internal robot control variables (*start delay* and *compression factor*) are used as optimization parameters. This guarantees an intuitive transferability to industrial plants. The entire process is model-based and offline, therefore only the newly calculated parameters must be changed in the controller. In addition, a collision detection is presented, which can be used as a nonlinear constraint within the optimization. The collision detection method was only applied to DTO, because collision in robot cells, as the one sketched in Fig. 1, are a matter of trajectory timing. This collision detection routine is a two-layered process [4] to guarantee a fast and accurate detection. The method will be tested in the simulated robot cell shown in Fig. 1.

In Sect. 2 the energy model used to simulate the overall energy consumption of a production process is described. Sect. 3 builds on the model of Sect. 2 to optimize the overall energy consumption. The optimization approach and the applied constrains are outlined in Sect. 3. To prevent collisions, which can occur during optimization from Sect. 3, an collision detection is described in Sect. 4. Results are evaluated in Sect. 5 and the paper concludes in Sect. 6.

2 Modeling of Industrial Robots

In this section the robot model used in optimization will be presented. Since the model is often invoked within the optimization, one of the main requirements, besides the accuracy, is a short computing time. The complete model was already presented in [12] but it will be described roughly for the sake of completeness. The general structure of the energy flow is shown in Fig. 2.

The power flows from the main grid P_{grid} to the constant consumers $P_{\mathrm{l,grid}}$. These are normal losses caused by fans and controllers. Besides that, the main part of P_{grid} flows thought the rectifier $\mathrm{T}_{\mathrm{DC},1}$ into the DC intermediate circuit. The DC intermediate circuit mainly supplies the inverters $\mathrm{T}_{\mathrm{AC}_{1-n}}$, which in turn supply the motors M_{1-n}, whereby n represents the numbers of inverters/motors of each robot. Considering the gear, the motors generate a joint torque τ and a joint speed $\dot{\varphi}$. This drives the segments of the robot to follow a given path

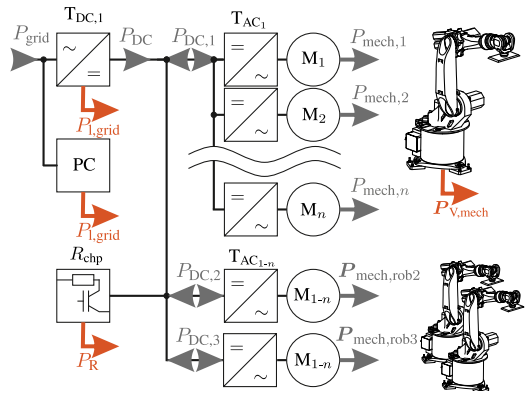


Fig. 2: Detailed energy flow in the investigated robots.

$\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t)$ with an additional weight $\boldsymbol{f}_{\mathrm{ext}}$. When the robot slows down, the motors turn into generators and recuperate energy. This energy is converted to DC and can supply other robots via the DC link. If too much power flows into the DC bus, the DC bus voltage increases. As overvoltage protection, a braking resistor R_{chp} converts the excess energy into heat. The complete energy model of the robot can be written as:

$$\boldsymbol{E} = \int_0^{t_{\mathrm{end}}} P_{\mathrm{grid}}(t)\mathrm{d}t = \boldsymbol{f}_{\mathrm{m}}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t), \boldsymbol{f}_{\mathrm{ext}}). \tag{1}$$

For a known robot, a robot path $\boldsymbol{q}$,$\dot{\boldsymbol{q}}$,$\ddot{\boldsymbol{q}}$ and an extra weight f_{ext} can be used to calculate the required energy. The model is nonlinear and therefore no gradient based methods can be used for optimization.

The energy consumption model is validated for multiple different KUKA industrial robots [17] and has a remaining model deviation of approx. 5 % over the whole spectrum of operating temperatures [8].

3 Optimization Strategy

Common industry robots have six joints, but the axes movements are all synchronized to the slowest joint. The output of a production, therefore, depends on the slowest axis movement. An unnecessarily strong acceleration of the other axes has no advantage with regard to the quantity produced. The synchronization can, therefore, lead to energy savings, but there is no recuperation from decelerating, because all the joints are slowing down at the same time. The main goal of the optimization is to make all robot movements equally fast/slow and prevent unnecessarily strong acceleration. The optimization problem thus consists of optimization variables for compression of the trajectory $\delta_{\mathrm{ov},j}$ and a start delay $\tau_{\mathrm{s},j}$ for a robot j to synchronize accelerating and decelerating from different robots. The effect of these variables on a single joint robot axis can be seen in Fig. 3.

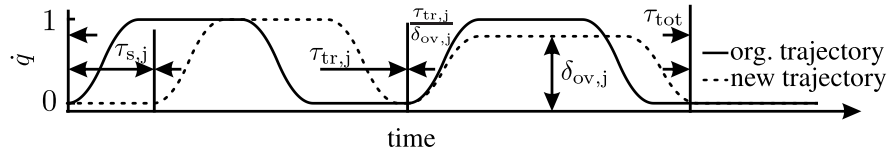


Fig. 3: Visualisation of the used optimization variables.

The start delay $\tau_{\mathrm{s},j}$ for a robot j shifts the trajectory n - time steps into the future. The lower limit is zero and the upper limit is the travel time for that trajectory of the slowest robot $\tau_{\mathrm{tr,krit}}$ subtracted by the travel time $\tau_{\mathrm{tr},j}$ of the robot j. By optimizing multiple robots with only one trajectory each, the travel time of the slowest robot

$\tau_{\mathrm{tr,krit}}$ equals the total travel time τ_{tot}, otherwise it is the sum of the the slowest trajectory $\tau_{\mathrm{tr}_l,\mathrm{krit}}$ of the respective l multi-robot movement. The compression variable $\delta_{\mathrm{ov},j}$ reduces the maximal velocity $\dot{q}_j$ and acceleration $\ddot{q}_j$ of the trajectory:

$$\dot{q}_{\mathrm{new,j}} = \delta_{\mathrm{ov},j} \cdot \dot{q}_{\mathrm{org},j} \quad \wedge \quad \ddot{q}_{\mathrm{new},j} = \delta_{\mathrm{ov},j} \cdot \ddot{q}_{\mathrm{org,j}}. \tag{2}$$

The travel time of the robot j changes to $\frac{\tau_{\mathrm{tr},j}}{\delta_{\mathrm{ov},j}}$. The lower limit is 0 and the upper limit is 1. The optimization problem finally can be described by:

$$\min_{\tau_{\mathrm{s}},\delta_{\mathrm{ov}}\in\mathbb{R}} E_{\mathrm{grid}}(\boldsymbol{\tau}_{\mathrm{s}}, \boldsymbol{\delta}_{\mathrm{ov}}) \;\; \text{s.t. } 0 \leq \delta_{\mathrm{ov},j} \leq 1 \tag{3}$$

$$0 \leq \tau_{\mathrm{s},j} \leq \tau_{\mathrm{tr,krit}} - \tau_{\mathrm{tr},j} \tag{4}$$

$$t(\boldsymbol{\tau}_{\mathrm{s}}, \boldsymbol{\delta}_{\mathrm{ov}}) \leq \tau_{\mathrm{tot}} \tag{5}$$

$$c(\boldsymbol{\tau}_{\mathrm{s}}, \boldsymbol{\delta}_{\mathrm{ov}}) < 1, \tag{6}$$

where two new nonlinear constraints $t(\cdot)$ and $c(\cdot)$ are introduced. The first constraint makes sure that the total travel is at least not slower than before the optimization. $c(\cdot)$ is a nonlinear function that returns one if a collision is detected. The collision detection will be described in the following section and is skipped here. With this the optimization problem for multiple robots and multiple trajectories is fully defined.

The non-linear optimization problem and especially the nonlinear constraints result in gaps in the solution area. For that reason a particle swarm optimizer (PSO [26]) was chosen. The PSO does not guarantee that the global minimum will be found, but returns an acceptable solution in reasonable time. A more detailed view on constraints handling and the gaps in the solution area will follow in the next section.

For runtime reasons it makes sense to split the optimization into several sub-problems and solve them individually. Sub-problems are single trajectories of each of the participating robots, so $\tau_{\mathrm{tr,krit}}$ equals τ_{tot}. The described optimization is only the secondary level, so the SCO matches the trajectories with the most energy saving potential. Since it is already known from the SCO which trajectories of the different robots have to be superimposed, there is no need to optimize n_{tr} trajectories of m robots. It is sufficient if only one trajectory per robot, selected by SCO, is optimized in one sub problem.

The drawback of this method is that the success of the underlying optimization depends on the SCO. By dividing the optimization problem, a satisfactory solution in a satisfactory time is obtained, but if the SCO does not find the best matches, the solution is not the best possible. However, without dividing an exemplary test case with 3 robots and 10 trajectories each can no longer be solved on a PC in a satisfactory time.

The fact that only one trajectory per robot is considered simplifies the constraints of optimization. The lower limit for the compression factor $\delta_{\mathrm{ov},j}$ of robot j is now limited by the travel time of the slowest robot in the considered subproblem. Before there were multiple trajectories of the same robot in a row, so there was no defined end of a single trajectory, because trajectories could overlap. The start and end time of the critical robots are known by the SCO and therefore the trajectories can be separated. The upper limit for the start delay now also depends on the compression factor. In contrast to the standard problem, again without overlap, the formulation of the constraints is easier. This simplifies the constraints to:

$$\frac{\tau_{\mathrm{tr},j}}{\tau_{\mathrm{tr,krit}}} \leq \delta_{\mathrm{ov},j} \leq 1, \tag{7}$$

$$0 \leq \tau_{\mathrm{s},j} \leq \tau_{\mathrm{tr,krit}} - (\tau_{\mathrm{tr},i} \cdot \delta_{\mathrm{ov},j}), \tag{8}$$

$$c(\boldsymbol{\tau}_{\mathrm{s}}, \boldsymbol{\delta}_{\mathrm{ov}}) < 1. \tag{9}$$

So there is no need for a nonlinear time constraint, because only through linear constraints it can be ensured that the total travel time τ_{tot} remains the same. The collision detection $c(\cdot)$ stays the same and will be described in the next section.

4 Collision Detection

Using the model from Sec. 2 the optimization strategy from Sec. 3 works well if the robots do not share a common workspace. However, if two or more robots share a workspace there is the possibility that optimized trajectories may lead to collisions even if non-optimized trajectories do not collide. For example, there are two robots (A and B) standing across each other. Robot A is moving vertically and robot B is moving horizontally, so there is a possibility that these two are colliding in the middle. Assuming that robot B is the critical/ slowest robot, then there are two optimization parameters: The compression factor (override) and trajectory start delay of robot A. The collision detection leads now to a divided solution space as shown in Fig. 4. This (red) barrier depends on the compression factor and the start delay and marks a collision. Because of this every generated trajectory of the optimization needs to be checked for collision to only consider valid solutions.

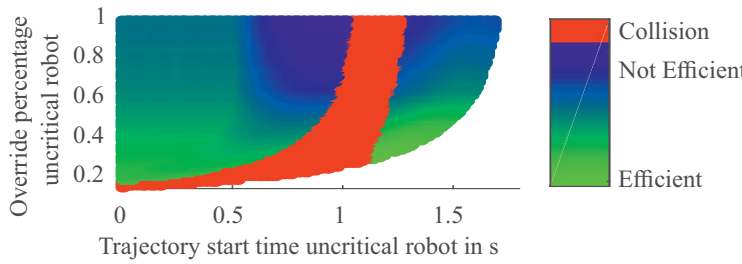


Fig. 4: Solution space divided by collisions of a two robot cell optimization problem

The collision model uses CAD data of the robots as well as other objects in the cell and imports it as a polygon mesh model. For every sample time step the positions of the objects are recalculated from the robot joint angles of the current trajectory. If no collision is found for all sample time steps, the current parameter vector is valid and can be used in optimization.

The repeating collision detection routine is, as shown in [4, 18, 6], divided into a broad phase that approximates the objects with bounding boxes and a narrow phase

which determines exactly if two objects collide. The broad phase calculates candidates that may collide and need further investigation in the narrow phase. In the past lots of algorithms and solutions for use in broad phase [20, 6, 11, 19, 16] and narrow phase [21, 23, 10, 3, 1, 25] with advantages and disadvantages depending on the number of objects, complexity of the meshes or compactness of the objects have been developed.

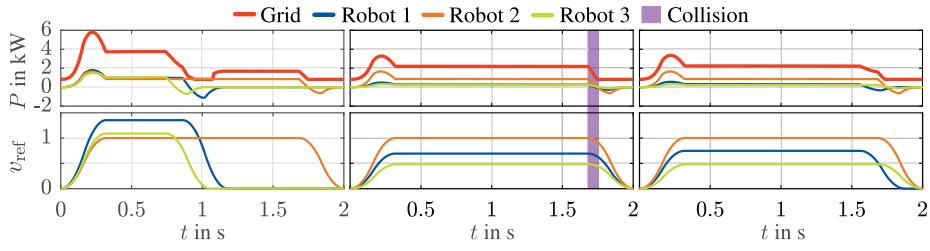
Therefore for the collision detection routine within the trajectory optimization three different methods for the broad phase (axis-aligned bounding box pair test (AABB) [6], the Sweep and Prune algorithm (S&P) [6] and an oriented bounding box pair test (OBB)[11]) and two for the narrow phase (V-Clip [23] and the Gilbert–Johnson–Keerthi separating-axis algorithm (GJK) [1]) were implemented and evaluated. The selection was made on the basis of speed, implementation effort and availability of literature.

To compare the selected algorithms three runtime tests were performed. This and all other test were conducted in MATLAB on an i5-6400 desktop computer. In order to guarantee comparability with the robot cell application, three tests with one trajectory each time were performed for the set-up of the test cell (Fig. 1) . Table 1 shows the average calculation times in ms for the collision check for each case. Each test was repeated 200 times and the average value was obtained. Due to the small number of tests, the deviation is very high (up to 10%), but a generalized statements can still be derived: Sweep-and-Prune shows its advantages in this application-related test, because the system is made up of many objects that move only partially, which is the strong point of this algorithm. Further detailed tests have shown that GJK is faster than V-Clip for the narrow phase, therefore GJK and S&P were chosen as the best solutions.

To speed up the optimization it is also possible to only examine those objects that can collide. With the proposed optimization strategy the robots change their trajectory but not their path. Two objects can only collide if the spatial paths, for a given optimization parameter set, cross. The collision detection approach is therefore expanded by an extra precomputation phase. There, in advance the objects that have the potential to collide will be determined. The precomputation phase is not exact but an approximation, because the reconstruction of a polygonal mesh from multiple meshes is too complex for a preexamination. This generates from any parameter set a point cloud for each object consisting of all vertices from all collision sample time steps. These point clouds represent the paths with all time dependencies eliminated. If two points from different

Table 1: Speed comparison for 3 different test cases in ms. Two times 100 runs are averaged in each case. Case 1. is test cell (Fig. 1) with 16 objects, one trajectories per robot and no collision, 2. are 24 objects with collision and 3. 24 objects and no collision.

collision detection routine		test case		
broad phase	narrow phase	1.	2.	3.
S&P	V-Clip	1629	242	963
	GJK	1582	237	1359
OBB	V-Clip	3469	520	2082
	GJK	3405	499	1975
AABB	V-Clip	3121	486	1858
	GJK	3125	513	1923



(a) No optimization (b) Optimization, no collision detection (c) Optimization with collision detection

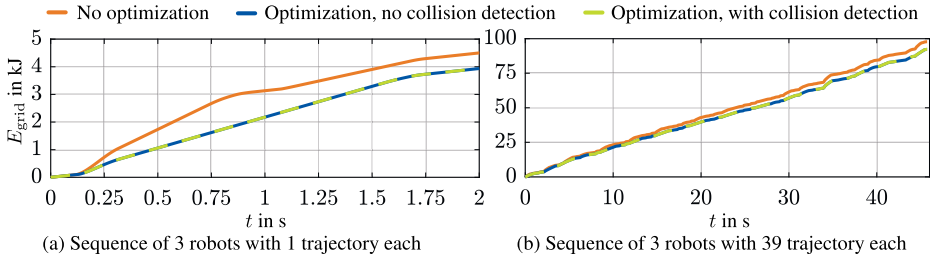
Fig. 5: Power demand and relative robot speeds for the same trajectory

point clouds are within a detection radius r they are marked as possibly colliding and relevant for the collision detection routine. All other objects in this robot cell e. g. robot bases, fences, conveyor or similar are irrelevant, as they don't change their position and can never collide with the robots.

To integrate the collision detection into the PSO three types for restraining the non-linear constraints are available: absorbing walls, reflecting walls and invisible walls [26]. Fig. 4 shows that a divided solution space can result from the layout of the cell. Since the particles of the chosen PSO algorithm must be able to diverge through these boundaries, only the invisible walls [26] are suitable for constraints handling.

5 Results

Fig. 5 shows the simulation results of an example system consisting of three robots as displayed in Fig. 1. In Fig 5 the overall grid power demand and the DC bus power flow between each individual robot and the DC bus in comparison to the relative robot speed V_{ref} is shown. The relative robot speed is defined as the absolute sum of all robot joint velocities standardised to the maximum value of the slowest axis of the slowest robot. The travel time of robot 2 is the longest. Therefore, robot 2 is the critical robot of this example system. Its trajectory parameters are not optimized and the movement does not change between the portrayed optimization types. Fig. 5(a) shows the non-optimized



(a) Sequence of 3 robots with 1 trajectory each (b) Sequence of 3 robots with 39 trajectory each

Fig. 6: Comparison of the energy demand with different optimization strategies

case. Each robot moves from its start point to its endpoint as fast as possible. This results in wait times for robots 1 and 3 and high grid power demands while acceleration.

Fig. 5(b) shows the case of optimization without considering collisions. The optimization leads to slower movements of robots 1 and 3 in such a way that all robot trajectories are synchronized. However these trajectories lead between 1.68 s and 1.78 s to a collision between robot 1 and 2 and are therefore not feasible. The time of the collision is marked in red. Using the nonlinear collision detection constraint as shown in Fig. 5(c) this problem is solved because robot 1 moves faster and avoids the collision.

Fig. 6(a) shows a comparison between the energy demands of the different optimization strategies. Without any kind of optimization the example system consumes 4563 J. An optimization without collision detection leads to an energy demand of 3983 J which is a saving of approximate 12.7 %. Changes to the energy demand of the example system through collision detection are minimal and lead to 3984 J. The optimisation results in Fig. 5(b) and (c) indicate that it is not as efficient to match acceleration and deceleration as it is to just compress the trajectories, because there the trajectories were simply stretched. This has two possible causes: First, the DC link capacity consists of three times the amount of a single robot. This means that more energy can be stored for a short time. Since this capacity is not used as an energy storage but mainly as a smoothing capacity, it will probably be smaller for industrial applications, because smaller means less investment costs. The second reason is, that by slowing down the trajectory the acceleration and deceleration was also reduced and thus also less braking energy is recuperated. These observations can be confirmed for a larger system consisting of 3 robots with 39 trajectories each, as shown in Fig. 6(b). By the sub problem optimization the overall energy saving is approximately 9 % compared to trajectories in which each robots moves as fast as possible and needs to wait for the critical robot. The overall results can be seen in Table 2.

Table 2: Simulation results for different types of trajectory optimisation for the test cell in Fig. 1 consisting of 3 robots with 39 trajectories each.

	required power of the DC link		process time	collision
case	in J	in %	in s	
no optimization	60018	100	45.585	no
optimization without collision detection	54461	90.74	45.585	yes
optimization with collision detection	54538	90.87	45.585	no

6 Conclusion

In this paper the second part of a novel engineering methodology for the development of energy-efficient production system controllers is presented. The functional principle and results of the first optimization level (SCO) have already been published in [12]. This publication presents the underlying optimization strategy (DTO) together with a collisions detection. The complete optimization problem was set up, with two optimization parameters i. e. start delay and compression factor. Based on the already presented SCO the optimization problem could be divided into multiple independent sub problems. Simplification for the optimization constraints were also be derived from this division, so that no non-linear constraints are required except for collision detection. Only by these simplifications it is possible to solve systems with more than 10 consecutive trajectories.

The disadvantage of this division is, however, that the success of this procedure depends heavily on the SCO. If the SCO does not find a good solution, the result of the DTO is also not optimal. For small cells with few trajectories (< 10) it may make sense to solve the general optimization problem.

The implemented two-step collision detection was presented, as well as a short runtime comparison of the different methods. In addition, a method was presented which will remove robots from the collision detection if they can never collide with other robots. All methods presented were tested on a virtual test cell (Fig. 1). The DTO reduced the energy consumption when loading the pallet by 9 %. Especially due to the large DC-link capacity, the optimization did not match acceleration and deceleration phases, but slowed down the trajectories as much as possible. The collision detection has only marginally reduced the energy saving. With this method it is possible to optimize DC-powered robots even though the space in the cell is limited.

Acknowledgment

The presented research is funded by German Research Foundation (Deutsche Forschungsgemeinschaft/DFG).

References

1. G. v. d. Bergen. A fast and robust gjk implementation for collision detection of convex objects. *Journal of graphics tools*, 4(2):7–25, 1999.
2. S. Björkenstam, D. Gleeson, R. Bohlin, J. S. Carlson, and B. Lennartson. Energy efficient and collision free motion of industrial robots using optimal control. In *2013 IEEE international conference on automation science and engineering (CASE)*, pages 510–515. IEEE, 2013.
3. S. Cameron. Enhancing gjk: Computing minimum and penetration distances between convex polyhedra. In *International Conference on Robotics and Automation*, 04 1997.
4. S. A. Cameron. *Modelling solids in motion*. Dissertation, University of Edinburgh, 1984.
5. G. Carabin, E. Wehrle, and R. Vidoni. A review on energy-saving optimization methods for robotic and automatic systems. *Robotics*, 6(4):39, 2017.
6. J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi. I-collide: An interactive and exact collision detection system for large-scale environments. In *Symposium on Interactive 3D graphics*, pages 189–218. ACM, 1995.

7. E. Commission. 2030 energy strategy.
8. K. Eggers, E. Knöchelmann, S. Tappe, and T. Ortmaier. Modeling and experimental validation of the influence of robot temperature on its energy consumption. In *ICIT*, pages 239–243. IEEE, 2018.
9. K. Eggers, Z. Ziaukas, J. Kotlarski, and T. Ortmaier. On the relationship of travel time and energy efficiency of industrial robots. In *IcoIESE*. Atlantis Press, 2018.
10. E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.
11. S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Annual conference on Computer graphics and interactive techniques*, volume 23, pages 171–180. ACM, 1996.
12. D. Gritzner, E. Knöchelmann, J. Greenyer, K. Eggers, S. Tappe, and T. Ortmaier. Specifying and synthesizing energy-efficient production system controllers that exploit braking energy recuperation. In *CASE*, pages 408–413. IEEE, 2018.
13. C. Hansen, K. Eggers, J. Kotlarski, and T. Ortmaier. Comparative evaluation of energy storage application in multi-axis servo systems. *Proceedings of The 14th IFToMM*, 2015.
14. C. Hansen, K. Eggers, J. Kotlarski, and T. Ortmaier. Concurrent energy efficiency optimization of multi-axis positioning tasks. In *IEEE ICIEA*, pages 518–525, 2015.
15. C. Hansen, J. Kotlarski, and T. Ortmaier. Experimental validation of advanced minimum energy robot trajectory optimization. In *IEEE ICAR*, pages 1–8, 2013.
16. J. T. Klosowski, M. Held, J. S. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.
17. E. Knöchelmann, J. Kotlarski, T. Böhm, S. Tappe, and T. Ortmaier. Potential of energy storage systems for industrial robots. In *Tagungsband des 4. Kongresses Montage Handhabung Industrieroboter*, pages 168–177. Springer, 2019.
18. S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe. Collision detection: A survey. In *International Conference on Systems, Man and Cybernetics*, pages 4046–4051. IEEE, 2007.
19. E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technischer Bericht TR99-018, Department of Computer Science, University of North Carolina, 1999.
20. M. C. Lin. *Efficient collision detection for animation and robotics*. Dissertation, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1993.
21. M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *International Conference on Robotics and Automation*, pages 1008–1014. IEEE, 1991.
22. G. Litzenberger. Welcome to ifr eb meeting in carate brianza. 2016.
23. B. Mirtich. V-clip: Fast and robust polyhedral collision detection. *ACM Transactions On Graphics*, 17(3):177–208, Juli 1998.
24. A. Muller. Energy optimal control of serial manipulators avoiding collisions. In *Proceedings of the IEEE Conference on Mechatronics*, pages 299–304. IEEE, 2004.
25. C. J. Ong and E. G. Gilbert. Fast versions of the gilbert-johnson-keerthi distance algorithm: Additional results and comparisons. *IEEE transactions on robotics and automation*, 17(4):531–539, August 2001.
26. J. Robinson and Y. Rahmat-Samii. Particle swarm optimization in electromagnetics. *IEEE transactions on antennas and propagation*, 52(2):397–407, Februar 2004.
27. B. Sattler and S. Wiesner. Energiewende meets industrie 4.0, 03.2019.
28. O. Wigström. *Energy efficient multi-robot coordination.* PhD thesis, Chalmers University of Technology, Gothenburg, Sweden, 2016.
29. O. Wigström, B. Lennartson, A. Vergnano, and C. Breitholtz. High-level scheduling of energy optimal trajectories. *IEEE T-ASE*, 10(1):57–64, 2013.
30. Z. Ziaukas, K. Eggers, J. Kotlarski, and T. Ortmaier. Optimizing ptp motions of industrial robots through addition of via-points. In *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics*, pages 527–538. SCITEPRESS, 2017.