Journal of Intelligent & Robotic Systems (2021) 102:25
https://doi.org/10.1007/s10846-020-01309-7

REGULAR PAPER

Robot Speed Adaption in Multiple Trajectory Planning and Integration in a Simulation Tool for Human-Robot Interaction

Paul Glogowski[1] · Alexander Böhmer[1] · Alfred Hypki[1] · Bernd Kuhlenkötter[1]

Received: 1 May 2020 / Accepted: 30 December 2020 / Published online: 30 April 2021

Abstract
Speed and separation monitoring (SSM) is one of the four permissible collaborative operations in human-robot interaction (HRI). At all times, it must be ensured that the speed-dependent separation distance is maintained. To guarantee this, the robot speed or the robot path can be adapted. In this paper, the robot speed adaption for multiple trajectories is implemented in an HRI simulation tool and tested in an application example. Thereby, numerous complex process situations, such as a temporary robot stop or obstacles in the collaborative workspace, can be simulated. The simulation tool enables a comprehensive simulation, analysis and optimisation of human and robot motions within the HRI, already in the planning phase.

Keywords Human-robot interaction · Speed and separation monitoring · Robot speed adaption · Multiple trajectory planning

1 Introduction

Due to the elimination of the separating safeguards, which was enabled by the strong development in the field of industrial robotics and sensor technology, humans and robots share a common workstation within a collaborative workspace. The resulting human-robot interaction (HRI) enables an enormous increase in productivity. At the same time, however, this leads to a potential risk because collisions between the human and the robot can occur. Depending on the process situation, the robot motions must be limited in a way that a safe operation is guaranteed at all times.

According to ISO/TS 15066 [1], four collaborative operations are permissible in HRI. One of these collaborative operations is *speed and separation monitoring* (SSM). Here, a speed-dependent separation distance between the human and the robot in the collaborative workspace is determined. At any time, the current distance must not fall below the separation distance; otherwise, the robot system must stop immediately. During production processes, a robot stop is often the worst case scenario. Therefore, various methods are used to prevent the robot from stopping, for example, by adapting the robot speed. The robot must then reduce its speed that much that the separation distance can be shortened sufficiently and can be maintained. Alternatively, it is feasible to plan a modified robot path in order to ensure the necessary separation distance. At the same time, the robot is supposed to work as fast as possible to provide short cycle times and thus a high productivity. It is precisely this conflict of objectives that this paper addresses, examining the opportunities and risks of these adaptive strategies.

Due to the complexity of the collaborative production system and the necessary safety requirements, a simulation tool is of enormous importance. It enables the planning, visualisation and simulation of a collaborative system well before it goes into initial operation. This can greatly reduce costs and avoid hazards to humans that may occur during necessary (physical) test runs. Therefore, the development of a simulation tool that considers the human and the robot simulation in combination is an essential aspect for the successful implementation of a collaborative production system. Preliminary work [2–4] already developed an HRI simulation tool. This tool provides a manufacturer-independent simulation framework and enables the simulation of typical HRI scenarios with different robot systems and human models. Up to now,

✉ Paul Glogowski
glogowski@lps.ruhr-uni-bochum.de

[1] Faculty of Mechanical Engineering, Chair of Production Systems, Ruhr University Bochum, Universitätsstr. 150, 44801 Bochum, Germany

adaptive motion planning was not part of the simulation tool.

The aim of this paper is to extend the existing simulation tool to provide a reliable tool for planning and simulation of HRI scenarios with adaptive motion planning. Thus, different HRI scenarios can be modelled, analysed and simulated with regard to SSM. The necessary separation distances for various robot and human models can be taken into account and the best possible safe robot motion can be calculated. A particular focus of this paper is the necessary discretisation of the motion trajectories of the human and the robot. The necessary adaption of the robot speed creates a mismatch between the positions and times of the trajectories, which must be manipulated and corrected accordingly. As a further point, we model possible situations of speed and path adaption in the context of the adaptive motion planning and implement them into the simulation tool. For example, we consider the situation when the current distance is fallen so far below the separation distance that the speed adaption fails. In this case, a robot stop must be executed.

To avoid the robot stop as often as possible, the planning of an alternative robot path is considered. For the alternative path planning, we generate a large number of arbitrary trajectories. In order to make these trajectories safe, it may be necessary to adapt the robot speeds here as well. To execute the optimal trajectory, we compare different planned robot trajectories for a specific task.

The paper is structured as follows: Section 2 describes the state of research and technology in relation to SSM. In the following, Section 3 discusses the advanced concept for calculating the separation distance between the human and the robot. Section 4 deals with the adaptive safety strategies that enable the maintenance of the separation distance. The focus is on the algorithm for speed adaption. Section 5 explains the resulting adaption of the planned robot trajectory. Finally, Section 6 outlines the implementation of our adaptive strategy for multiple trajectories in the simulation tool. Section 7 evaluates it in an application example.

2 Related Work

For several years, SSM has been an important part of research within the scope of HRI. The research work [5–7] deals with the problem of collision avoidance and calculates danger or safety fields around a source of danger, e.g. the robot. Lacevic et al. [5] describe a hazard evaluation for environmental objects within robot cells. Based on a so-called kinetostatic danger field, the authors determine the complete robot state in terms of position and velocity and derive a danger level in the proximity of the collision object. In further work, Lacevic et al. [6] transfer the kinetostatic danger field into a control strategy. They present a method with which the information from the danger field can be directly mapped into position and speed instructions for the robot. Polverini et al. [7] adopt the concept of the kinetostatic danger field and extend it to moving objects (e.g. a human body) on which the danger is applied. The approach is introduced as a kinetostatic safety field. The kinetostatic safety field depends on both the distance and the relative speed between two objects (e.g. the human and the robot), where the danger field is calculated. Marvel [8] suggests a set of metrics for evaluating and comparing collision avoidance algorithms. The approach considers not only the relative distance between the human and the robot but also their relative speed to each other. Marvel et al. [9] decompose the formula for calculating the necessary separation distance in detail and evaluate it with regard to its applicability. In addition, the authors give guidelines for the implementation and integration of SSM in collaborative robot work cells. Kim et al. [10] investigated the probability of human hand intrusion into the separation distance. It has been shown that this probability can be greatly reduced by modifying various parameters such as the braking time or uncertainty factors of the robot. The probability of intrusion was found to be a suitable index for the productivity of an SSM application. Savur et al. [11] present an experimentation platform for HRI with subcomponents such as a virtual world representation or a human motion capture system. Special attention is paid to a subsystem with the ability to monitor the human physiological feedback during an HRI task. The framework is validated in the real environment in various application examples.

One option to maintain the defined separation distance between the human and the robot is to adapt the robot speed. Byner et al. [12] adapt the robot speed under two aspects in SSM: On the one hand, the authors take into account the current distance between the human and the robot; on the other hand, they also consider the robot's direction. In experimental studies, the authors evaluate a possible increase in productivity through the adapted robot speeds and compare their methods with conventional safety strategies. The experiments show that the continuous speed adaption offers a significant advantage in the productivity of assembly scenarios within SSM. Lasota et al. [13] also perform a continuous robot speed adaption based on a user-defined distance function. Kumar et al. [14] compare the implementation of a SSM scenario in the real environment with a virtual simulation. The speed is also adapted here depending on the HRI scenario. Finally, a comparison between reality and simulation is drawn.

An alternative method to ensure the separation distance at all times is to adapt the robot path. A convex distance envelope is the basis for a method presented by Dröder et al.

[15]. Here, the safety area around the human is covered by a grid of waypoints, which is represented by a cylinder and a hemisphere. In this way, the space that the robot is not allowed to enter moves with the human. The grid points represent waypoints for a subsequent path interpolation to calculate a new (safe) robot path. The preliminary work [16] follows a similar approach. Here, an approach for a human-centred HRI simulation with adaptive motion planning is presented. The calculation of the required separation distances and the associated speed adaption is considered as a central point. The basis for adaptive motion planning is a dynamic distance cylinder, which is located in the origin of the human. The calculated separation distance between the human and the robot determines the radius of the cylinder. Schmidt et al. [17] describe a possible evasive movement of the robot, which depends on the current distance, the robot speed and a defined function for the evasive speed. As soon as an obstacle like the human is detected, a decision is made to change the current robot path based on the calculated distance to the obstacle. The central idea is that from a defined minimum distance on, the collision direction is modified in such a way that there is no speed component in the direction of the human. Liu et al. [18] describe a method that initiates a procedure for collision avoidance, depending on a risk index. The generated trajectories are checked for collisions with the aid of the risk index and adjusted with rounding factors so that the motion sequence is as smooth as possible.

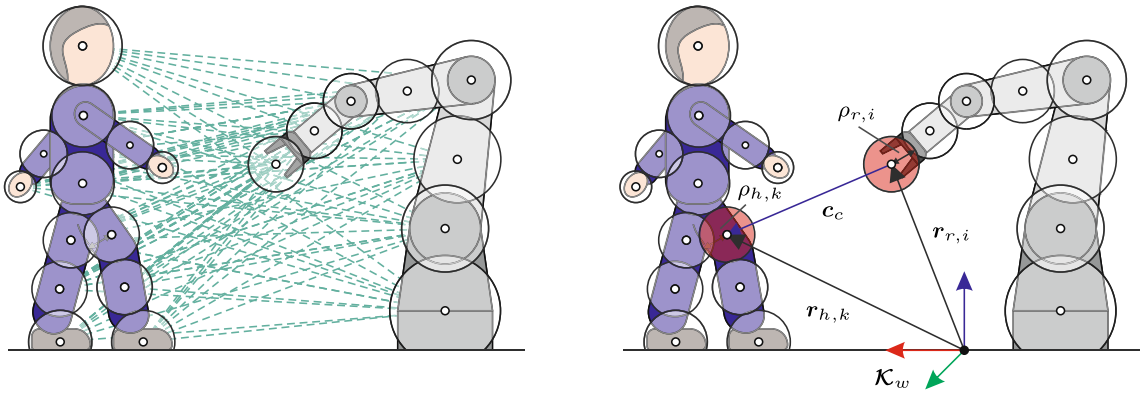
Most previous approaches for the modelling and integration of SSM usually only consider a single representative coordinate to describe the robot motion (e.g. the TCP). Vicentini et al. [19] calculate the trajectory dependent separation distance along the entire kinematic chain of the robot. Zanchettin et al. [20, 21] describe a parameter for the safety evaluation to ensure the separation distance between the robot and the human. At the same time, productivity is increased by the robot speed adaption. This approach also considers the entire kinematic chain of the robot. If the robot falls below a defined separation distance, it reduces its speed. Rosenstrauch et al. [22] present not only a solution for several representative robot coordinates, but also for different human coordinates. For the speed adaption, a scaling factor is introduced, which reduces the speed to a sufficient level. Another important aspect in SSM is the identification of critical points in order to perform the separation distance calculations and, if necessary, an adaption of the robot motion. The preliminary work [23] developed extension concepts for the calculation of the separation distance and examined these in simulation studies. It is shown that the relevant reference points for determining the separation distance on the robot kinematics can vary within a given robot path. Using a calculated time to collision, it is demonstrated that the identification of the most critical point is not only dependent on the shortest distance between the human and the robot. Moreover, the directed relative speed and the current collision direction must be taken into account. The preliminary work [23] also considers the whole kinematic chain of the human and the robot.

As has been shown, there are already a number of promising approaches in SSM, each with its own strengths and weaknesses. However, the existing approaches and safety strategies have so far hardly been integrated into simulation frameworks, or only with insufficient shortcomings. This includes in particular the partly rudimentary modelling of human and robot movements. For example, in many cases only the robot TCP is considered or a constant human speed is assumed. Furthermore, the presented approaches are limited to a certain (prototypical) use case in SSM, e.g. an explicit assembly scenario, a specific robot system, a characteristic human motion, etc. This is precisely where a manufacturer-independent simulation tool can provide a significant advantage.

In this paper, we combine the individual approaches in SSM and integrate them within the simulation tool to form an overall HRI simulation system. The simulation tool allows the universal modelling and simulation of collaborative assembly scenarios. It offers the possibility to compare different robot systems, to examine different processes and tasks, to use different human models, to calculate resulting cycle times of the human and the robot, and to test the selected collaborative production system extensively before a later implementation. Through a simple and intuitive process modelling with the help of assembly task elements (e.g. `MoveRobot`, `PickObject`, `PlaceObject` etc.), a large number of simulation studies can be carried out in a comparatively short time. A process planner can reproducibly simulate many different processes with different robots, tasks and human motions and evaluate their results. Especially in the field of SSM, there are many possibilities for an optimised process and layout design of an assembly workstation; for example by varying the start or target pose of the robot or by changing the human motions. All these factors can have a great impact on the resulting separation distances and the permitted robot speeds. This in turn has a direct influence on the productivity of the entire HRI system. In this context, our simulation tool automatically plans a large number of trajectories and adapts them with regard to the required safety conditions. Finally, the fastest safe trajectory and the most productive solution for a specific task will be executed.

3 Separation Distance

The following section describes an approach for calculating the separation distance between the human and the robot.



(a) Separation distances for all points (b) Separation distance only for the critical points

Fig. 1 Visualisation of the calculation effort for determining the separation distance

The concept described here is based on the preliminary research work in [16] and [23].

3.1 Modelling of the Human and the Robot

The more detailed the human and the robot are modelled, the higher is the calculation effort to determine the current distances between the human and the robot. Due to an increased calculation time, it is not possible to determine the distance between every single point of the human and every single point of the robot. Simplifications have to be made which lead to a feasible calculation effort. In this context, spheres provide a good approximation of the individual objects, since they can only be described geometrically by the position of their centres and by their radii. In this sense, overdimensioning due to large geometric objects cannot be considered critical at first, as this provides additional protection for the system. Despite all this, the enveloping spheres must be as large as necessary but as small as possible, in order to keep the necessary separation distances to a minimum.

To model the robot, we consider n_i moving parts of the robot. Each individual robot link $i \in \{1, \cdots, n_i\}$ is surrounded by an enveloping sphere with the radius $\rho_{r,i}$ (see Fig. 1). The centre of each sphere is located in the centre of gravity of the corresponding link or in the corresponding joint coordinate frame. In order to consider the human motions, the human is also divided into n_k bodies, such as head, hands, shoulders, torso, knees and feet. Here, every single human body $k \in \{1, \cdots, n_k\}$ is described by an enveloping sphere with the radius $\rho_{h,k}$.

3.2 Calculation of the Separation Distance

An essential aspect for the calculation of the separation distance is the description of the directed speeds of the human $v_{h,c}$ and the robot $v_{r,c}$. For this purpose, the velocities of the human $\boldsymbol{v}_{h,k}$ and the robot $\boldsymbol{v}_{r,i}$ are projected onto the collision vector

$$\boldsymbol{c}_c = \boldsymbol{r}_{h,k} - \boldsymbol{r}_{r,i} \tag{1}$$

between the human and the robot.[1] $\boldsymbol{r}_{h,k}$ or $\boldsymbol{r}_{r,i}$ denote the respective position vector to a human body k or to a robot link i with respect to the world coordinate frame $\mathcal{K}_w$. The following applies to the directed speeds:

$$v_{r,c} = \boldsymbol{v}_{r,i} \frac{\boldsymbol{c}_c}{|\boldsymbol{c}_c|} \qquad v_{h,c} = -\boldsymbol{v}_{h,k} \frac{\boldsymbol{c}_c}{|\boldsymbol{c}_c|} \tag{2}$$

The directed speeds $v_{r,c}$ and $v_{h,c}$ are defined in such a way that they move towards each other in the positive case. The current distance between the considered human body k and the robot link i is calculated as follows:

$$C_c = |\boldsymbol{c}_c| - \rho_{r,i} - \rho_{h,k} \tag{3}$$

In many applications, the separation distance S_c between a robot link i and a human body k is calculated in a linearised form:

$$S_c = v_{h,c}\,(T_r + T_{s,i}) + v_{r,c}\,T_r + B_c + S_m \tag{4}$$

T_r is the response time of the robot system and safety technology, $T_{s,i}$ is the braking time of the robot link i. The Cartesian (directed) braking distance of the robot link i is described by B_c. The term S_m defines the minimum distance which results from the depth of penetration and the measurement uncertainties of the used sensors. For the calculation of the braking distances B_c and braking times $T_{s,i}$, we use braking data for the individual robot axes specified by the manufacturer. The manufacturer's values refer to stop 1 (cf. [24]). A detailed calculation of the directed braking distance is described in [23].

[1] All variables that carry the index c refer to the collision vector $\boldsymbol{c}_c$ and are always defined in relation to a human body k and a robot link i. As a result, the indices i and k are generally omitted. The indices i and k are only used in cases where an explicit distinction between the individual links i and bodies k is required.

Another important parameter in the safety considerations is the so-called collision angle φ_c, which is defined between the direction of the robot's motion and the collision vector $\boldsymbol{c}_c$. It holds:

$$\varphi_c = \cos^{-1}\left(\frac{\boldsymbol{v}_{r,i}\,\boldsymbol{c}_c}{|\boldsymbol{v}_{r,i}|\;|\boldsymbol{c}_c|}\right) \tag{5}$$

For the directed robot speed, it applies depending on the collision angle φ_c:

$$v_{r,c}\begin{cases} > 0, & \text{if } \varphi_c < \frac{\pi}{2} \quad (6a)\\ = 0, & \text{if } \varphi_c = \frac{\pi}{2} \quad (6b)\\ < 0, & \text{if } \varphi_c > \frac{\pi}{2} \quad (6c)\end{cases}$$

If the collision angle φ_c is smaller than $\frac{\pi}{2}$, the robot moves towards the human and the directed robot speed is positive. In case $\varphi_c = \frac{\pi}{2}$, the directed speed becomes exactly zero. This means that at this point the robot does not move towards or away from the human. If the collision angle φ_c becomes larger than $\frac{\pi}{2}$, $v_{r,c}$ becomes negative. In this case, the robot moves away from the human, so that there exists no immediate danger from the robot. A consideration of the collision angle φ_c is sufficient in many cases to get a simple evaluation whether the robot represents a danger for the human.

3.3 Identification of Critical Points

Considering multiple points on the robot and human kinematics, it is necessary to identify relevant points in order to perform the separation distance calculations between the human and the robot (see Fig. 1). A consideration of all points also leads to an immensely high calculation effort, as it can be seen in Fig. 1a. To identify these critical points, in many applications often the two points of the human and the robot are considered which have the smallest Euclidean distance to each other. In the preliminary work of [23], however, it is shown that the reference points on the robot kinematics can vary within a given robot path. The identification of the most critical points does not only depend on the shortest distance between the human and the robot; rather, the directed relative speed and the current collision direction must be taken into account.

For each human body k and each robot link i, we determine a pair of points at each time step, which would collide most likely and thus has the highest collision potential. It is assumed that the separation distance of these critical points is greater than the separation distance of all remaining point combinations. The distance between two relevant points as well as their relative speeds must be considered as a selection criterion for determining the critical pair of points. For this purpose, according to [8], the time to collision between a robot link i and a human body k is used, which includes both positions and speeds equally.

$$t_c = \begin{cases} \frac{C_c}{v_c}, & \text{if } v_c > 0 \quad (7a)\\ \infty, & \text{else} \quad (7b)\end{cases}$$

For the relative speed between a robot point i and a human point k, it applies:

$$v_c = v_{r,c} + v_{h,c} \tag{8}$$

For a positive relative speed v_c, the collision time indicates the hypothetical time period until a collision occurs. A collision occurs when the human ($\rho_{h,k}$) and robot ($\rho_{r,i}$) spheres collide (see Fig. 1b). A possible collision is only considered as relevant if the conditions $v_c > 0$ and $v_{r,c} > 0$ apply. For the given condition $v_c \leq 0$, it holds $t_c \to \infty$, since a collision between the robot and the human is not possible here. The calculation from Eq. 7a is only performed for those point combinations for which the given conditions are valid. For all other points, the calculation of the time to collision is not necessary. Given t_c for all human bodies k and robot links i at time t, the minimum time to collision

$$T_c = \forall i, k \;\; \min\{t_c\} \tag{9}$$

identifies the critical points between the robot and the human. To avoid values in the infinity, the reciprocal value of the time to collision is introduced as collision rate:

$$f_c = 1/t_c \tag{10}$$

The higher the collision rate, the higher the collision potential and the more dangerous are the considered critical points. In analogy to Eq. 9, the maximum collision rate

$$F_c = \forall i, k \;\; \max\{f_c\} \tag{11}$$

serves as a selection criterion for the critical points. If two critical points are identified for a considered time step, the separation distance S_c is calculated according to Eq. 4.

4 Adaptive Safety Strategies

There are basically three different strategies to maintain the necessary separation distance S_c:

1. The robot speed has a significant influence on the separation distance. If the necessary process parameters (e.g. positions and speeds of the human and the robot) are known, the robot speed can be adapted to maintain the separation distance. The previously planned robot path remains unaffected by the speed adaption.
2. In addition to the robot speed, the robot path can be adapted to fulfil the required separation distance at any time. Here, the robot always moves at its maximum

possible speed depending on the joint configuration. In contrast to the speed adaption, the adaption of the robot path also changes many other relevant process parameters, such as the collision direction between the human and robot. This also has a strong influence on the resulting separation distance and results in a highly dynamic system with complex interactions. Furthermore, a path adaption is only suitable for very limited processes, i.e. those that do not require path constancy (path welding or gluing does not work, for instance).

3. Furthermore, both the robot path and the robot speed can be adapted together to ensure the separation distance. The objective is to find an optimum between an adaption of the robot speed and the robot path.

4.1 Conditions for Adaption

A central aspect is the examination, whether the required separation distance is maintained. There are various situations in which a collision between the human and the robot is possible. Assuming that the conditions $v_c > 0$ and $v_{r,c} > 0$ are met, the necessary condition for a safe robot motion is that the current distance must not be smaller than the separation distance. Therefore, $S_c \leq C_c$ must be valid at any time t. If this condition is fulfilled, no adaption of the robot motion is necessary. Otherwise, the robot speed or the robot path must be continuously adapted. If this is not possible to a sufficient extent, because for example the actors cannot decelerate to the required speed, an emergency stop of the robot must be performed. If, however, the conditions $v_c > 0$ and $v_{r,c} > 0$ are not fulfilled, so that there is no risk of a collision between the human and the robot, the current distance can also fall below the separation distance S_c. Nevertheless, the minimum distance S_m must be maintained at all times. The schematic procedure for robot speed adaption is shown in Fig. 2.

4.2 Robot Speed Adaption

The adapted (directed) robot speed $\widetilde{v}_{r,c}$ is the speed of the robot, which is necessary to maintain the required separation distance S_c at a considered time t. According to Eq. 4, the terms T_r, S_m and $v_{h,c}$ cannot be manipulated specifically when calculating the separation distance S_c, since they are constants or they dependent on the human motions. The braking time $T_s(v_{r,c})$ and the braking distance $B_c(v_{r,c})$ depend on the directed robot speed $v_{r,c}$. For this reason, the separation distance $S_c = S_c(v_{r,c})$ is considered as a function of the directed robot speed $v_{r,c}$. For a time-optimised robot motion, the robot speed must be specified so that $S_c(\widetilde{v}_{r,c}) \equiv C_c$ applies (see Fig. 3).

An analytical solution is described in [23]. However, strong simplifications had to be made there, such as the use of constantly high braking values for stop 0. To use the speed-dependent terms $T_s(v_{r,c})$ and $B_c(v_{r,c})$ for stop 1, a numerical solution for determining the adapted robot speed $\widetilde{v}_{r,c}$ is necessary. At this point, we use *Newton's method* [25, 26], since it is highly convergent and often requires only a few iteration steps. In order to use Newton's method, Eq. 4 is modified and described as function $f(v_{r,c})$, whose zero is investigated:

$$f(v_{r,c}) = S_c(v_{r,c}) - C_c = 0 \tag{12}$$

A drawback of the Newton's method is that it may move away from the searched solution. This can happen

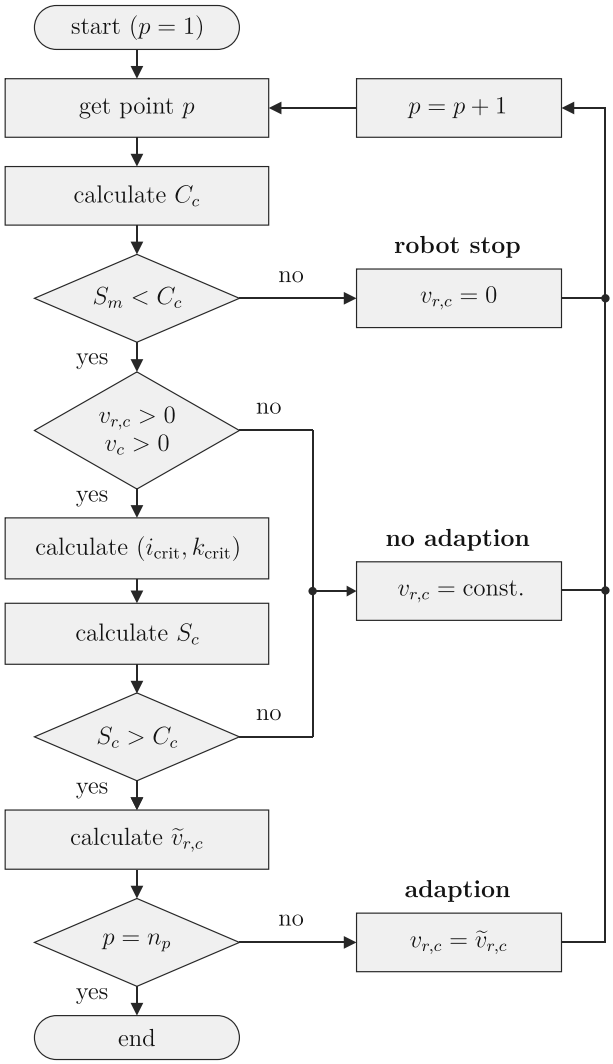
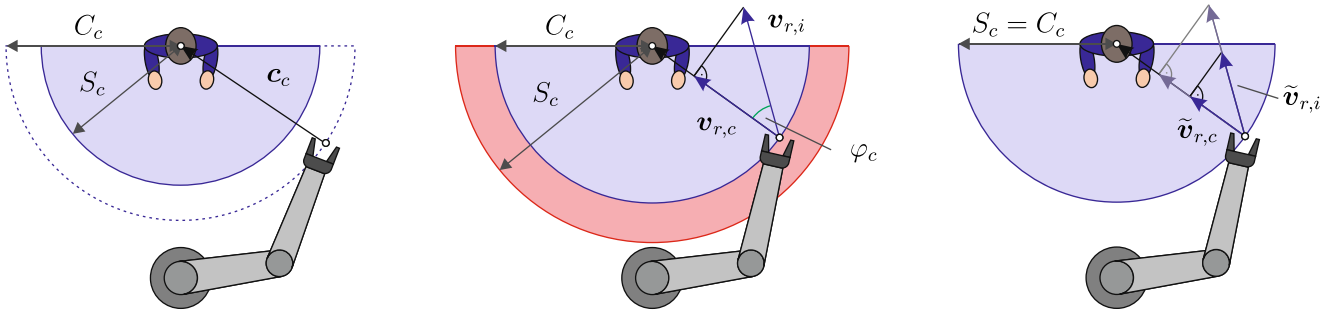


Fig. 2 Flow chart for speed adaption to ensure the separation distance, taking into account the critical points



(a) $S_c \leq C_c$: no adaption (b) $S_c > C_c$: adaption necessary (c) $S_c = C_c$: adaption successful

Fig. 3 Adaption of the robot speed when the separation distance is not maintained (if $v_{r,c} > 0$ and $v_c > 0$)

if the given function is not monotonically increasing. In the context of SSM, this problem does not occur because increased robot speeds under same process conditions lead to a greater separation distance, i.e. $f(v_{r,c})$ is a monotonically increasing and continuous function. Thus, $f(v_{r,c})$ has exactly one zero which is the required adapted robot speed, where the separation distance S_c corresponds to the current distance C_c. For each iteration step n with $f(v_n) = f(v_{r,c})$, it applies:

$$v_{n+1} = v_n - \frac{f(v_n)}{f'(v_n)} \tag{13}$$

The starting value for the iterations is $v_n = v_{r,c}$. Since $f(v_n)$ is not differentiable, we approximate the derivative $f'(v_n)$ in each iteration step n:

$$f'(v_n) \approx \frac{f(v_n) - f(v_{n-1})}{v_n - v_{n-1}} \tag{14}$$

A second starting value (e.g. $v_n \cdot 0.9$) that is required for the derivation is determined in the first iteration step. Since the function $f(v_n)$ has only one single zero, the iterations work for any other output value too.

For $|f(v_n)| < \varepsilon$, the iteration loop terminates with a permissible tolerance ε. If the calculated zero of Eq. 12 is positive, it corresponds to the adapted speed $\widetilde{v}_{r,c} \approx v_n$. On the other hand, if there is a negative value for v_n, the current distance is so far fallen below the separation distance that the speed adaption fails. Equation 12 can then only be fulfilled for speeds in the opposite direction – i.e. away from the human. A change in the direction of motion is not intended, i.e. the robot must stop immediately ($\widetilde{v}_{r,c} = 0$). The schematic procedure is illustrated in Algorithm 1.

Algorithm 1 Newton's method.

```
n = 1;
v(n) = v_{r,c};
while |f(n)| > ε do
    f(n) = S_c(v(n)) − C_c;
    if n = 1 then
        v(n + 1) = v(n) · 0.9;
    else
        f'(n) = (f(n) − f(n − 1))/(v(n) − v(n − 1));
        v(n + 1) = v(n) − f(n)/f'(n);
    end if
    n = n + 1;
end while
if v(n) > 0 then
    ṽ_{r,c} = v(n);
else
    ṽ_{r,c} = 0;
end if
```

5 Adaption of the Planned Trajectory

As shown in the previous sections, the necessary separation distance can be calculated based on the current states of the human and the robot at any time of a considered application scenario. Related to this, the maximum allowed (adapted) robot speed can be determined. The central idea now is to manipulate the (collision-free) planned robot trajectory before the motion is executed. The aim is to achieve the required robot speeds during the execution of the robot's motion and thus to maintain the necessary separation distances. The exact procedure for manipulating the planned robot trajectory is described in the following.

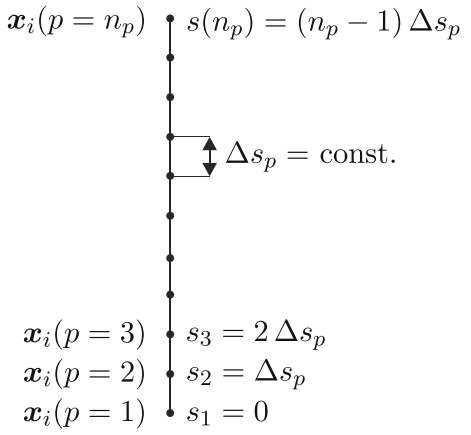
5.1 Discretisation of the Robot Trajectory

In order to perform a robot speed adaption in the context of an adaptive motion planning, the positions and velocities must be available for all robot links as well as for all human bodies. In reality, the robot motions are continuous trajectories, but in the simulation the robot trajectory is discretised into several waypoints $p \in \{1, \cdots, n_p\}$. There are two approaches to define the discrete waypoints:

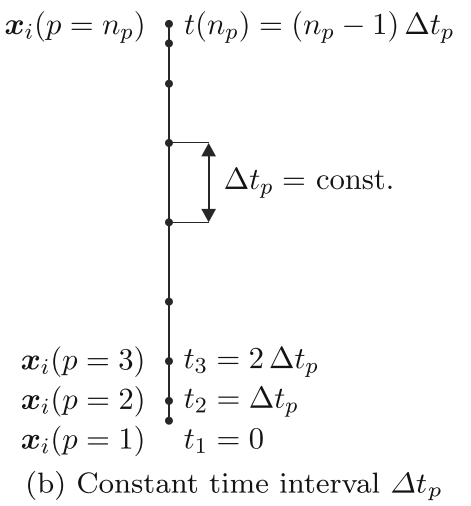
- constant path segment Δs_p between the waypoints
- constant time interval Δt_p between the waypoints

Both approaches are shown in Fig. 4. In the first case, the distance between all waypoints is always identical. However, the time intervals between these points can vary significantly. For example, at very low robot speeds, the time steps can be very far apart, as it takes a long time to cover the distance between two points. In the second case, the waypoints can be very far away from each other at very high speeds because the robot can move very far in the time interval. As a result, there are only very few waypoints at high speeds since a long distance has to be made.

Basically, the number of waypoints n_p should be kept as low as possible to achieve an acceptable computing time but as high as necessary to map the continuous (real) robot trajectory well. If the selection of a relatively small path segment Δs_p or a small time interval Δt_p generates a sufficiently high number of waypoints n_p, then both procedures can be used equally for discretising the robot trajectory.



(a) Constant path Δs_p



(b) Constant time interval Δt_p

Fig. 4 Different strategies for trajectory discretisation

5.2 Time Adaption

Due to the discretisation of the robot trajectory, the robot poses $\boldsymbol{x}_i$ are coupled to fixed time steps t_p. The time interval Δt_p between one waypoint p and the following waypoint $p + 1$ is calculated as follows:

$$\Delta t_p = t_{p+1} - t_p \tag{15}$$

By changing the robot speed at a waypoint p, this coupling is no longer valid because the corresponding robot poses cannot be reached in the same time. To ensure that the poses $\boldsymbol{x}_i$ again correspond to the time t_p, either the pose or time data at a waypoint p have to be modified. Since the pose data have many other data attached to it, such as the joint angles or the Jacobians, the more elegant solution is to adapt the time steps. To modify these time steps, the following three cases are now defined:

- speed adaption
- no speed adaption
- robot stop

5.2.1 Speed adaption

As soon as the robot speed is reduced at a waypoint p, the robot needs a longer time for the following distance to the next point $p + 1$. It is assumed that the time t_p is not affected by the speed reduction itself because the reduced speed only affects the next segment of the trajectory. Since the waypoints themselves remain unchanged by the selected time adaption strategy, the robot poses $\boldsymbol{x}_i$ and the associated joint angles $\boldsymbol{q}$ at a waypoint p are identical, even after adaption. It applies:

$$\widetilde{\boldsymbol{x}}_i = \boldsymbol{x}_i \qquad \widetilde{\boldsymbol{q}} = \boldsymbol{q} \tag{16}$$

Each Jacobian $\boldsymbol{J}_i(\boldsymbol{q})$ also depends only on the joint angles $\boldsymbol{q}$ but not on the joint angular velocities $\boldsymbol{\omega}$. Therefore, the Jacobians also remain the same after the adaption:

$$\boldsymbol{J}_i(\widetilde{\boldsymbol{q}}) = \boldsymbol{J}_i(\boldsymbol{q}) \tag{17}$$

The adapted joint angular velocities $\widetilde{\omega}_i$ for all joints i are given in the adapted joint angle vector $\widetilde{\boldsymbol{\omega}}$. The adapted robot velocity $\widetilde{\boldsymbol{v}}_{r,i}$ of one robot link i in Cartesian space is part of the adapted velocity vector $\widetilde{\boldsymbol{\xi}}_i$. With the Jacobian $\boldsymbol{J}_i$, the following well-known correlations apply:

$$\boldsymbol{\xi}_i = \boldsymbol{J}_i\,\boldsymbol{\omega} \qquad \widetilde{\boldsymbol{\xi}}_i = \boldsymbol{J}_i\,\widetilde{\boldsymbol{\omega}} \tag{18}$$

The adapted velocities $\widetilde{\boldsymbol{\xi}}_i$ and $\widetilde{\boldsymbol{\omega}}$ can be represented by the velocity scaling factors $\mu_{x,i}$ and μ_q using the initial velocities $\boldsymbol{\xi}_i$ and $\boldsymbol{\omega}$. Thus, the following applies:

$$\mu_{x,i}\,\boldsymbol{\xi}_i = \boldsymbol{J}_i\,(\mu_q\,\boldsymbol{\omega}) \tag{19}$$

The velocity scaling factors indicate the amount of reduction relative to the initial velocities. As already

mentioned, the Jacobians $\boldsymbol{J}_i$ are only dependent on the joint angles $\boldsymbol{q}$. As a result, the scaling factor μ_q, which refers directly to the joint angular velocities $\boldsymbol{\omega}$, has no effect on the Jacobians. This is due to the fact that the robot path is completely independent of the robot speed, since the same waypoints are always accessed – possibly at different time steps. We obtain:

$$\mu_{x,i}\ \boldsymbol{\xi}_i = \mu_q\ \boldsymbol{J}_i\ \boldsymbol{\omega} \tag{20}$$

With the use of Eq. 18, it follows for the velocity scaling factors:

$$\mu_{x,i} = \mu_q = \mu \tag{21}$$

The scaling factor $\mu = \mu(t_p)$ is determined for each waypoint p at the time t_p via the adapted Cartesian or directed speeds:

$$\mu = \frac{\widetilde{v}_{r,i}}{v_{r,i}} = \frac{\widetilde{v}_{r,c}}{v_{r,c}} \tag{22}$$

In general, at $\mu = 1$, the robot speed is not reduced compared to the originally planned robot trajectory. At $\mu = 0$, the robot stops completely. If the Cartesian speed $v_{r,i}$ of the robot body i is now reduced by the factor μ, then all joint angular velocities ω_i are reduced by the same factor to maintain the planned robot paths $\boldsymbol{x}_i$. The corresponding adapted joint angular velocities result from this:

$$\widetilde{\omega}_i = \mu\ \omega_i \tag{23}$$

For the assumption that very small path segments exist, i.e. $\Delta \boldsymbol{x}_i \to 0$ applies, the following calculation of the joint angular velocities is valid:

$$\omega_i = \frac{\Delta q_i}{\Delta t_p} \qquad \widetilde{\omega}_i = \frac{\Delta \widetilde{q}_i}{\Delta \widetilde{t}_p} \tag{24}$$

If these quotients are now expressed in Eq. 23, the calculation formula of the velocity scaling factor μ is obtained from the given time steps:

$$\mu = \frac{\widetilde{\omega}_i}{\omega_i} = \frac{\Delta \widetilde{q}_i / \Delta \widetilde{t}_p}{\Delta q_i / \Delta t_p} \tag{25}$$

Since the joint angles – even after a speed adaption – are identical for a certain waypoint p, so $\Delta q_i = \Delta \widetilde{q}_i$, thus follows:

$$\mu = \frac{\Delta t_p}{\Delta \widetilde{t}_p} \tag{26}$$

From this, the adapted time steps can be determined:

$$\widetilde{t}_{p+1} = \widetilde{t}_p + \frac{1}{\mu_p} \Delta t_p \tag{27}$$

5.2.2 No speed adaption

If the robot speed is not reduced in a waypoint p, i.e. it is a safe waypoint, the time interval Δt_p to the next waypoint does not change with respect to the initial time interval. Nevertheless, the initial time t_{p+1} cannot be used in this case. This is due to the fact that the current time $\widetilde{t}_p$ does not necessarily correspond to the time t_p because of a possible speed reduction in previous waypoints. The time $\widetilde{t}_{p+1}$, if there is no speed adaption in point p, can be determined as follows:

$$\widetilde{t}_{p+1} = \widetilde{t}_p + \Delta t_p \tag{28}$$

5.2.3 Robot stop

If the robot falls below the separation distance S_c so much, that even the speed adaption can no longer generate a positive solution for $\widetilde{v}_{r,c}$, the robot must stop immediately. Hence, $\widetilde{\boldsymbol{\xi}}_i = 0$ and $\widetilde{\boldsymbol{\omega}} = 0$ applies. Now, we have to calculate the time $\widetilde{t}_{p+1}$ when the robot can start moving again. Since the robot itself is standing still, we have to wait until the human has a greater distance C_c to the robot than the separation distance S_c, or until the human and the robot move away from each other ($v_c \leq 0$). The waiting time t_{wait} for the robot stop is determined iteratively (see Algorithm 2). The time at which the robot can restart is given by:

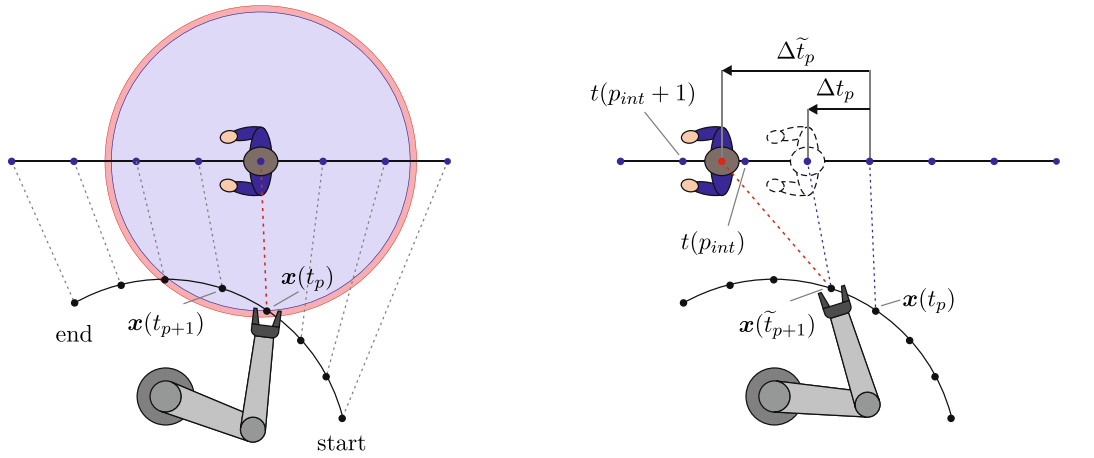
$$\widetilde{t}_{p+1} = \widetilde{t}_p + t_{\text{wait}} \tag{29}$$

Algorithm 2 Robot stop.

Input: $p, i, k, \widetilde{t}_p, v_{r,c} = 0, \Delta t = 0.01$;
  $t_{\text{wait}} = 0$;
  while $(S_c > C_c) \wedge (v_c > 0)$ **do**
    $\widetilde{t}_{p+1} = \widetilde{t}_p + t_{\text{wait}}$;
    $(\boldsymbol{r}_{h,k}, \boldsymbol{v}_{h,k}) \leftarrow$ `setAdaptedHumanStates` $(p, k, \widetilde{t}_{p+1})$;
    $C_c \leftarrow$ `getDistance`(p, i, k);
    $S_c \leftarrow$ `getSeparationDistance`$(p, i, k, v_{r,c})$;
    $v_c \leftarrow$ `getDirectedRelativeSpeed`(p, i, k);
    $t_{\text{wait}} = t_{\text{wait}} + \Delta t$;
  end while

5.3 Adaption of the Human Motion

Human motions cannot be exactly predicted in the real execution of the robot path. Assuming that the human motions are known in the simulation, the human motions need to be discretised in the same way as the robot motions. In addition, the human points must be synchronised with the points of the robot (see Fig. 5a), in order to have an identical relation to the given time steps. The positions and velocities of all human bodies are then determined for these waypoints. The human moves completely independent of the robot path, so the robot speed adaption has no direct influence on the human motions. However, the human



(a) Separation distance is not maintained; time t_p immediately before speed adaption

(b) Speed is reduced; adapted time $\widetilde{t}_{p+1}$ after speed adaption

Fig. 5 Time delay due to the robot speed adaption at a waypoint p

motions are coupled to the robot trajectory via the discrete time steps. Therefore, the human and the robot are at the same time t_p at their respective waypoint p. Since the time steps t_p have been adapted, the human positions $\boldsymbol{r}_{h,k}$ and velocities $\boldsymbol{v}_{h,k}$ must be related to the adapted time steps $\widetilde{t}_p$ (see Fig. 5).

Since the human motions themselves remain the same, the motion data for the adapted time steps can be obtained by the original motion functions. Here, a simple linear interpolation is suitable, which is sufficiently accurate for the assumption of very small path segments ($\Delta \boldsymbol{x}_i \rightarrow 0$). The adapted time $\widetilde{t}_p$ is no longer located at waypoint p but between two later points p_{int} and $p_{int}+1$ (see Fig. 5b), for which the following applies:

$$t(p_{int}) \leq \widetilde{t}_p \leq t(p_{int}+1) \tag{30}$$

Between these two points, the human positions and velocities are interpolated. Subsequently, the curves of $\boldsymbol{r}_{h,k}(t_p)$ and $\boldsymbol{r}_{h,k}(\widetilde{t}_p)$, as well as $\boldsymbol{v}_{h,k}(t_p)$ and $\boldsymbol{v}_{h,k}(\widetilde{t}_p)$ are identical again, but the discrete motion data is now related to the adapted time steps.

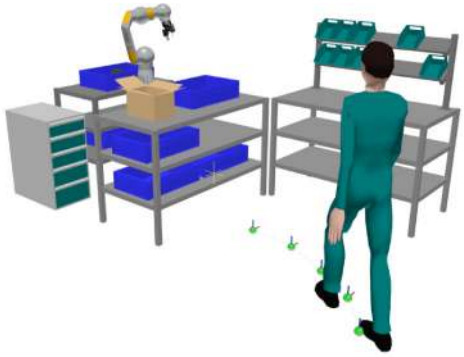
6 Implementation

One objective of this paper is to integrate the developed methods for the adaptive motion planning into an existing HRI simulation tool [2–4]. The central part of this simulation tool is a robot and peripheral simulation based on the software framework *Robot Operating System* (ROS) [27]. All methods for processing the robot and peripheral simulation are integrated into a ROS workspace (`kompi_ws`). A further component of the simulation tool is a human and process simulation, which is based on the *ema Work Designer* (EMA) [28, 29] from imk automotive GmbH. An essential part of the simulation tool is a developed data interface between EMA and ROS. The main package of `kompi_ws` is `kompi_interface`, which is responsible for the communication between EMA and ROS. In addition, this interface package controls the robot and gripper actions.

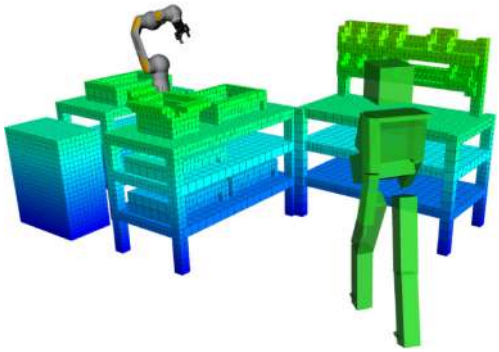
The basic procedure of the simulation is as follows: The production planner in EMA models the entire application scenario with all human, robot and environmental models as well as task descriptions of the human (e.g. `PickObject`) and the robot (e.g. `MoveRobot PTP`). All process, environmental and human data are transferred from EMA to ROS, where the robot and peripheral simulation takes place. Afterwards, ROS sends the calculated robot and peripheral data back to EMA via the data interface. Finally, human, robots and peripherals are controlled and visualised in the simulation environment of EMA.

6.1 Environmental and Human Model

An essential requirement for executing the robot motions without collisions, is the knowledge of the human and environmental data in the given process. All environmental data is stored in a 3D voxel field generated by EMA. The digital human model from EMA is approximated and transferred to ROS as a simplified hull geometry. To describe the approximated human model, EMA provides the human data (poses, dimensions etc.) for $n_k = 18$ human bodies with its corresponding parameters at each simulation time step. Figure 6 shows how the environmental and human models from EMA are reconstructed as a virtual image in ROS.



(a) Modeled HRI scenario in EMA



(b) Approximated HRI scenario in ROS

Fig. 6 Transfer of the environmental model as 3D voxel field and transfer of the human model as simplified hull geometry

6.2 Motion Planning

The framework *MoveIt!* serves as the central element for motion planning in ROS. As a meta package, *MoveIt!* combines the current algorithms for motion planning, manipulation, 3D perception, kinematics, control and navigation of robots. For motion planning, *MoveIt!* uses the *Open Motion Planning Library* (OMPL) [30, 31] by default, which has a large number of algorithms for collision-free motion planning. As a standard (collision-free) path planning algorithm, this paper uses *RRT-Connect* [32]. The *Trajectory Processing Routine* in *MoveIt!* handles the time aspects of motion planning. Taking into account the velocity and acceleration limits of the individual joints, this routine calculates a suitable time-parameterised trajectory.

6.3 Adaptive Motion Planning

A new component of the ROS workspace `kompi_ws` is the package `kompi_speed_adaption`. The implemented node `adaptive_motion_planner` contains the following methods:

- `Human::getHumanStates`
- `Robot::getRobotStatesPlan`
- `SpeedAdaption::calcSpeedAdaption`

The callback method `getHumanStates` receives and stores the human motion data $(\boldsymbol{r}_{h,k},\ \boldsymbol{v}_{h,k})$ for all human bodies k coming from EMA. The callback method `getRobotStatesPlan` receives and stores the motion data $(\boldsymbol{q},\ \boldsymbol{\omega})$ of the robot joints for the planned, unadapted trajectory coming from ROS. Once the joint data have been received, we calculate the Cartesian positions and velocities $(\boldsymbol{r}_{r,i},\ \boldsymbol{v}_{r,i})$ for all robot links i. Within the method `calcSpeedAdaption`, the entire adaptive motion planning for all waypoints p, all human bodies k and all robot links i is then performed (see Algorithm 3). For the critical points $(i_{\text{crit}}, k_{\text{crit}})$, the internal method `getAdaptedSpeed` calculates the adapted robot speed on the basis of the previously planned trajectory $\boldsymbol{T}_\lambda$.

In order to easily exchange the multidimensional data between the individual subprograms, a new message type `SpeedAdaption.msg` is defined. A part of this message with its relevant data types is shown in Fig. 7. An access to the current distance C_c between the human body k and the robot link i for a specific waypoint p is for example done by calling:

```
msg.points[p].links[i].bodies[k].
distance;
```

6.3.1 Multiple trajectories

As can be seen in Algorithm 4, a total number of n_λ different trajectories are planned for each application scenario. The ROS motion planning algorithm generates a set of discrete waypoints that connects the start and target points in the best possible way via the command `move_group.plan`.

Algorithm 3 Adaption of a trajectory.

for $p = 1$ **to** n_p **do**
  for $k = 1$ **to** n_k **do**
    $(\boldsymbol{r}_{h,k}, \boldsymbol{v}_{h,k}) \leftarrow$ `setAdaptedHumanStates` (p, k, t_p);
    for $i = 1$ **to** n_i **do**
      $C_c \leftarrow$ `getDistance`(p, i, k);
      $v_{r,c} \leftarrow$ `getDirectedRobotSpeed` (p, i, k);
      $t_c \leftarrow$ `getTimeToCollision`(p, i, k);
      $f_c \leftarrow$ `getCollisionRate`(p, i, k);
    end for
  end for
  $(i_{\text{crit}}, k_{\text{crit}}) \leftarrow$ `getCriticalPoints`(p);
  $S_c \leftarrow$ `getSeparationDistance` $(p, i_{\text{crit}}, k_{\text{crit}}, v_{r,c})$;
  if $(S_c > C_c) \wedge (v_{r,c} > 0) \wedge (v_c > 0)$ **then**
    $\widetilde{v}_{r,c} \leftarrow$ `getAdaptedSpeed`$(p, i_{\text{crit}}, k_{\text{crit}})$;
  end if
end for

```
kompi_msgs/SpeedAdaption.msg
└─ SpeedAdaptionPoint[] points
   ├─ int32 point
   ├─ float64 timestep
   ├─ float64 velocity_scaling
   └─ SpeedAdaptionLink[] links
      ├─ float64 robot_adapted_speed
      └─ SpeedAdaptionBody[] bodies
         ├─ float64 separation_distance
         ├─ float64 distance
         ├─ float64 collision_angle
         ├─ float64 time_to_collision
         ├─ float64 collision_rate
         ├─ float64 directed_relative_speed
         ├─ float64 directed_robot_speed
         └─ float64 directed_human_speed
```

Fig. 7 Composition of the message `SpeedAdaption.msg` with the associated data types for transmitting the adapted motion data

However, if the plan fails (`error` $\neq$ 1), because e.g. a collision between a robot link and an environmental object occurs, this planned trajectory must be rejected. The various (successfully planned) trajectories $\boldsymbol{T}_\lambda$ are then adapted within the method `adaptTrajectory`, if the safety requirements are not met. This is done by adapting the time steps of all waypoints (see Section 5.2). The adapted cycle times $\widetilde{t}_\lambda$ are used as a criteria for selecting a trajectory to be executed by the robot. The adapted trajectory $\widetilde{\boldsymbol{T}}_{\lambda_{min}}$ with the shortest cycle time after adaption

$$\widetilde{t}_{min} = \forall\lambda \ \min\{\widetilde{t}_\lambda\} \tag{31}$$

is then executed via `move_group.execute`.

7 Simulation and Analysis

In the following section, the methods presented in this paper will be analysed using an application example within a simulation study. As it can been seen in Fig. 8, the application example considers a shared collaborative workstation between a human and a robot. The robot used is a conventional six-axis industrial robot KUKA KR 16-2. By applying the collaborative operation of SSM, it is even possible to use conventional robots with higher payloads.

In the application example, the robot moves along a trajectory $\boldsymbol{x}(t)$ from a start pose $\boldsymbol{x}(t_0)$ to a target pose $\boldsymbol{x}(t_e)$. The planned robot trajectory is discretised with a constant path Δs_p between the single trajectory points (see Section 5.1). Related to this is a set of motion data for all human bodies. The human first moves strongly towards the robot, up to a point with minimal distance. As the human continues to move past the robot in the same direction, both move away from each other at the end of the application example. In addition, the example considers two different application scenarios:

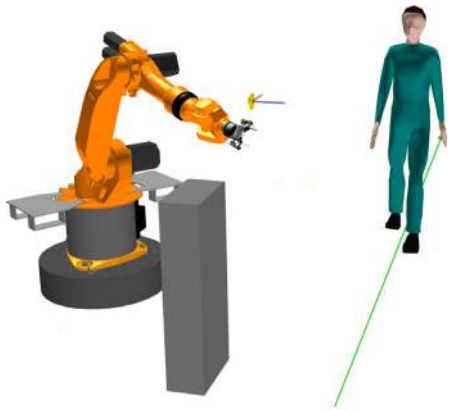
– no obstacle in the collaborative workspace

Algorithm 4 Calculation of multiple trajectories.

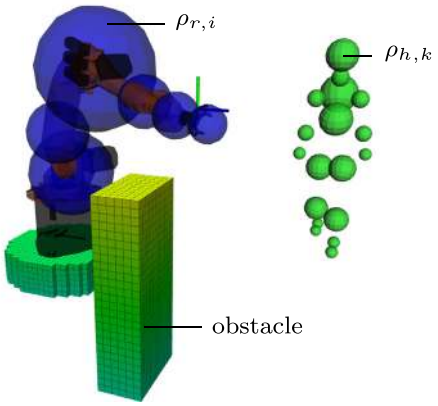
for $\lambda = 1$ **to** n_λ **do**
  while `error` $\neq 1$ **do**
    `error` $\leftarrow$ `move_group.plan`$(\boldsymbol{T}_\lambda)$;
    $(\widetilde{\boldsymbol{T}}_\lambda, \widetilde{t}_\lambda) \leftarrow$ `adaptTrajectory`$(\boldsymbol{T}_\lambda)$;
  end while
end for
$(\widetilde{t}_{\min}, \lambda_{\min}) \leftarrow \min(\widetilde{t}_\lambda)$;
`error` $\leftarrow$ `move_group.execute`$(\widetilde{\boldsymbol{T}}_{\lambda_{\min}})$;

– obstacle in the collaborative workspace

In this case, the obstacle does not refer to the human model but to all environmental objects in the proximity of the robot which can affect and possibly restrict the robot motions. Thus, the presence of an obstacle has a great influence on the robot's motion planning.



(a) Modeled HRI scenario in EMA



(b) Approximated HRI scenario in ROS

Fig. 8 Transfer of the human, robot and environmental model for the adaptive motion planning; the human model is approximated as simplified hull geometry ($n_k = 18$ spheres with radii $\rho_{h,k}$); all environmental objects (e.g. the obstacle) are represented by a 3D voxel field; the robot is surrounded by $n_i = 6$ enveloping spheres with radii $\rho_{r,i}$

Table 1 Joint angles q_i for the target joint configurations of the corresponding trajectory $\boldsymbol{T}_\lambda$

$\boldsymbol{T}_\lambda$	q_1	q_2	q_3	q_4	q_5	q_6
$\boldsymbol{T}_1$	−90°	−65°	110°	180°	−46°	−270°
$\boldsymbol{T}_5$	−90°	−65°	110°	−180°	−46°	−270°
$\boldsymbol{T}_7$	−90°	−65°	110°	0°	46°	−90°
$\boldsymbol{T}_{59}$	90°	−154°	−44°	−180°	72°	270°
$\boldsymbol{T}_{76}$	90°	−154°	−44°	180°	72°	270°
$\boldsymbol{T}_{80}$	90°	−154°	−44°	0°	−72°	90°

In both scenarios, we plan and analyse $n_\lambda = 100$ trajectories with regard to the speed adaption. The used computer platform is an Ubuntu (16.04) machine with a Core i7-7820HQ 2.90 GHz processor, the concerning ROS distribution is Kinetic Kame. The total computation time for the speed adaption of the 100 trajectories was 36.38 s. Averaged over all trajectories, this results in 2.45 ms for a single waypoint.

7.1 Application Scenario without Obstacles

In the first case, the adaptive motion planning and execution of the robot consider no obstacles in the collaborative workspace.

7.1.1 Adaptive motion planning

Comparing the planned robot trajectories $\boldsymbol{T}_\lambda$, it is noticeable that many of the planned robot paths have an identical profile. This is due to the fact that the motion planning algorithm always generates the kinematically most feasible trajectory in an obstacle-free environment, i.e. the trajectories in which

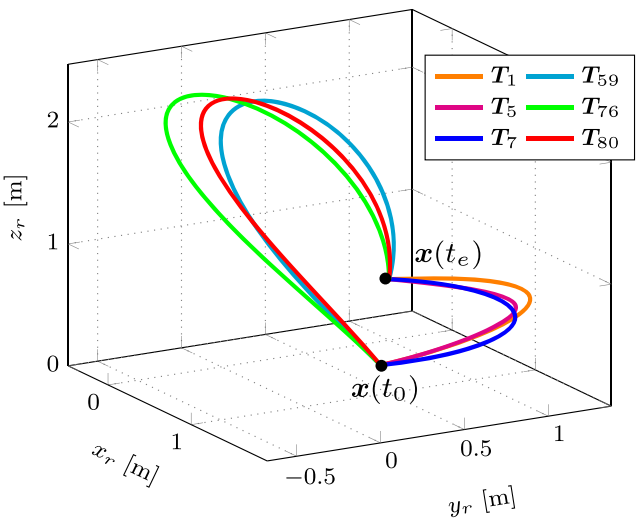


Fig. 9 Various trajectories $\boldsymbol{T}_\lambda$ with the corresponding robot positions $\boldsymbol{r}_r = (x_r, y_r, z_r)$ with regard to the world coordinate frame

Table 2 Comparing categories I, II and III regarding the travelled distances and cycle times

Category	I	II	III
λ	76	80	7
s_λ	5.65 m	5.33 m	2.72 m
t_λ	4.89 s	3.74 s	3.15 s
$\widetilde{t}_\lambda$	4.89 s	4.14 s	4.06 s
Δt_λ	0 s (0%)	0.40 s (10.8 %)	0.91 s (28.6 %)

the joints perform the fastest and smoothest motions. In the present case, six different trajectories and target joint configurations can be identified, which are listed in Table 1.

In Fig. 9, each of the six different trajectories is highlighted. It can be seen that the travelled distances s_λ are of different lengths. The shortest path ($\boldsymbol{T}_7$) is only 2.72 m long, whereas the longest path ($\boldsymbol{T}_{76}$) with 5.65 m is more than twice as long. Nevertheless, it cannot be assumed that a shorter distance necessarily results in a shorter cycle time. The time to execute a trajectory also depends on the robot dynamics, i.e. the maximum velocities and accelerations of the joints in the particular configuration. In addition, the cycle times in the context of SSM depend on whether and to which extent the robot speed has to be reduced due to the required separation distance, and even if a complete robot stop is necessary. Therefore, the trajectories are divided into three categories:

– Category I: no speed adaption
– Category II: speed adaption (without robot stop)

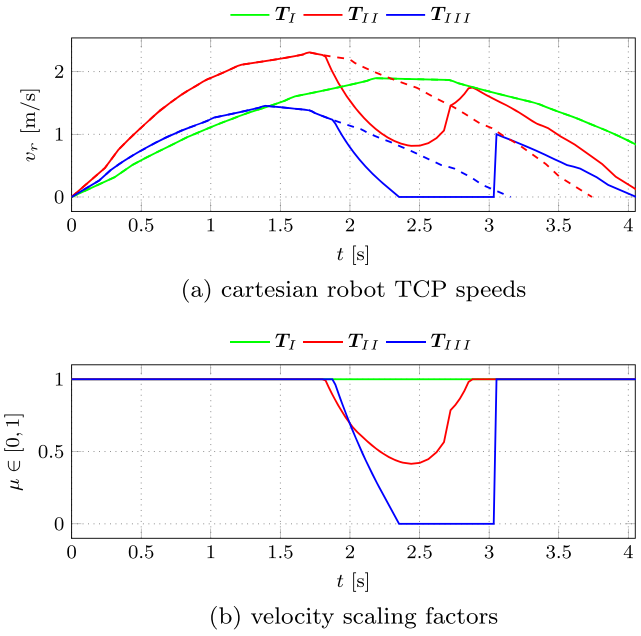
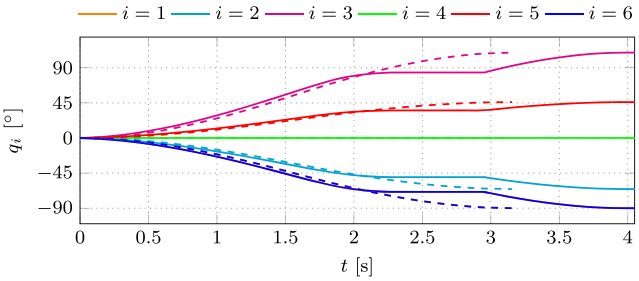
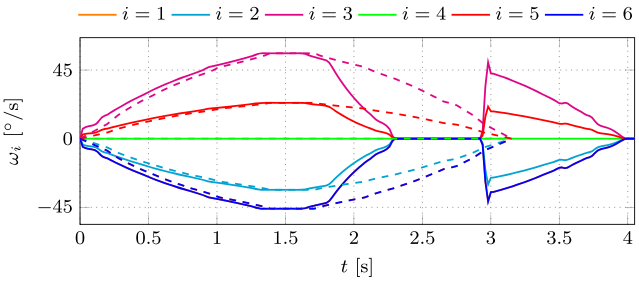


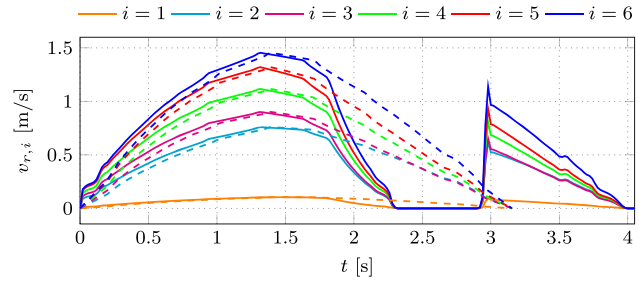
Fig. 10 Comparing the trajectories of categories I, II and III for the initial plan (−−) and the adapted execution with regard to the robot speeds and velocity scaling factors



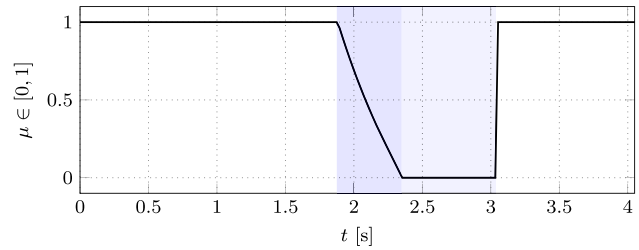
(a) joint angles



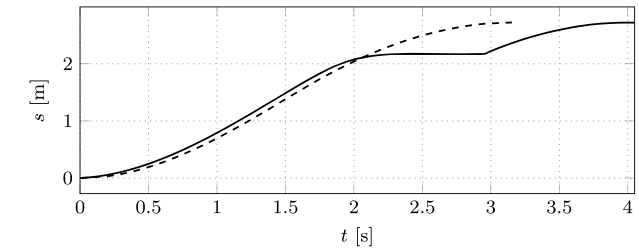
(b) joint angular velocities



(c) cartesian robot speeds



(d) velocity scaling factor



(e) distance travelled

Fig. 11 Robot states for the initial plan (−−) and the adapted execution for trajectory $\boldsymbol{T}_{III}$

– Category III: speed adaption (with robot stop)

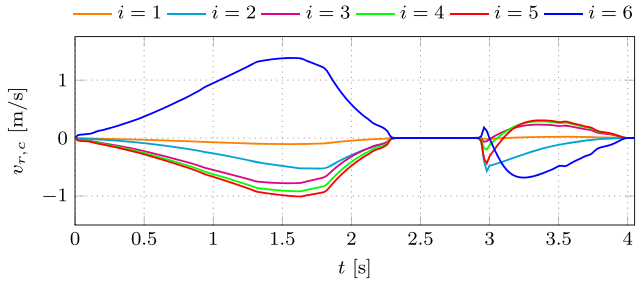
Of all 100 trajectories, only three do not require any speed adaption (category I). Another seven trajectories can be assigned to category II. Here, the speed must be adapted in parts, but a complete robot stop is not necessary. The remaining 90 trajectories require a complete robot stop to ensure a safe execution (category III). Figure 15a shows the trajectories with their corresponding cycle times (with and without speed adaption).

In the following, the trajectory for each of the three categories is examined more detailed, which has the shortest cycle time $\widetilde{t}_\lambda$ after adaption. The trajectories $\boldsymbol{T}_I$, $\boldsymbol{T}_{II}$ and $\boldsymbol{T}_{III}$ or the corresponding adapted trajectories $\widetilde{\boldsymbol{T}}_I$, $\widetilde{\boldsymbol{T}}_{II}$ and $\widetilde{\boldsymbol{T}}_{III}$ can be seen in Table 2. The TCP speeds and velocity scaling factors of the three trajectories are shown in Fig. 10. A first noticeable feature of the speed characteristics is the different acceleration behaviour of the three trajectories (see Fig. 10a). This is due to the fact that all three trajectories use different joint configurations and the acceleration capacity depends strongly on the configuration. In this case, the trajectory $\boldsymbol{T}_{II}$ reaches the highest maximum speed, whereas $\boldsymbol{T}_I$ and $\boldsymbol{T}_{III}$ move much slower. The cycle time for $\boldsymbol{T}_{III}$ is the shortest in the plan, followed by $\boldsymbol{T}_{II}$ and $\boldsymbol{T}_I$. While $\boldsymbol{T}_I$ can be moved exactly according to the plan, i.e. it does not require any adaption ($\boldsymbol{T}_I = \widetilde{\boldsymbol{T}}_I$), the speeds for $\boldsymbol{T}_{II}$ and $\boldsymbol{T}_{III}$ must be adapted. In this example, it is indeed valid that the fastest trajectory $\boldsymbol{T}_{III}$ also covers the shortest distance.

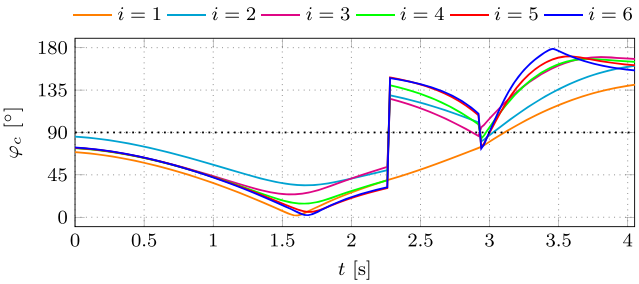
For the trajectory $\widetilde{\boldsymbol{T}}_{II}$, the speed adaption starts after 1.83 s. As can be seen in Fig. 10b, the velocity scaling factor reaches an absolute minimum of 0.42 for $\widetilde{\boldsymbol{T}}_{II}$. Afterwards, the scaling factor gradually increases again to 100%, so that from the time 2.88 s on, the trajectory can be proceeded according to plan. A complete robot stop can be avoided, even if the current distance temporarily falls below the separation distance. This is different for $\widetilde{\boldsymbol{T}}_{III}$, where the trajectory needs to be adapted from 1.90 s. Despite a strong speed adaption, the necessary separation distance cannot be maintained from 2.35 s, so that μ drops to zero. Thus, the robot stops completely. In this case, it takes up to $t_{\text{wait}} = 0.70$ s until the robot can move again. After the speed adaption, $\widetilde{\boldsymbol{T}}_{III}$ has the shortest cycle time, although a robot stop was necessary and the time was extended the most. $\widetilde{\boldsymbol{T}}_I$ is, although no adaption is necessary, the worst choice in terms of cycle times. In this situation, $\widetilde{\boldsymbol{T}}_{III}$ is therefore considered as the best alternative and is executed in ROS.

7.1.2 Adaptive motion execution

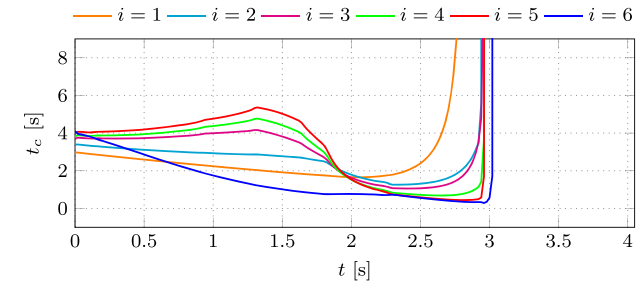
The executed trajectory $\widetilde{\boldsymbol{T}}_{III}$ is shown in Fig. 11. The joint angles in Fig. 11a change from the start configuration $\boldsymbol{q}(t_0)$



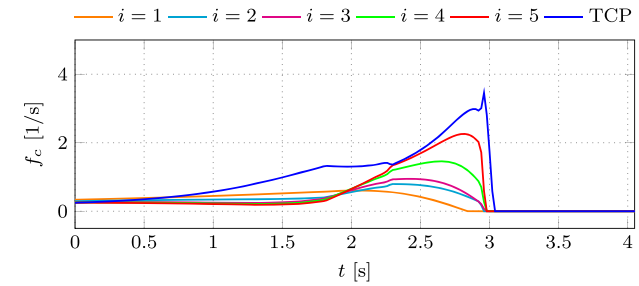
(a) directed robot speeds



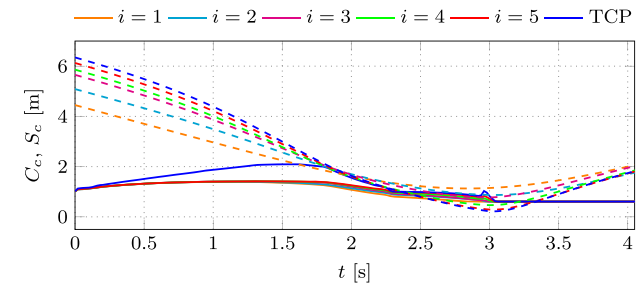
(b) collision angles



(c) time to collisions

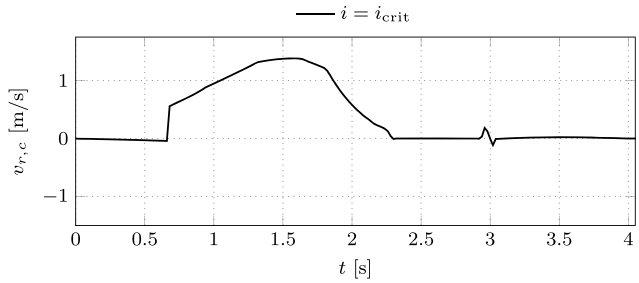


(d) collision rates

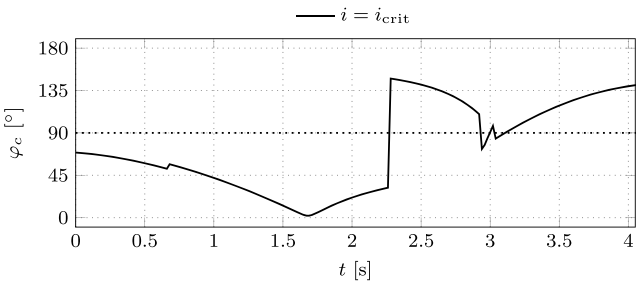


(e) current and separation distances

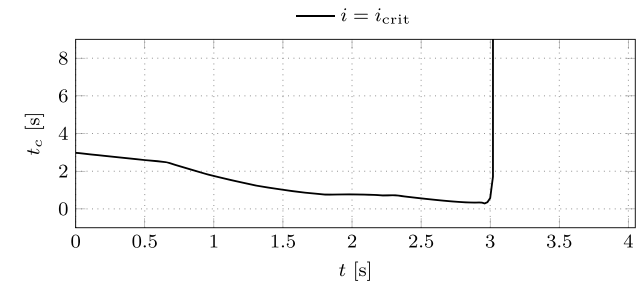
Fig. 12 States of the adaptive motion planning for all robot links with respect to the critical human body for the executed trajectory $\widetilde{\boldsymbol{T}}_{III}$



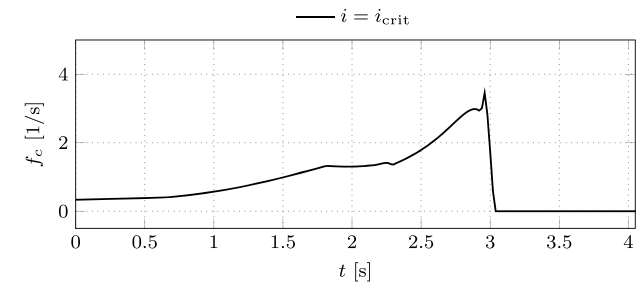
(a) critical directed robot speed



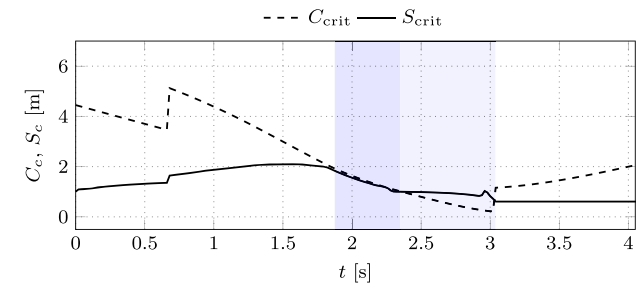
(b) critical collision angle



(c) critical time to collision



(d) critical collision rate



(e) critical current and separation distance

Fig. 13 States of the adaptive motion planning for the critical points for the executed trajectory $\widetilde{\boldsymbol{T}}_{III}$

to their target values. The motion has the typical shape for PTP motions in joint space, but it is interrupted by a robot stop. The robot stop causes the joint angles to remain constant in the time period t_{wait}, which is illustrated by Fig. 11b and c. The joint angular velocities are zero at the start and end positions but additionally for the time of the robot stop, so that the joint angles cannot change here. Directly before the robot stops, the speed adaption can be seen. Here, the robot speeds gradually become more and more different from the plan and finally turn into a stop. This characteristic can also be recognised in the velocity scaling factor in Fig. 11d. Two types of zones are highlighted here: the adaptation zone ($0 < \mu < 1$), where a reduced robot speed is applied, and the stop zone, where the robot performs a complete robot stop ($\mu = 0$). It also shows that there is only one velocity scaling factor, which affects all joints to the same extent. This ensures that the robot maintains its predetermined path despite variations in the robot speed. The distance travelled after the adaption is accordingly the same as shown in Fig. 11e, but it is shifted in time by 0.90 s due to the adaption and the stop.

Figure 12 explains in detail how the adaption and the robot stop work. The diagrams shown here all refer to the critical human body at a certain time step t_p. In principle, each of the six robot links can be potentially dangerous. Therefore, the directed speeds (Fig. 12a) and collision angles (Fig. 12b) of all links must be known. The speed adaption should then be related to the critical link or joint, i.e. to the part of the robot that potentially poses the greatest danger to the related human body. The velocity scaling factor ensures that all links are taken into account equally, so that even less dangerous links are finally safe. For the adaption, the critical link must be checked again for each point of the path. Figure 12c shows that the robot TCP ($i = 6$) is usually critical in the given trajectory. Only for a small area at the beginning, the first link is critical. It is noticeable that especially the links which are close to the TCP show a similar behaviour. The same results can be seen for the collision rates f_c in Fig. 12d. The calculation of the separation distances and the resulting speed adaption is performed for the corresponding critical robot link and the corresponding critical human body. As shown in Fig. 12e, the separation distances S_c are maintained for all robot links as soon as the speed adaption of the critical link becomes active. An exception occurs in the case of a robot stop. This will be explained in more detail later on.

Figure 13 shows the relevant curves related to the critical robot link (and the corresponding critical human body). A change in the critical points is often reflected in a function step in the various charts. For example, the change in the critical points at 0.66 s results in a function step in the course of the critical directed robot speed, in the critical collision angle and in the critical separation distance (see Fig. 13a, b and e). Furthermore, four different areas can be identified in Fig. 13e. In the first area, it applies: $C_c > S_c$. Therefore, no robot speed adaption is necessary in this area. In the second area, the curve of S_c follows the distance C_c. This is again highlighted as an adaption zone similar to Fig. 13d. Here, the speed adaption takes place without a robot stop. It shows how Newton's method (see Section 4.2) works. By searching for the zero of the function $f(v_{r,c})$, $v_{r,c}$ is calculated exactly as follows: $C_c \approx S_c$. In Table 3, the iteration steps are shown as an example for the waypoint $p = 75$. The strongly convergent behaviour of Newton's method is also presented. In the given example, only $n = 5$ steps are necessary to achieve the required tolerance ε. For the other waypoints, the speed adaption is also performed, up to the point where $\widetilde{v}_{r,c}$ is calculated to zero. At this point, the stop zone begins: the speed can no longer be reduced and the robot stops completely. As soon as the robot stops, the current distance may fall below the separation distance. As a result, the following applies in this third area: $C_c < S_c$. Although the robot stops, C_c changes continuously as the human moves on. The last area is characterised by the fact that the human now finally moves past the robot. This can also be seen from the collision angle, which is again greater than 90 °, i.e. the human and the robot move away from each other. As a result, it applies: $t_c \to \infty$ and $f_c \to 0$. From now on, the robot can start moving again.

Table 3 Applying the Newton's method at n iteration steps to determine the adapted robot speed $\widetilde{v}_{r,c} = v_n$ for the critical points ($i_{\text{crit}}, k_{\text{crit}}$) as a function of the separation distance $S_c(v_n)$; the initial (directed) robot speed is given by $v_{r,c}$; in addition, the following applies: $f(n) = S_c - C_c$ and $\mu = v_{n+1}/v_{r,c}$

p	n	i_{crit}	k_{crit}	S_c [m]	C_c [m]	$f(n)$ [m]	$v_{r,c}$ [m s^{-1}]	v_{n+1} [m s^{-1}]	μ	$\widetilde{S}_c$ [m]	$\widetilde{S}_c - C_c$ [m]
75	1	6	1	1.8498	1.3388	0.5110	0.9709	0.8738	0.9000	1.7822	0.4434
75	2	6	1	1.7822	1.3388	0.4434	0.9709	0.2367	0.2437	1.2544	-0.0844
75	3	6	1	1.2544	1.3388	-0.0844	0.9709	0.3385	0.3487	1.3500	0.0112
75	4	6	1	1.3500	1.3388	0.0112	0.9709	0.3266	0.3363	1.3390	0.0002
75	5	6	1	1.3390	1.3388	0.0002	0.9709	0.3263	0.3361	1.3388	0.0000
75	6	6	1	1.3388	1.3388	0.0000	-	-	-	-	-

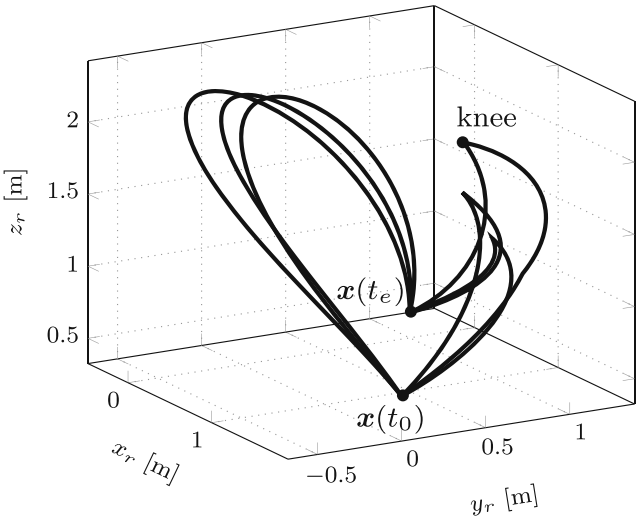


Fig. 14 Various trajectories $\boldsymbol{T}_\lambda$ with the corresponding robot positions $\boldsymbol{r}_r = (x_r, y_r, z_r)$ with respect to the world coordinate frame

7.2 Application Scenario with Obstacle

In the second application scenario, an obstacle is placed between the start and target pose of the robot so that the robot is restricted in its motion execution. This results in a change in the motion planning algorithm of ROS because the most kinematically feasible trajectories (as in the case without obstacle) may no longer be possible. If there is a collision with the object, the robot must plan another trajectory from that point on. As a result, many different trajectories are generated randomly with different joint configurations. Again, a number of $n_\lambda = 100$ trajectories are created, of which only a small part is shown in Fig. 14 for the sake of clarity.

It can be seen that some trajectories have knees. These knees are caused by the fact that the robot has to stop at these points and has to plan a new trajectory from there to avoid a potential collision with the obstacle. Accordingly, the entire trajectory is composed of two individual segments. Other trajectories, on the other hand, still manage without knees. Consequently, the different trajectories are divided into two further categories:

- Category A: no knee in the curve
- Category B: knee in the curve

In the present scenario, 15 of the 100 trajectories are for category A (without knee) and the other 85 for category B (with knee).

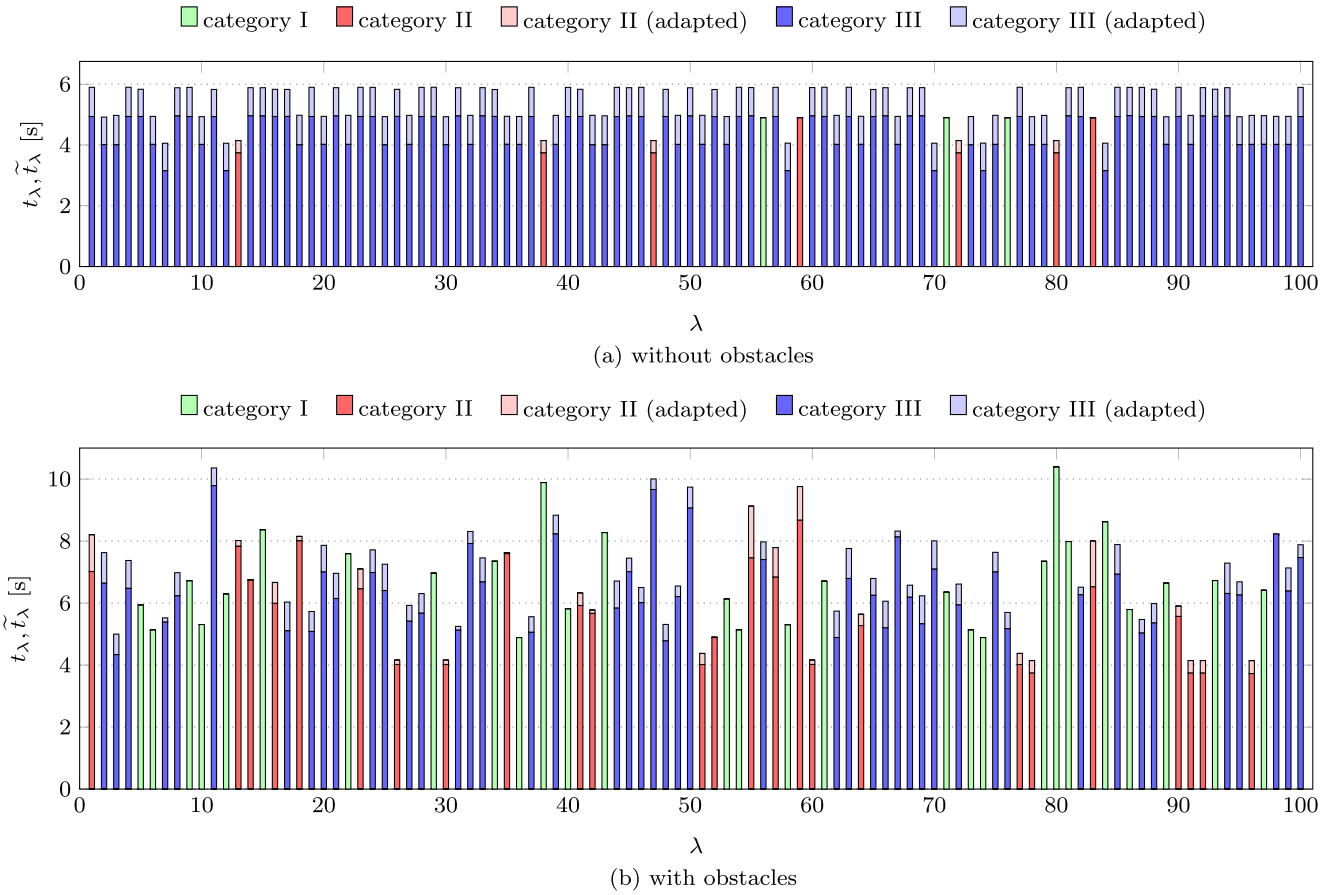


Fig. 15 Cycle times of the planned trajectories $\boldsymbol{T}_\lambda$ or adapted trajectories $\widetilde{\boldsymbol{T}}_\lambda$ with and without obstacles

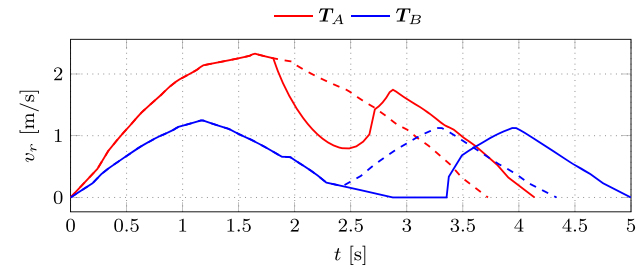
Table 4 Comparing categories A and B regarding the travelled distances and cycle times

Category	A	B
λ	96	3
s_λ	5.33 m	2.97 m
t_λ	3.72 s	4.34 s
$\widetilde{t_\lambda}$	4.14 s	5.00 s
Δt_λ	0.42 s (11.3 %)	0.66 s (15.2 %)

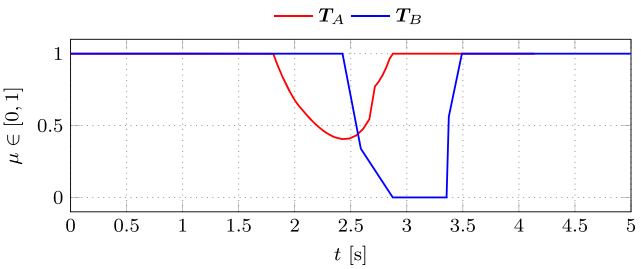
Due to the randomised motion planning algorithm, we obtain a much more differentiated pattern for the cycle times of the 100 trajectories. A total of 28 trajectories are allocated to category I, 25 to category II and 47 to category III. All category B trajectories must complete an almost complete robot stop due to the knee. Nevertheless, only those trajectories are assigned to category III that have to perform a stop due to the speed adaption. The cycle times of all 100 trajectories are given in Fig. 15b.

For both categories, the fastest trajectories are compared in Table 4. It is noticeable that the trajectory $\boldsymbol{T}_A$, before and after the adaption, is much faster than $\boldsymbol{T}_B$, although the distance travelled is much longer. In addition, for $\boldsymbol{T}_A$ only a speed adaption (category II) is required, whereas for $\boldsymbol{T}_B$ a complete robot stop (category III) is necessary.

The characteristics of the robot TCP speeds and velocity scaling factors are shown in Fig. 16. It can be seen that the acceleration at $\boldsymbol{T}_A$ is much higher than at $\boldsymbol{T}_B$. Accordingly, the configuration of the robot in this case must be more



(a) cartesian robot TCP speeds



(b) velocity scaling factors

Fig. 16 Comparing the trajectories of categories A and B for the initial plan (−−) and the adapted execution with regard to robot speeds and velocity scaling factors

suitable, so that this trajectory requires the shorter cycle time for execution, despite temporary adaption. In contrast, it can already be seen in the initially planned trajectory $\boldsymbol{T}_B$ that the speed profile consists of two wave-like motions. Exactly between these waves lies the knee that occurs due to the obstacle. Here, the speed drops to a low value close to zero since the robot has to change its direction abruptly at this knee, so that no smooth motion can be executed. In addition, the current distance is significantly fallen below the separation distance at this time. As a consequence, the robot must perform an adaptive stop for $t_{\text{wait}} = 0.50\,\text{s}$. Afterwards, the robot can move further.

8 Summary and Outlook

In this paper, the main aspects were highlighted that are developed for the planning, analysis and simulation of adaptive safety strategies in the context of SSM. First, based on preliminary work, the extended calculation of the separation distance between the human and the robot was discussed. In order to maintain this separation distance, various strategies were presented, with a particular focus on speed adaption for multiple trajectories. The adaption of the robot speed requires corrections regarding robot positions or time steps at the individual waypoints. These corrections were made by adapting the given time steps of the previously planned trajectory. The adapted time steps were calculated using velocity scaling factors. Since the human motions are coupled to the robot path via the time steps, they must also be adapted. Another important aspect is the robot stop, which occurs when the speed adaption fails. Then, a waiting time was determined until the robot can start moving again.

The described methods for the adaptive motion planning were integrated into the existing HRI simulation tool. Taking into account the required separation distances, a reliable tool for the planning and simulation of HRI scenarios with adaptive motion planning has been developed. This enables the modelling, analysis and simulation of various collaborative production scenarios with different robot systems and human models with regard to the safety requirements in SSM. The simulation tool was then tested in an application example with two different scenarios. For this purpose, a large number of trajectories was generated and compared for a defined task in order to find the best possible robot trajectory.

At first, an application scenario without any obstacles in the collaborative workspace was considered. In this scenario, there were only six different trajectories, which resulted from the six different possible target joint configurations. In this case, all trajectories were PTP trajectories, i.e. generally very fast trajectories, so that in

almost all cases a speed adaption or even a robot stop had to be performed. Furthermore, the cycle times of these trajectories were very similar. In the second scenario, an obstacle was randomly placed in the workspace so that the robot had to follow different trajectories than in the first scenario without obstacle. As a result, 100 different trajectories were generated by the randomised ROS motion planner. These trajectories were partly very slow or far away from the human bodies, so that in many cases no speed adaption and no robot stop was necessary. However, the obstacle has no direct influence on the speed adaption, but it influences the planning of different trajectories and so generates different results in the speed adaption.

Future work will include the consistent extension of the simulation tool. Especially ROS as a manufacturer-independent platform with interfaces to real robot systems has an enormous potential. In addition, the possibility for a real-time adaption is to be developed so that the adapted trajectories can be executed on a robot controller under consideration of real human motions. Even in the simulation itself, the simulation tool still offers many possibilities for enhancement. In this paper, the different robot trajectories were not computed specifically but generated by the randomised motion planning algorithm. In the future, similar to the speed adaption, a goal-oriented algorithm with path adaption will be developed. Especially robot stops could be reduced in this way, as the distance between the human and the robot could be directly influenced. Finally, robot speed and path adaption should be strategically combined in order to always find the time-optimal trajectory for each task and situation.

Author Contributions Paul Glogowski derived the models, performed the calculations, designed and performed the experiments, analysed the data and wrote the manuscript. Alex Böhmer aided in analysing and interpreting the results and worked on the manuscript. Alfred Hypki and Bernd Kuhlenkötter discussed the results, commented on the manuscript, provided critical feedback and helped shape the research, analysis and manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. The research and development project "KoMPI" was funded by the German Federal Ministry of Education and Research (BMBF) within the Framework Concept "Research for Tomorrow's Production" (fund number 02P15A060) and managed by the Project Management Agency Forschungszentrum Karlsruhe, Production and Manufacturing Technologies Division (PTKA-PFT).

Compliance with Ethical Standards

Conflict of Interests The authors declare that they do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Availability of data and material The authors confirm that the data supporting the findings of this study are available within this article.

Code availability not applicable

References

1. ISO/TS 15066: Robots and Robotic Devices – Collaborative Robots (2016)
2. Glogowski, P., Lemmerz, K., Hypki, A., Kuhlenkötter, B.: ROS-based Robot Simulation in Human-Robot Collaboration. In: Karafillidis, A., Weidner, R. (eds.) Developing Support Technologies: Integrating Multiple Perspectives to Create Assistance that People Really Want, pp. 237–246 (2018)
3. Glogowski, P., Lemmerz, K., Schulte, L., Barthelmey, A., Hypki, A., Kuhlenkötter, B., Deuse, J.: Task-based Simulation Tool for Human-Robot Collaboration within Assembly Systems. In: Schöppstuhl, T., Franke, J., Tracht, K. (eds.) Tagungsband des 2. Kongresses Montage Handhabung Industrieroboter, pp. 155–163. Springer, Berlin (2017)
4. Lemmerz, K., Glogowski, P., Hypki, A., Kuhlenkötter, B.: Functional Integration of a Robotics Software Framework into a Human Simulation System. In: 50th International Symposium on Robotics (ISR), Munich (2018)
5. Lacevic, B., Rocco, P.: Kinetostatic Danger Field – a Novel Safety Assessment for Human-Robot Interaction. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 2169–2174 (2010)
6. Lacevic, B., Rocco, P.: Safety-Oriented Control of Robotic Manipulators – a Kinematic Approach. IFAC Proc Vol **44**(1), 11508–11513 (2011)
7. Polverini, M.P., Zanchettin, A.M., Rocco, P.: Real-time Collision Avoidance in Human-Robot Interaction based on Kinetostatic Safety Field. In: IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 4136–4141 (2014)
8. Marvel, J.A.: Performance Metrics of Speed and Separation Monitoring in Shared Workspaces. IEEE Trans. Autom. Sci. Eng. **10**(2), 405–414 (2013)
9. Marvel, J.A., Norcross, R.: Implementing Speed and Separation Monitoring in Collaborative Robot Workcells. Robot. Comput. Integr. Manuf. **44**, 144–155 (2017)
10. Kim, E., Kirschner, R., Yamada, Y., Okamoto, S.: Estimating Probability of Human Hand Intrusion for Speed and Separation Monitoring using Interference Theory. Robot. Comput. Integr. Manuf. **61** (2020)
11. Savur, C., Kumar, S., Arora, S., Hazbar, T., Sahin, F.: HRC-SoS: Human Robot Collaboration Experimentation Platform as System of Systems. arXiv:1905.01026 (2019)
12. Byner, C., Matthias, B., Ding, H.: Dynamic Speed and Separation Monitoring for Collaborative Robot Applications – Concepts and Performance. Robot. Comput. Integr. Manuf. **58**, 239–252 (2019)
13. Lasota, P.A., Rossano, G.F., Shah, J.A.: Toward Safe Close-Proximity Human-Robot Interaction with Standard Industrial Robots. In: IEEE International Conference on Automation Science and Engineering (CASE), pp. 339–344 (2014)

14. Kumar, S., Arora, S., Sahin, F.: Speed and Separation Monitoring using On-Robot Time-of-Flight Laser-ranging Sensor Arrays. In: IEEE International Conference on Automation Science and Engineering (CASE), pp. 1684–1691 (2019)
15. Dröder, K., Bobka, P., Germann, T., Gabriel, F., Dietrich, F.: A Machine Learning-Enhanced Digital Twin Approach for Human-Robot-Collaboration. Procedia CIRP **76**, 187–192 (2018)
16. Glogowski, P., Lemmerz, K., Hypki, A., Kuhlenkötter, B.: Menschzentrierte Simulation mit adaptiver kollisionsfreier Roboterbahnplanung in der Mensch-Roboter-Kollaboration. In: Weidner, R., Karafillidis, A. (eds.) Band zur dritten Transdisziplinären Konferenz "Technische Unterstützungssysteme, die die Menschen wirklich wollen", pp. 47–57 (2018)
17. Schmidt, B., Wang, L.: Contact-less and Programming-less Human-Robot Collaboration. Procedia CIRP **7**, 545–550 (2013)
18. Liu, Z., Wang, X., Cai, Y., Xu, W., Liu, Q., Zhou, Z., Pham, D.T.: Dynamic Risk Assessment and Active Response Strategy for Industrial Human-Robot Collaboration. Comput. Ind. Eng. **141** (2020)
19. Vicentini, F., Giussani, M., Tosatti, L.M.: Trajectory-dependent Safe Distances in Human-Robot Interaction. In: IEEE Emerging Technology and Factory Automation (ETFA), pp. 1–4 (2014)
20. Zanchettin, A.M., Ceriani, N.M., Rocco, P., Ding, H., Matthias, B.: Safety in Human-Robot Collaborative Manufacturing Environments: Metrics and Control. IEEE Trans. Autom. Sci. Eng. **13**(2), 882–893 (2015)
21. Zanchettin, A.M., Rocco, P.: Path-consistent Safety in Mixed Human-Robot Collaborative Manufacturing Environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1131–1136. IEEE (2013)
22. Rosenstrauch, M.J., Pannen, T.J., Krüger, J.: Human Robot Collaboration – Using Kinect v2 for ISO/TS 15066 Speed and Separation Monitoring. Procedia CIRP **76**, 183–186 (2018)
23. Glogowski, P., Lemmerz, K., Hypki, A., Kuhlenkötter, B.: Extended Calculation of the Dynamic Separation Distance for Robot Speed Adaption in the Human-Robot Interaction. In: International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil (2019)
24. ISO 10218-1: Robots and Robotic Devices – Safety Requirements for Industrial Robots – Part 1: Robots (2012)
25. Deuflhard, P.: Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms. Springer Science & Business Media, vol. 35 (2011)
26. Gourdon, X., Sebah, P.: Newton's Method and High Order Iterations (2001)
27. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an Open-Source Robot Operating System. In: ICRA Workshop on Open Source Software (2009)
28. Bauer, S.: Prozesssprachenbasiertes System zur Ansteuerung digitaler Menschmodelle als Teilkomponente einer Software zur Planung und Visualisierung menschlicher Arbeit in der Digitalen Fabrik, Ph.D. Thesis (2015)
29. Fritzsche, L., Jendrusch, R., Leidholdt, W., Bauer, S., Jäckel, T., Pirger, A.: Introducing ema (Editor for Manual Work Activities) – A New Tool for Enhancing Accuracy and Efficiency of Human Simulations in Digital Production Planning. In: International Conference on Digital Human Modeling, pp. 272–281. Springer (2011)
30. Sucan, I.A.: Task and Motion Planning for Mobile Manipulators. Ph.D. Thesis (2012)
31. Sucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robot. Autom. Mag. **19**(4), 72–82 (2012)
32. Kuffner, J.J., LaValle, S.M.: RRT-Connect: An Efficient Approach to Single-Query Path Planning. In: International Conference on Robotics and Automation, vol. 2, pp. 995–1001. IEEE (2000)

Paul Glogowski received his Master of Science in mechanical engineering at the University Duisburg-Essen in 2015. Since 2015 he is working as a research assistant at the Chair of Production Systems (LPS) at the Ruhr University Bochum in the workgroup of industrial robotics. The research in his doctoral thesis focuses on the robot simulation and adaptive motion planning in the human-robot interaction.

Alexander Böhmer received his Bachelor of Science in mechanical engineering at the Ruhr University Bochum in 2019. He is currently completing his Master of Science and is studying physics at the Technical University Dortmund in parallel. His main subjects of study are human-robot interaction and additive manufacturing.

Alfred Hypki received his Diploma of Electrical Engineering at the Ruhr University Bochum in 1989. Until 2009 he was research assistant, head of the Department "Communication Structures and Intelligent Systems" and head of the computer center at the Institute of Robotics Research (IRF) at the Technical University Dortmund, where he also received his doctoral degree. From 2009 to 2012 he was chief engineer at the Chair of Industrial Robotics and Production Automation (IRPA) and from 2012 to 2015 he was chief engineer at the Institute for Production Systems (IPS), both at TU Dortmund. Since 2015 he is chief engineer at the Chair of Production Systems (LPS) at the Ruhr University Bochum. His main research activities are robot simulation and robot programming as well as human-robot collaboration.

Bernd Kuhlenkötter received his doctoral degree in 2001 at the Technical University Dortmund. From 1999–2004 he was department head in "Dortmunder Initiative zur rechnerintegrierten Fertigung (RIF) e.V.". From 2001–2005 he was chief engineer at the Department of Assembly and Handling Systems at TU Dortmund. From 2005–2007 he assumed responsibility for representing the professorship on the subject of "Industrial Robotics and Handling Systems" at the Institute for Robotics Research at TU Dortmund. From 2007–2009 he moved to ABB Automation GmbH as head of product management and technology. His area of responsibility encompassed the development of new robot technologies in cooperation with international ABB development centers. In April 2009 he took over the Professorship for "Industrial Robotics and Production Automation" at the Mechanical Engineering Faculty at TU Dortmund. He founded the Institute for Production Systems (IPS) in Dortmund in 2012 and acted as manager till March 2015. After that he took over the professorship "Production Systems" at the Ruhr University Bochum.