Optimizing Mobility of Robotic Arms in Collision-free Motion Planning

Sascha Kaden¹ · Ulrike Thomas¹

Received: 1 July 2020 / Accepted: 26 April 2021 / Published online: 28 May 2021
© The Author(s) 2021

Abstract

A major task in motion planning is to find paths that have a high ability to react to external influences while ensuring a collision-free operation at any time. This flexibility is even more important in human-robot collaboration since unforeseen events can occur anytime. Such ability can be described as mobility, which is composed of two characteristics. First, the ability to manipulate, and second, the distance to joint limits. This mobility needs to be optimized while generating collision-free motions so that there is always the flexibility of the robot to evade dynamic obstacles in the future execution of generated paths. For this purpose, we present a Rapidly-exploring Random Tree (RRT), which applies additional costs and sampling methods to increase mobility. Additionally, we present two methods for the optimization of a generated path. Our first approach utilizes the built-in capabilities of the RRT*. The second method optimizes the path with the stochastic trajectory optimization for motion planning (STOMP) approach with Gaussian Mixture Models. Moreover, we evaluate the algorithms in complex simulation and real environments and demonstrate an enhancement of mobility.

Keywords Motion planning · Sampling · RRT · Optimization methods · Manipulability · STOMP · Gaussian mixture models

1 Introduction

Motion planning is an essential topic in robotics. The requirements on it are constantly increasing in the last years, as even stronger cooperation between humans and robots is desired. Thus, the industrial workflow will change in the future. In general, the robot has to adapt to humans and not vice versa. In these scenarios, improved movements are required to increase safety instead of simply avoiding obstacles and reducing path lengths in joint space to save energy. An important criterion in motion planning is manipulability and the resulting avoidance of manipulator singularities, where the robot loses its degrees of freedom. Based on the prevention of singularities, it is also possible to enable Cartesian impedance control at any time, which is

a necessary condition for cooperation with humans and to guarantee their safety. Moreover, modern robotic arms with seven degrees of freedom are used for cooperative work because they offer better flexibility due to the redundancy of the seventh joint. However, these are often built like the DLR¹—light-weight arm LWR₁ and have been alternating revolute and torsion joints, which leads to numerous singularities in their working area. Because of these reasons, motion planning with the prevention of singularities is essential. A simple example of a poor configuration is the fully extended manipulator, where a robotic arm loses its degrees of freedom and can only perform very limited movements as shown in Fig. 2.

Yoshikawa [31] first presented the measure of manipulability for a robot. It describes the proximity to singularities with the concept of the manipulability ellipsoid. The measure of manipulability is applied in a variety of fields of robotics. In [28] it is used for manipulability optimization control to enable Robot-Assisted Minimally Invasive Surgery. Other applications include the optimization of redundant manipulators using dynamic neural networks [8],

✉ Sascha Kaden
sascha.kaden@etit.tu-chemnitz.de

Ulrike Thomas
ulrike.thomas@etit.tu-chemnitz.de

¹ Robotics and Human-Machine-Interaction Lab, Chemnitz
University of Technology, Chemnitz, Germany

¹German Aerospace Center

solving the inverse kinematics of redundant robots constructed according to the DLR manipulator scheme [4], or realizing proximity servoing for manipulators with distance sensors [3]. Another field of application is planning of optimal movements for mobile platforms equipped with a manipulator, as shown in [21] and [1].

Sampling-based motion planning was developed for collision-free motion planning of high-dimensional robots, such as redundant manipulators. In this approach, not the entire Cartesian space is mapped into the configuration joint-space, instead only a valid subspace is constructed from random samples, which will be used for the search of a valid path. Since determining the entire configuration space is too complex. An algorithm to perform single queries with sampling is the Rapidly-exploring Random Tree (RRT), which was developed by LaValle & Kuffner [15]. In this approach a tree of random and valid configurations is grown from the start to the target configuration. Over the years this method was adapted and optimized several times. The first extension was the expansion of two trees with the RRT-Connect algorithm [14], beginning at the start and target configuration simultaneously. To solve the query, only a connection between the trees has to be found, which leads to significantly shorter computation times. A second adaptation is the RRT* [11] where local optimization is performed to shorten the path length in joint space. These two changes have been combined in [13] to reduce calculation times and to find optimal paths at the same time. There also exist many other adjustments, such as [22], where an extension for real-time capability is presented, in which the path is reshaped during execution to generate an optimal path. A further adaption is the “Rapidly-exploring Random Vines” [29], where the nodes know whether they are in front of a narrow passage or a passageway. With the help of this information, a quick exploration in a narrow space is possible.

Apart from sampling-based motion planning, other approaches to optimize trajectories have been developed. The most widely known are covariant Hamiltonian optimization for motion planning (CHOMP) [25] and stochastic trajectory optimization for motion planning (STOMP) [10]. All these optimization techniques share the assumption of an initially non-optimized trajectory between the start and end configuration, which will be continuously optimized. The different optimization methods adapt the trajectory in such a way that it complies with constraints such as collision avoidance and joint limitations, while it optimizes simultaneously other parameters such as distance to obstacles or smoothness costs. The CHOMP method applies covariant gradient techniques to adapt the initial discrete trajectory with respect to the optimization and constraint criteria. In contrast, STOMP does not require gradient information; instead, random trajectories are sampled near the initial one.

Based on the costs of these near trajectories, an update of the original motion is performed. CHOMP and STOMP operate iteratively and will be repeated until all criteria are met. In another approach, particle filters are used for the optimization [12]. A further group of trajectory optimization methods are the Gaussian Processes (GP) [19, 20], which have been developed recently. A continuous trajectory is first generated using GP and factor graphs, which is then adapted with a gradient-based optimization algorithm.

In the next section, the calculation of the manipulability and joint limits is presented. This is followed by the motion planning algorithms with optimization of mobility in Section 3. In Section 4 our simulation results are presented, and a conclusion and outlook are offered in Section 5.

2 State Costs of Mobility

We use two additional criteria to optimize motion planning. First, the manipulability λ and second, the distance to the joint limits ψ . In our motion planning algorithms, costs are always normalized and lie between 0 and 1. We guarantee with this that the calculations are general and independent of the applied manipulator. Furthermore, a reasonable scaling can be realized. For subsequent evaluation and visualization, these criteria are defined as qualities and should thus be maximized.

2.1 Manipulability Quality

The manipulability is a predictable and scalar measure of the manipulation ability of a robotic arm [31]. It describes its capability to move within the Cartesian freedoms, in other words, to translate or rotate. We consider a manipulator with n joints or degrees of freedom, which are denoted by $\theta_i, i = 1, \dots, n$. The transformation between Cartesian task space velocities $\dot{x} \in \mathcal{R}^m$ and the joint velocities $\dot{\theta} \in \mathcal{R}^n$ of the manipulator is provided by the robot Jacobian $J(\theta) \in \mathcal{R}^{m \times n}$, where the Jacobian can be interpreted as scaling:

$$\dot{x} = J(\theta)\dot{\theta}. \quad (1)$$

From the Jacobian we can calculate the scalar manipulability measure as:

$$\lambda(\theta) = \sqrt{\det(JJ^T)} = \sigma_1\sigma_2 \cdots \sigma_m. \quad (2)$$

Where $\sigma_1, \sigma_2, \dots, \sigma_m$ are the singular values of the Jacobian and $\lambda(\theta)$ is related to the manipulability ellipsoid with the axes σ_i . If one degree of freedom of the robot is lost, then one of the singular values is equal to zero ($\sigma_i = 0$) and one axis of the ellipsoid has no length. If $m = n$, as it is the case with non-redundant manipulators, the calculation can

be reduced to $\lambda(\theta) = |\det(J(\theta))|$. A further description and calculation of the manipulability is provided in [9].

It has to be noted that the magnitude of the measurement depends on the manipulator, since $J(\theta)$ represents the correlation of the serial chain of the manipulator and its link lengths. To avoid this dependency, we operate with the normalized manipulability in the range of $[0, 1]$. For this reason, we divide the determined value by the highest measured manipulability λ_{max} , which we estimated from our experiments in Section 4:

$$|\hat{\lambda}(\theta)| = \frac{\lambda(\theta)}{\lambda_{max}} \quad (3)$$

In contrast to our former work [9], we pay attention to a numerical problem of the manipulability value. There is a variation of the Jacobian eigenvalues from 1×10^6 to 1×10^{-6} and therefore they can cancel each other out. Thus, an almost complete loss of individual Cartesian degrees of freedom cannot be detected.

$$J_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \rightarrow \sigma_{1,1} = \sigma_{1,2} = 1 \rightarrow \lambda_1 = 1 \quad (4)$$

$$J_2 = \begin{bmatrix} 1 \times 10^{-3} & 0 \\ 0 & 1 \times 10^3 \end{bmatrix} \rightarrow \sigma_{2,1} = 1 \times 10^{-3}, \quad \sigma_{2,2} = 1 \times 10^3 \rightarrow \lambda_2 = 1 \quad (5)$$

In Eqs. 4 and 5 two Jacobians can be seen with their corresponding eigenvalues and manipulability. It is obvious that although both Jacobians have the same manipulability $\lambda_{1,2} = 1$, J_2 has almost lost one degree of freedom. To overcome this problem, $\lambda(\theta)$ is divided by the manipulability condition number $\lambda_{cond}(\theta) = \frac{\sigma_{max}}{\sigma_{min}} \geq 1$. With this ratio, the manipulability is weighted so that the highest eigenvalue is removed and the weight of the smallest eigenvalue is squared. This finally results in the following equation to calculate the manipulability:

$$|\lambda(\theta)| = \frac{\lambda(\theta)}{\lambda_{max} \lambda_{cond}(\theta)} \quad (6)$$

2.2 Joint Limit Quality

To guarantee the mobility of the manipulator, it is necessary to prevent the joints from moving to their limits. For this, we use the joint limit quality. As a basis of our method, we took the *distance from mechanical joint limits* function from [26]. However, this method, as well as other presented methods come with the shortcoming of not punishing sufficiently isolated joints at their boundaries. Therefore, we developed a method with exponential costs. The calculation of the joint angle costs is done in several steps. The first step is to

calculate the normalized distance from the center of the joint angle range:

$$\hat{\psi}_i(\theta) = 2 \frac{\theta_i - \theta_{center,i}}{\theta_{range,i}} \quad (7)$$

Where θ_i is the i th joint value and $\theta_{center,i}$ the corresponding joint center and $\theta_{range,i}$ the range. Afterward the distances $\hat{\psi}_i$ are adjusted through an exponentially growing function, so that the costs are increased when positions close to the joint limits are reached:

$$\psi'_i = 2.9641 \hat{\psi}_i^5 + 0.0359 \hat{\psi}_i. \quad (8)$$

The weights of the quintic function have been chosen to ensure that the joint costs reach a value of 1 for 80 % displacement from the center of the joint. A further displacement is additionally rated, since in this area there may be a strong restriction of the robot's mobility. The progression of the costs is shown in Fig. 1. From these individual joint costs, the mean value $\bar{\psi} = \frac{1}{n} \sum_{i=1}^n \psi'_i$ is then calculated and a final conversion to joint quality and a case differentiation is made to ensure a minimum quality of zero:

$$\psi = \begin{cases} 1 - \bar{\psi} & \text{if } \bar{\psi} > 1 \\ 0 & \text{else} \end{cases} \quad (9)$$

The purpose of this weighting is to ensure that an above-average cost is assigned when a single joint approaches its limits. This is important since it is already sufficient for the motion restriction if one joint reaches its end position. Therefore, a quality value of 0 does not imply that a manipulator has reached its joint limits with all joints.

3 Motion Planning

We separate the motion planning into two parts. First, we use sampling-based motion planning to ensure that a valid path is found, even if narrow passages have to be traversed. In the second step, we use two optimization strategies to smooth the path and to achieve further optimization of the mobility. However, the problem with optimization strategies is that they get stuck in narrow passages or local minima [17]. With our combination, we achieve the ability to find paths in complex environments and additionally optimize the path according to manipulability and joint limits.

3.1 Sampling-based Motion Planning

The here developed sampling-based motion planner is based on the RRT*-Connect [13] and its extension the RRT*-Quick method [7], to achieve a faster convergence rate. First, we extended these RRT-algorithms with state and distance costs to achieve additional optimization. The

Fig. 1 Illustration of the polynomial in Eq. 8 to calculate the cost of a joint

enhancement of the criteria is achieved by the internal local optimization of the RRT*. Secondly, we developed two new sampling algorithms for the optimization of manipulability. The main process of our RRT is in Algorithm 1 illustrated. Our RRT algorithm is designed for parallel computation to achieve fast and efficient calculations. This means that all modules like graph (tree), collision detection, sampling, trajectory calculation, or distance and state cost calculation of the motion planner are thread-independent and can therefore run in parallel. Because of this structure, we grow a tree always in parallel by several execution threads, since a parallel operation takes advantage of the local optimization of the RRT*. Moreover, we simultaneously expand from the start and target configuration for k_{nodes} and run both threads in completion. We justify this additional separation and simultaneous calculation from the publication [6], which shows a limit on the parallel speedup for the expansion speed of the tree. Therefore, if enough processor cores are available, it is recommended to expand these two trees concurrently. Afterward, we test for a connection between both trees. If this is possible, a valid path was found and as the last step, we run a path optimization, which is described in Section 3.2. Otherwise, we expand again or abort the planning if a maximum number of attempts is exceeded.

The $EXTENDTHREAD(T, k_{nodes})$ function performs the expansion of the passed tree T , where it calls $k_{nodes} \times EXTEND(T)$. This method then applies for the extension of the RRT algorithm to a new configuration, as illustrated in Algorithm 2. For the generation of samples, we use one of our two developed sampling strategies. For both, we take a random configuration in the first step, where we set a goal bias of 5 %. Afterward, one of our developed sampling strategies either Manipulability bias or Manipulability

Algorithm 1 RRT*-Quick-Connect.

```

1: function RRT*-CONNECT( $\theta_{start}, \theta_{goal}, k_{nodes}$ )
2:    $T_{start}.INIT(\theta_{start})$ 
3:    $T_{goal}.INIT(\theta_{goal})$ 
4:   for  $i \leftarrow 1$  to attempts do
5:      $EXTENDTHREAD(T_{start}, k_{nodes})$ 
6:      $EXTENDTHREAD(T_{goal}, k_{nodes})$ 
7:      $JOIN(T_{start}, T_{goal})$ 
8:     if  $CONNECT(T_{start}, T_{goal})$  then
9:        $OPTIMIZE(T_{start}, PATH(T_{start}))$ 
10:      return  $PATH(T_{start})$ 
11:  return failure

```

Random Gradient Descent (RGD) is used to evaluate or adjust the samples. Both sampling strategies are illustrated in Algorithm 3.

3.1.1 Manipulability Bias

This method is inspired by the manipulability bias of [16], which is constructed for the sampling algorithm: probabilistic roadmap (PRM). However, we do not think their concept is suitable for the RRT approach since samples have to be generated for the entire configuration space like it's the case with PRM. For this reason, we developed a new method to suit the RRT. As first, we define a pseudo probability value ρ that determines the usage of a sample:

$$\rho = |\lambda(\theta)| + \delta_{bias} \quad (10)$$

where δ_{bias} is a user defined threshold value. ρ results from the manipulability value of a configuration and δ_{bias} . We apply an additional δ_{bias} to allow sampling of

configurations with low manipulability. For example, the robot may have to pass through a narrow passage, where only a limited manipulability can be achieved, see Fig. 2.

3.1.2 Manipulability RGD

As a second method, we use the manipulability RGD, which shifts the random configuration, such that the manipulability increases. This algorithm is inspired by the RGD algorithm from [27], although their approach is designed for Cartesian constraints. Initially, a random direction vector θ_{ray} with a small length is first generated. A new configuration is then created with: $\theta_{new} = \theta_{rand} + \theta_{ray}$. When the manipulability increases ($|\lambda(\theta_{new})| > |\lambda(\theta_{rand})|$), a continuing translation into this direction is performed. This is repeated until the maximum displacement δ_{RGD} is reached, or no more enhancement of λ occurs. To set an adaptive value for the maximum displacement, we link it to the step size of the RRT. As a standard case, we apply 50 %.

3.1.3 State and Distance Costs

The next steps are: Cost calculation and linking of θ_{rand} to the tree. From sample θ_{rand} , the closest neighbor in the tree T is searched. The distance criterion in the search is Euclidean distance in joint space and not the introduced cost value of the nodes, as these costs are distinct. Then the new configuration θ_{new} for the linkage is determined using the STEER algorithm of the RRT. Based on θ_{new} a range search within the step size is performed to determine the near nodes Θ_{near} . These nodes are then extended by their ancestor nodes. The degree of the ancestors m is a fixed value and

Fig. 2 Sample environment where the robot needs to move close to a singularity to change its position from left to right side of the obstacle

Algorithm 2 Extend.

```

1: function EXTEND( $T = (V, E)$ )
2:    $\theta_{rand} \leftarrow \text{GETSAMPLE}()$ 
3:    $\theta_{nearest} \leftarrow \text{NEAREST}(T, \theta_{rand})$ 
4:    $\theta_{new} \leftarrow \text{STEER}(\theta_{rand}, \theta_{nearest})$ 
5:   if INCOLLISION( $\theta_{new}$ ) then
6:     return false
7:   // Connect to node with minimum cost.
8:    $\Theta_{near} \leftarrow \text{NEAR}(T, \theta_{new})$ 
9:    $\Theta_{near}.\text{INSERT}(\text{ANCESTORS}(\Theta_{near}, m))$ 
10:  SORTBYCOSTS( $\Theta_{near}$ )
11:  for each  $\theta_{near} \in \Theta_{near}$  do
12:    if TRAJECTORYCOLFREE( $\theta_{new}, \theta_{near}$ ) then
13:       $\theta_{new}.\text{cost} = \text{CALCCOST}(\theta_{new}, \theta_{near})$ 
14:       $T.\text{ADDNODE}(\theta_{new});$ 
15:       $T.\text{ADDEDGE}(\theta_{near}, \theta_{new});$ 
16:      BREAK
17:  if valid connection found then
18:     $\text{REWIRE}(\theta_{new}, \Theta_{near} \setminus \theta_{near})$ 
19:  return true

```

is set to three by default, as it is in the original publication [7]. In addition to [7], we sort the nodes by their cost to get the best possible connection and thus save computing time. Therefore, the first valid linkage is also the best possible one.

If a valid connection was found, a cost calculation of θ_{new} will be performed, which represents another important new aspect in our RRT algorithm. State costs are calculated in addition to the distance costs and assigned to each tree node during the expansion phase using a cost function, see Algorithm 4. We interpret the state costs of the RRT as a

Algorithm 3 Sampling.

```

1: function GETSAMPLE()
2:    $\theta_{rand} \leftarrow \text{RANDOMCONFIG}()$ 
3:   if useManipulabilityBias then
4:      $\rho = \lambda(\theta) + \delta_{bias}$ 
5:     if  $\rho > \text{rand}([0, 1])$  then
6:       return  $\theta_{rand}$ 
7:     else
8:       return  $\emptyset$ 
9:   else
10:    for  $i \leftarrow 1$  to attempts do
11:       $\theta_{diff} = \theta_{ray} \leftarrow \text{RANDOMRAY}()$ 
12:      if  $\lambda(\theta_{rand} + \theta_{diff}) > \lambda(\theta_{rand})$  then
13:        while  $\lambda(\theta_{rand} + \theta_{diff})$  increase and
14:           $|\theta_{diff}| < \delta_{RGD}$  do
15:             $\theta_{diff} += \theta_{ray}$ 
16:        return  $\theta_{rand} + \theta_{diff}$ 

```

Algorithm 4 calcCost.

```

1: function CALCCOST( $\theta_{new}, \theta_{near}$ )
2:    $c(\theta_{new}) \leftarrow \alpha_\psi \psi(\theta_{new}) + \alpha_\lambda (1 - |\lambda(\theta_{new})|)$ 
3:    $c_{dist} \leftarrow \text{EUCDIST}(\theta_{new}, \theta_{near}) + \text{MANIPULABILITYDIST}(\theta_{new}, \theta_{near})$ 
4:   return  $c(\theta_{new}) + c_{dist} + \theta_{near}.\text{COST}()$ 

```

summation of different costs, such as joint limits $c_\psi(\theta) = 1 - \psi(\theta)$, distance to obstacles $c_o(\theta)$ or manipulability $c_\lambda(\theta) = 1 - |\lambda(\theta)|$, as shown in Eq. 11. The additional conversion of manipulability and joint limitation costs is based on the transformation of qualities into costs that can be used in motion planning.

$$c(\theta) = \alpha_\psi c_\psi(\theta) + \alpha_d c_d(\theta) + \alpha_\lambda c_\lambda(\theta) \quad (11)$$

In this publication we use $c_\lambda(\theta)$ and $c_\psi(\theta)$ as state costs to optimize the path. After the calculation of the states, the distance costs are determined. Here we again combine two kinds of costs, the usual Euclidean distance in the joint space of the standard RRT* and the manipulability as distance costs. For this purpose, the manipulability is summed in 0.5° increments along the path. The final cost of a node is then calculated from the distance, state, and previous cost of the parent node.

After the cost calculation, the new node and its edge are added to the tree. Finally, the rewiring algorithm of the RRT* [11] is applied, whereby the ancestor nodes are also tested and connected if it cause less cost over the new node. Furthermore, collision checking is always additionally performed to ensure a non-colliding robot. Collision calculations are carried out with the flexible collision library (fcl) [23].

By extension of the state and distance costs, we reach an additional optimization of the RRT algorithm. There are no limitations to types and combinations of different costs. In Section 4 we present the results of the sampling-based motion planning with manipulability and joint limits as state costs and the developed sampling strategies.

3.2 Optimization Methods

After a valid path has been found, path optimization is performed to increase the manipulability. We developed two methods to optimize the path. First, an optimization using the ability of the RRT* algorithm of further expansion of the tree, we name it: the RRT* manipulability expansion. Secondly, we developed a stochastic optimization consisting of a combination of STOMP [10] and GMMs [18], which are inspired by the Gaussian Process Motion Planning (GPMP) [20]. These two methods run in parallel to achieve

short computation times. In Section 4.1 we discuss their results and whether their application or a combination of them is rewarded.

3.2.1 RRT* Manipulability Expansion

For the enhancement of the path, the local optimization capability of the RRT* is used. We take advantage of this by letting the tree grow locally in the vicinity of the found path. Here we do not consider the often mentioned Informed-RRT* [5], as we discovered in our tests that the hyperspheroid often already occupies the whole joint space of a manipulator, since this space is already very limited through the joint limits. The first step is to find all nodes that are located in a radius of r_{path} to the nodes of the previously calculated goal path. By default, we set r_{path} to twice the width of the RRT* step size. In the second step, potential configurations must be generated to extend the tree. For this, we use a spherical Poisson sampler, which is inspired by [2] and [24]. It generates uniformly distributed samples Θ_{sphere} within a sphere based on Poisson disk sampling. Two parameters the sphere radius r_s and the disc radius r_d are needed. For this purpose, we first perform an oversampling within r_s and then delete unnecessary points using r_d . An example sphere and its samples are shown in Fig. 3. For each configuration from the set Θ_{sphere} the manipulability is calculated and Θ_{sphere} sorted accordingly. With the u best configurations, the RRT is then extended and the path is optimized. The process is illustrated in Algorithm 5 and as input the following data is required:

- Θ_{path} , the nodes of the determined path,
- T the tree to be expanded and
- u the number of extensions of a node.

Algorithm 5 Optimize RRT.

```

1: function OPTIMIZERRT( $\Theta_{path}, T, u$ )
2:    $\Theta_{near} \leftarrow \text{RANGESEARCH}(\Theta_{path}, T, range)$ 
3:   for each  $\theta_{near}$  in  $\Theta_{near}$  until  $n_{Nodes}$  do
4:      $\Theta_{sphere}.\text{SPHERESAMPLER}(\theta_{near}, u)$ 
5:      $\text{SORTBYMANIPULABILITY}(\Theta_{sphere})$ 
6:      $\text{RESIZE}(n_{Nodes})$ 
7:     for each  $\theta_{sphere}$  in  $\Theta_{near}$  until  $n_{Nodes}$  do
8:        $\text{EXTEND}(\theta_{sphere})$ 

```

3.2.2 Stochastic Optimization

Furthermore, we make use of a stochastic optimization, which is based on the STOMP algorithm [10], but has two major differences to it. First, unlike the original, we do not

Fig. 3 Examples of uniformly distributed Poisson disk samples within a unit sphere in the range of $[-1, 1]$ and a disk radius of 0.2

use a linear connection as the initial path, instead, we take the optimized and collision-free path from the sampling-based motion planner. As a result, this optimization always runs at the end, even if an RRT* manipulability expansion takes place. Second, we do not generate the random trajectories of the stochastic optimization with a uniform distribution like STOMP, instead, we apply a GMM on the input trajectory to generate our trajectories. Hereby, it is possible to set a variable covariance over the individual trajectory steps. Which is to calculate the covariances of the GMM as a function of the individual trajectory step costs. Using the GMM, we create our K random trajectories to optimize the path.

The optimization consist of following steps. First, the initial trajectory is re-interpolated so that there are sufficient grid points. Then K random trajectories are calculated from the GMM for the statistical adjustment. For each grid point, the costs resulting from Eq. 12 are calculated.

$$s(\theta) = \alpha_s (c_o(\theta) + c_\lambda(\theta) + c_\psi(\theta)) \quad (12)$$

These consist of distance ($c_o(\theta)$), manipulability ($c_\lambda(\theta)$) and joint ($c_\psi(\theta)$) costs. Finally, the path is optimized based on the cost of all random trajectories. This optimization is described in detail in [9], although no joint costs are used in our previous version.

Fig. 4 Simulation in a cluttered environment to evaluate the manipulability optimization. The motion starts at the back (orange) and ends at the front pose (green)

Fig. 5 Second test case to demonstrate that paths can be found in a narrow passage. The motion starts at the back (red) and ends at the front pose (green) and has to be performed under the archway

In the standard case, we repeat this calculation twice to obtain an optimized trajectory concerning the costs from Eq. 12 and smoothness of the path.

4 Results and Discussion

We evaluated our algorithms in different simulation environments with the iiwa manipulator from KUKA. First, we selected a cluttered environment to compare the different algorithms of the motion planner. This environment is shown in Fig. 4 and represents a human-robot cooperation cell (HRC), where the robot is attached to the ceiling to provide the option of multi-robot coordination. First, individual sub-algorithms are evaluated to optimize the manipulability. In the second part, the final methods are

tested in further simulation and real scenarios, see the scenarios in Figs. 5 and 6. And thirdly, we perform our motion planning in a real environment. For all evaluations, we use a workstation with two Intel Xeon E5-2670 v3 CPUs, each with 12 processor cores and 64 GB Ram. All tests run with 24 threads. For the implementation, we use C++ as programming language.

4.1 Evaluation of Sub-algorithms

To compare the individual algorithms for an optimization of the manipulability, we use the simulation environment in Fig. 4. All tests are repeated 15 times, since the calculation with the RRT is not deterministic. First, we perform a motion planning without optimization of manipulability. The results are shown in Fig. 7, in (a) the computation times are shown as box plot, in (b) is the progression of the manipulability of all 15 paths illustrated and in (c) is the course of the joint qualities of the paths illustrated. It can be seen that λ becomes zero several times and therefore a loss of degrees of freedom of the manipulator occurs (Fig. 8).

Next, the different algorithms to enhance the manipulability are compared. The following scales of the cost calculation are defined: $c_\psi = 1$, $c_\lambda = 5$ and both distance costs have a scaling of 1. In addition, the step size of the RRT* is defined to be 60° .

First, we compare the two sampling algorithms: Manipulability bias and Manipulability RGD. A threshold $\delta_{bias} = 0.2$ is set for the bias method. For the RGD, the maximum displacement is fixed at 50 % of the step size of the RRT*, which corresponds to a value of $\delta_{RGD} = 30^\circ$. The results of the bias sampling are shown in Figs. 9 and in 10 the outcome of the RGD sampling is illustrated. In this context, it can be seen that the bias sampling achieves slightly better results, which is manifested by lower fluctuations and fewer zero values of the progressions of manipulability. In terms of computational time, both algorithms with manipulability costs take twice as long to compute to the non-optimized version, although this is somewhat greater with RGD. This results from the fact that more expensive sampling algorithms and complex calculation of the state and path costs are used because the number of required collision calculations remains constant.

Fig. 6 Simulation environment with the iiwa manipulator to evaluate the manipulability maximization. Our tested motion contains three poses, it starts in **a**, moves via the defined intermediate pose **b** and finishes in **c**

Fig. 7 Simulation results of the HRC scenario in Fig. 4 with a simple sampling and without state costs and only Euclidean distance costs in the joint space. **a** shows the total calculation time, **b** and **c** present the progression of manipulability and joint limits of all 15 paths

In the second part, we compare bias and RGD sampling in a subsequent improvement with the RRT* manipulability expansion. For the sphere sampler the radii are set to $r_{sphere} = 40^\circ$ and $r_{disk} = 20^\circ$. This results in about 700 equally distributed samples within the sphere. Furthermore, we set $u = 3$ and expand the tree with the three best samples according to the state costs from line 2 of Algorithm 4. The results of optimization with a previous bias sampling are shown in Fig. 11 and with RGD sampling in Fig. 12. The additional optimization of the RRT* leads to a major increase in manipulability, with no zero crossings occurring when bias sampling is used. When comparing the two sampling methods, it is also clearly evident that bias sampling delivers better results, which is due to better manipulability values. The slightly longer computing times of the bias sampling are a result of the increased number of required collision calculations.

In the second last test, we compare the two sampling methods with a subsequent improvement with the stochastic optimization. This optimization performs two iterations. The scaling is chosen as in the paper [9] with the values: $\alpha_s = 0.04$, $\alpha_d = 10$ and $\sigma_d = 2.5$. The scaling of the state

Fig. 8 Mean values of manipulability through 50 runs from the evaluation of the sub-algorithms: **a** without optimization, **b** bias sampling, **c** bias and RRT* opt., **d** bias and stochastic opt., **e** bias and both opt., **f** RGD sampling, **g** RGD and RRT* opt., **h** RGD and stochastic opt

costs is the same as in the RRT*. This optimization leads again to a major increase in manipulability. Thereby bias sampling again offers better results (Figs. 13, 14 and 15).

The final test is a motion planning with bias sampling, as well as RRT* expansion and stochastic optimization. The results are shown in Fig. 16. It can be seen that optimizing with both algorithms leads to a further enhancement of the manipulability, while also requiring longer computing times.

In addition, 50 runs were performed for all methods to determine the average manipulability values. The results are shown in Fig. 8. Here it can also be seen that the developed methods result in a clear increase in manipulability. It is also evident that bias sampling regardless of the optimization method leads to better results than RGD sampling. Moreover, it can be seen that the optimization methods produce an improvement, with similar results. And lastly, performing both optimizations leads to a further increase of manipulability.

From these results, we conclude first that bias sampling should be used, as it provides better results than RGD sampling, no matter what optimization is done afterward. From the last test, we can see that optimizing with both methods leads partly to better values of manipulability. The question of which optimization should be preferred, cannot be answered finally using the environment from the scenario in Fig. 4. Because of this, both optimizations and their combination are tested in the next section.

4.2 Evaluation of Further Environments

Finally, we evaluate two more scenarios to demonstrate the capabilities of the motion planner. We prove with these environments that a valid path is always found, and that the manipulability is maximized whenever possible. For each scenario, four cases are calculated, firstly without optimized manipulability, secondly with bias sampling and RRT* expansion, thirdly bias sampling and

Fig. 9 Simulation results of the HRC environment in Fig. 4 with manipulability bias sampling. **a** shows the total calculation time, **b** and **c** present the progression of manipulability and joint limits of all 15 paths

Fig. 10 Simulation results of the HRC environment in Fig. 4 with manipulability RGD sampling. **a** shows the calculation time, **b** the manipulability, **d** the joint limits of all 15 paths

Fig. 11 Simulation results of the HRC environment in Fig. 4 with manipulability bias sampling and RRT* optimization. **a** shows the calculation time, **b** the manipulability, **d** the joint limits of all 15 paths

Fig. 12 Simulation results of the HRC environment in Fig. 4 with manipulability RGD sampling and RRT* optimization. **a** shows the total calculation time, **b** and **c** present the progression of manipulability and joint limits of all 15 paths

Fig. 13 Simulation results of the HRC environment in Fig. 4 with manipulability bias sampling and stochastic optimization. **a** shows the calculation time, **b** the manipulability, **d** the joint limits of all 15 paths

Fig. 14 Simulation results of the HRC environment in Fig. 4 with manipulability RGD sampling and stochastic optimization. **a** shows the calculation time, **b** the manipulability, **d** the joint limits of all 15 paths

Fig. 15 Simulation results of the HRC environment in Fig. 4 with manipulability bias sampling and RRT* and stochastic optimization. **a** shows the calculation time, **b** the manipulability, **d** the joint limits of all 15 paths

Fig. 16 Simulation results of the HRC environment in Fig. 4 with manipulability bias sampling and RRT* and stochastic optimization. **a** shows the calculation time, **b** the manipulability, **d** the joint limits of all 15 paths

Fig. 17 Mean values of the manipulability over 50 runs of the further environments: **a** archway without optimization, **b** archway with RRT* opt., **c** archway with stochastic opt., **d** archway with both opt., **e** table without optimization, **f** table with RRT* opt., **g** table with stochastic opt., **h** table with both opt

Fig. 18 Manipulability course of all 15 paths from simulation in Fig. 6, in **a** without Manipulability, in **b** bias sampling and RRT* expansion, in **c** bias sampling and stochastic optimization and in **d** bias sampling and both optimization methods

Fig. 19 Manipulability course of all 15 paths from simulation in Fig. 5, in **a** without Manipulability, in **b** bias sampling and RRT* expansion, in **c** bias sampling and stochastic optimization and in **d** bias sampling and both optimization methods

Fig. 20 Derivation of one joint path from the simulation Fig. 6 in **a** a computation without Manipulability, in **b** with bias sampling and additional RRT* expansion and in **c** with bias sampling and stochastic optimization

stochastic optimization, and fourthly bias sampling and both optimizations. Again, 15 runs are calculated in each case. The joint costs of both scenarios are never close to zero and therefore not presented (Fig. 17).

The first scene with manipulator mounted on a table, shown in Figs. 6 and 18, illustrates the corresponding manipulability of the determined paths. Both optimizations provide an increase in manipulability, but this is significantly more distinct in the RRT* expansion. The last and most complex scene is shown in Fig. 5, where the manipulator has to pass through an archway and therefore a very limited joint space exists. We picked this scenario to show that in difficult environments, like narrow passages, a collision-free path is still found and that the enhancement of the mobility is only an optimization criterion and does not block the path search. In Fig. 19 the diagrams of the manipulability progression of all 15 paths are shown. In addition, the courses of the derived joint values are plotted in Fig. 20. In all cases, a valid path was found. Due to the narrow passage of the scenario, the manipulability was optimized only to a small extent. Furthermore, the derivation of the joint values is another feature of our stochastic optimization, which reduces the amplitudes and therefore smooths the contours of the path in joint space. Smoothing occurs in all stochastic optimizations and was therefore shown only once. The results of the average values of the manipulability are shown in Fig. 17. It can be seen that the RRT* optimization leads to a much better enhancement compared to the stochastic optimization. Furthermore, the combination of both optimizations does not lead in these tests to any further improvement of the manipulability. Therefore, we conclude that the choice of the optimization method depends on application and environment of the manipulator.

4.3 Execution in Real Environment

Finally, we tested the motion planning on a real iiwa manipulator. The structure of the workcell and a sequence

of the robot motion is shown in Fig. 21. For this test, we performed a simple path consisting of three sections, where we used manipulability bias sampling and RRT* optimization. During the execution of the motions, we performed an impedance control in Cartesian space and applied external influences on the manipulator by a human user. We see this realization as proof of maximizing the manipulability. Since a complete loss of manipulability would lead to a singularity, which in turn would lead to an unwanted (infinite) acceleration of the manipulator. Figure 21 shows a sequence of the motions of the robot. Furthermore, it can be seen that the manipulator always tries to use nearly rectangular joint configurations, which is an indication of maximizing the manipulability. The entire video of the motion can be accessed on the journal's website.

5 Conclusion and Outlook

In this paper, we presented various methods to maximize the mobility of a robot, while still providing a collision-free operation. For us, mobility consists of manipulability and joint limitation costs. These criteria are integrated into a sampling-based motion planner, which consists of an RRT method and optimization algorithms. They combine state and distance costs to optimize the mobility of a robot arm. In the experiments, we proved that we can find paths in complex environments and simultaneously achieve an improvement in mobility. We see this optimized motion planning as a previous step for dynamic planning. We generate paths that have a high degree of freedom to react to unplanned or dynamic obstacles later during the execution. Because dynamic motion planning can only react to unplanned obstacles if the initial path allows freedom to change. In this way, we want to increase safety and flexibility in human-robot cooperation. The proposed algorithm is not specialized and can be applied to all types

Fig. 21 Motion sequence of the top-mounted iiwa manipulator with maximized manipulability and external disturbances

of robot manipulators. To reduce the computing times, it shall be investigated to what extent the use of a look-up table of the manipulability is reasonable, as it is partially used in [30]. This involves analyzing what resolution of this table is required and how fast data access and processing is possible since this database will probably be several gigabytes in size. Furthermore, an investigation of methods to shorten the path length while retaining and increasing the manipulability values is needed. Moreover, it should be investigated to what extent a separation of the Jacobian into the orientation and position part is reasonable. For this purpose, suitable applications should be tested in which a preference for orientation or position exists.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s10846-021-01407-0>.

Author Contributions All authors contributed to the study conception and design. Data collection and analysis were performed by Sascha Kaden. The first draft of the manuscript was written by Sascha Kaden and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. No funding was received for conducting this study.

Availability of Data and Material The raw evaluation data can be viewed at: https://figshare.com/projects/Journal_Paper_Optimizing_Mobility_of_Robotic_Arms_in_Collision-free_Motion_Planning/94454.

Code Availability Custom code of the chair.

Declarations

Conflict of Interests The authors have no relevant financial or non-financial interests to disclose.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Chen, M., Zalzal, A.M.S.: A genetic approach to motion planning of redundant mobile manipulator systems considering safety and configuration. *J. Robot. Syst.* **14**(7), 529–544 (1997)
- Corsini, M., Cignoni, P., Scopigno, R.: Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. Visual. Comput. Graph.* **18**(6), 914–924 (2012)
- Ding, Y., Wilhelm, F., Faulhammer, L., Thomas, U.: Proximity servoing methods for safe human-robot-interaction. In: 2019 IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS). IEEE (2019)
- Faria, C., Ferreira, F., Erhagen, W., Monteiro, S., Bicho, E.: Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance. *Mech. Mach. Theory* **121**, 317–334 (2018)
- Gammell, J.D., Srinivasa, S.S., Barfoot, T.D.: Informed Rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2997–3004. IEEE (2014)
- Ichnowski, J., Alterovitz, R.: Parallel sampling-based motion planning with superlinear speedup. In: IROS, pp. 1206–1212 (2012)
- Jeong, I.B., Lee, S.J., Kim, J.H.: Rrt*-Quick: A motion planning algorithm with faster convergence rate. In: Robot Intelligence Technology and Applications 3, pp. 67–76. Springer (2015)
- Jin, L., Li, S., La, H.M., Luo, X.: Manipulability optimization of redundant manipulators using dynamic neural networks. *IEEE Trans. Ind. Electron.* **64**(6), 4710–4720 (2017)
- Kaden, S., Thomas, U.: Maximizing Robot Manipulability along Paths in Collision-Free Motion Planning. In: 2019 19Th International Conference on Advanced Robotics (ICAR), pp. 105–110. IEEE (2019)
- Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., Schaal, S.: STOMP: Stochastic trajectory optimization for motion planning. In: 2011 IEEE International Conference on Robotics and Automation, pp. 4569–4574 (2011)
- Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **30**(7), 846–894 (2011)
- Kim, J.J., Lee, J.J.: Trajectory optimization with particle swarm optimization for manipulator motion planning. *IEEE Trans. Indust. Inform.* **11**(3), 620–631 (2015)
- Klemm, S., Oberländer, J., Hermann, A., Roennau, A., Schamm, T., Zollner, J.M., Dillmann, R.: RRT*-Connect: Faster, asymptotically optimal motion planning. In: 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1670–1677. IEEE (2015)
- Kuffner, J.J. Jr., LaValle, S.M.: RRT-Connect: An efficient approach to single-query path planning. In: ICRA, vol. 2 (2000)
- LaValle, S.M., Kuffner, J.J. Jr.: Rapidly-exploring random trees: Progress and prospects (2000)
- Leven, P., Hutchinson, S.: Using manipulability to bias sampling during the construction of probabilistic roadmaps. *IEEE Trans. Robot. Autom.* **19**(6), 1020–1026 (2003)
- Luo, J., Hauser, K.: An empirical study of optimal motion planning. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1761–1768. IEEE (2014)
- McLachlan, G., Peel, D.: Finite mixture models. Wiley, New York (2004)
- Mukadam, M., Dong, J., Yan, X., Dellaert, F., Boots, B.: Continuous-time Gaussian process motion planning via probabilistic inference. *Int. J. Robot. Res.* **37**(11), 1319–1340 (2018)
- Mukadam, M., Yan, X., Boots, B.: Gaussian Process Motion Planning. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), vol. 2016-June, pp. 9–15. IEEE (2016)
- Nagatani, K., Hirayama, T., Gofuku, A., Tanaka, Y.: Motion planning for mobile manipulator with keeping manipulability. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 1663–1668. IEEE (2002)
- Otte, M., Frazzoli, E.: RRTX: Asymptotically Optimal single-query sampling-based motion planning with quick replanning. *Int. J. Robot. Res.* **35**(7), 797–822 (2016)

23. Pan, J., Chitta, S., Manocha, D.: FCL: A general purpose library for collision and proximity queries. In: 2012 IEEE International Conference on Robotics and Automation, pp. 3859–3866. IEEE (2012)
24. Park, C., Pan, J., Manocha, D.: Poisson-Rrt. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 4667–4673. IEEE (2014)
25. Ratliff, N., Zucker, M., Bagnell, J.A., Srinivasa, S.: CHOMP: Gradient optimization techniques for efficient motion planning, pp. 489–494 (2009)
26. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics: Modelling, planning and control. Springer Science & Business Media, New York (2010)
27. Stilman, M.: Task constrained motion planning in robot joint space. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3074–3081. IEEE (2007)
28. Su, H., Li, S., Manivannan, J., Bascetta, L., Ferrigno, G., De Momi, E.: Manipulability Optimization Control of a Serial Redundant Robot for Robot-Assisted Minimally Invasive Surgery. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 1323–1328. IEEE (2019)
29. Tahirovic, A., Ferizbegovic, M.: Rapidly-exploring random vines (RRV) for motion planning in configuration spaces with narrow passages. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 7055–7062 (2018)
30. Vahrenkamp, N., Asfour, T.: Representing the robot's workspace through constrained manipulability analysis. *Auton. Robot.* **38**(1), 17–30 (2015)
31. Yoshikawa, T.: Manipulability of robotic mechanisms. *Int. J. Robot. Res.* **4**(2), 3–9 (1985)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Sascha Kaden received the B.Sc. degree in electrical engineering and the master's degree in automation systems from the Chemnitz University of Technology, Chemnitz, Germany, in 2015 and 2018, respectively. He is currently working as a researcher with the Chair of Robotics and Human-Machine-Interaction Lab, Chemnitz University of Technology. His research interests include the sampling-based and optimized motion planning as well as assembly sequencing.

Ulrike Thomas received the Diploma degree in computer science in 2000 and the Ph.D. degree in 2008 in robotics both from the Institute of Robotics and Process Control, Technical University of Braunschweig, Braunschweig, Germany. From 2007 to 2015, she was with the Robotics and Mechatronics Institute, German Aerospace Center, Weßling, Germany. Since 2015, she has been a Professor and Lead of the Robotics and Human-Machine-Interaction Lab, Chemnitz University of Technology, Chemnitz, Germany. Her research interests are in the field of production technology and service robotics, which include human-machine interfaces for robots, control architectures, automated assembly planning, and machine learning as well as computer vision for robotics.