Autonomous Robots (2019) 43:2033–2054
https://doi.org/10.1007/s10514-019-09847-2

# Task-centric optimization of configurations for assistive robots

Ariel Kapusta[1] · Charles C. Kemp[1]

Received: 7 February 2018 / Accepted: 9 March 2019 / Published online: 11 April 2019

**Abstract**
Robots can provide assistance to a human by moving objects to locations around the person's body. With a well-chosen initial configuration, a robot can better reach locations important to an assistive task despite model error, pose uncertainty, and other sources of variation. However, finding effective configurations can be challenging due to complex geometry, a large number of degrees of freedom, task complexity, and other factors. We present task-centric optimization of robot configurations (TOC), which is an algorithm that finds configurations from which the robot can better reach task-relevant locations and handle task variation. Notably, TOC can return one or two configurations to be used sequentially while assisting with a task. TOC performs computationally demanding optimizations offline to generate a function that rapidly outputs the configurations online based on the robot's observations. TOC explicitly models the task, environment, and user, and implicitly handles error using representations of robot dexterity. We evaluated TOC with a software simulation of a mobile manipulator (a PR2) providing assistance with 9 activities of daily living to a user in a wheelchair and a robotic bed. TOC had an overall average success rate of 90.6% compared to 50.4%, 43.5%, and 58.9% for three baseline algorithms based on state-of-the-art methods from the literature. We additionally demonstrate how TOC can find configurations for more than one robot and can help with the optimization of environments for assistance.

**Keywords** Mobile manipulation · Assistive robotics · Human–robot interaction · Robot autonomy

## 1 Introduction

Robotic assistance with activities of daily living (ADLs) (Wiener et al. 1990) could potentially enable people to be more independent. This may improve quality of life (Vest et al. 2011; Andersen et al. 2004) and help address societal challenges, such as aging populations, high health care costs, and shortages of health care workers found in the United States and other countries (Institute of Medicine 2008; Janiszewski Goodin 2000).

Many ADLs involve manipulation around a person's body. For example, tasks related to hygiene, feeding, and bathing often involve moving an object with respect to a person's body, such as an electric shaver, a spoon, or a sponge. As such, a key challenge for assistive robots is reaching task-relevant locations around a person's body. In this paper, we present task-centric optimization of robot configurations (TOC), which enables robots to better position themselves prior to providing assistance with ADLs. Figures 1 and 2 show the configurations selected for the cleaning legs task for a user in bed and the shaving task for a user in a wheelchair, respectively. Offline, TOC uses geometric and kinematic models of the assistive task, the human, the robot, and the environment to generate a function that rapidly estimates where the robot should position itself given the observed pose of a human body (see Fig. 4). Online, the robot uses this function to decide where to position itself in order to provide assistance with a task. In contrast to algorithms that only use a model of the robot, TOC generates a function that selects positions that take into account the human's capabilities,

We thank Henry and Jane Evans for providing input on assistive tasks and actuated beds. We also thank Daehyung Park, Zackory Erickson, Greg Turk, and C. Karen Liu for their valuable feedback. This work was supported in part by the National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR), Grant 90RE5016-01-00 via RERC TechSAge,and by NSF Awards IIS-1514258 and IIS-1150157. Dr. Kemp is a cofounder, a board member, an equity holder, and the CTO of Hello Robot, Inc., which is developing products related to this research. This research could affect his personal financial status. The terms of this arrangement have been reviewed and approved by Georgia Tech in accordance with its conflict of interest policies.

✉ Ariel Kapusta
akapusta@gatech.edu

[1] Georgia Institute of Technology, Atlanta, USA

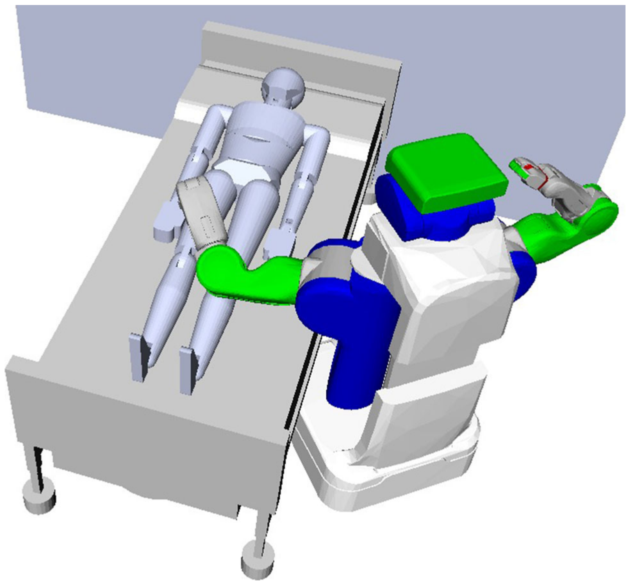


**Fig. 1** TOC can select a configuration for the PR2 and the robotic bed so the PR2 can better reach task-relevant locations. This figure shows the configuration for the task of a PR2 cleaning the legs of a human model in a robotic bed

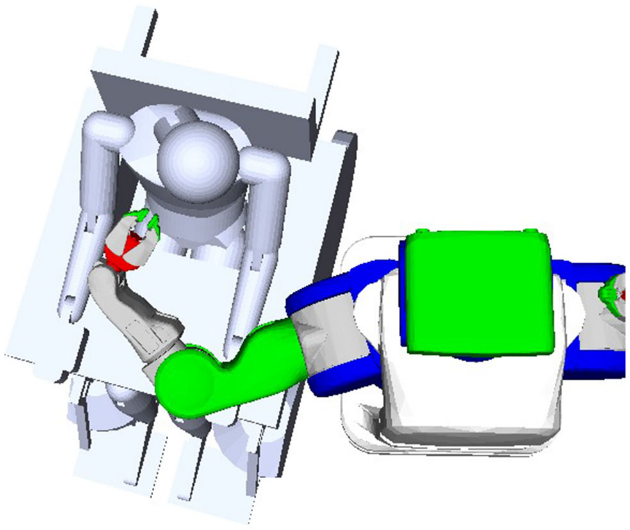


**Fig. 2** TOC can model cooperative motion from the human, such as the human model rotating its head

obstacles in the environment such as a bed or wheelchair, and the task requirements. More specifically, the function outputs one or two configurations of a robot from which the robot is likely to reach a set of target end-effector poses around the human body without collision, while achieving high dexterity at the target poses. Each configuration includes the pose of the robot's base as well as other configurable parameters selected for the task, such as the height of the robot's arm.

Specialized assistive devices can help people with motor impairments perform ADLs on their own (Canadian Partnership for Stroke Recovery 2015; The Wright Stuff, Inc. 2017). Specialized robots, such as desktop feeding devices, have been successful for a narrow range of assistive tasks when placed in fixed and designated positions with respect to the user (Topping and Smith 1998; Topping 1999). The choice of where to place such robots is important, as it can impact the robot's ability to provide effective assistance. General-purpose mobile manipulators have the advantage of mobility, which may allow them to provide assistance across a wide range of tasks, users, and environments (Chen et al. 2013). However, this mobility introduces additional complexity. For example, Hawkins et al. (2014) used two positions of a PR2 (a mobile manipulator made by Willow Garage) to provide assistance with shaving both the left and right sides of the face of a user in a wheelchair.

Our work focuses on the challenge of finding one or two good configurations from which a robot can provide assistance with activities of daily living (ADLs). ADLs are an important category of task that exhibits substantial structure. ADLs related to hygiene, bathing, and feeding can often be usefully modeled as a sparse set of end-effector poses around the human body. During assistance with ADLs, the human body is often reclining in a bed or sitting in a chair. The body's pose can vary between sessions of assistance, and can sometimes move during a task to help the robot reach a task-relevant location. For robotic assistance with ADLs, the tasks, the environment, the robot, and the user can often be specified in advance. We have developed TOC to fit well with this structure.

TOC performs an optimization with respect to a non-smooth, simulation-based objective function. It performs this optimization offline to generate a function that can be applied rapidly online to select robot configurations based on the robot's run time observations. To mitigate the effects of differences between the models used offline and the real world encountered by the robot, TOC's objective function uses two representations of robot dexterity that we developed, task-centric reachability (TC-reachability) and task-centric manipulability (TC-manipulability). TC-reachability and TC-manipulability help TOC efficiently select robot configurations that are robust to error. TOC optimizes its objective function, a linear combination of TC-reachability and TC-manipulability, using covariance matrix adaptation evolution strategy (CMA-ES) (Hansen 2006), a derivative-free optimization algorithm from the literature that only requires a function that can be evaluated.

We evaluated TOC in simulation with a PR2 assisting a user with 9 assistive tasks in both a wheelchair and a robotic bed. For our evaluations, we implemented three algorithms based on state-of-the-art methods from the literature as baselines. The first used an inverse-kinematics (IK) solver to select a configuration with a collision-free IK solution to all task-relevant goal poses. The second and third algorithms used a capability map from Zacharias et al. (2007) to select a configuration with high capability score to the goal poses. The third algorithm also checked that a collision-free IK solution existed for each goal pose.

We ran Monte-Carlo simulations of pose estimation error and found that TOC's average success rate was higher than or comparable to the other algorithms for each task. Success in these simulations was the robot being able to find a collision-free IK solution to all end-effector goal poses for the task. TOC had an overall average success rate of 90.6% compared to 50.4% for the IK solver, 43.5% for capability map, and 58.9% for capability map with collision checking. Additionally, we provide evidence that TOC's objective function is positively correlated with robustness to error, and we demonstrate how TOC can be used to assist in designing environments to improve robotic assistance. As we discuss in detail in Sect. 5, our results suggest that TOC would outperform the other baseline algorithms in real-world tests in the context of ADLs.

## 2 Related work

TOC relates to a number of past works on robot dexterity, methods for selecting robot configurations, proxemics, and assistive robotics.

### 2.1 Prior work from the authors

This paper relates to a previous conference paper and a three page workshop paper from the authors (Kapusta et al. 2015; Kapusta and Kemp 2016). In the conference paper (Kapusta et al. 2015), we introduced task-centric initial configuration selection (TCS) which used a brute-force, discretized approach to find initial configurations for robots. TCS was limited in the number of degrees-of-freedom it could handle and the quality of the solutions it could find due to coarse discretizations. In addition, TCS used an inferior objective function with different terms and did not model the ability of the user to move his or her body. Finally, the offline component of TCS output sets of configurations for online optimization that could be large in size. This is in contrast to TOC, which outputs an approximate function for rapid online use that requires no further optimization.

In our brief three page workshop paper (Kapusta and Kemp 2016), we presented a preliminary version of TOC with minimal content, including limited descriptions, evaluation, and related work. We have substantially improved the implementation of TOC that we now present. We also provide a thorough evaluation that compares TOCs performance against related algorithms in the literature and substantially more thorough descriptions of the algorithm and related work.

### 2.2 Representations of robot dexterity

Many metrics have been developed to quantify the kinematic dexterity of robot manipulators. These metrics can be broadly divided into those that use the manipulator's Jacobian, $\boldsymbol{J}(\boldsymbol{q})$ (Spong et al. 2006) and those that do not. These metrics can also be divided into those that find global measures (a metric for the robot irrespective of joint configuration) and those that find local, configuration-dependent measures of dexterity. Global dexterity metrics are often used for robot design (Stocco et al. 1998; Hammond and Shimada 2011). As we are focused on dexterity measures to assist in positioning existing robots, we will focus on discussing local metrics. Yoshikawa (1984) proposed the local Jacobian-based metric called measure of manipulability (or just, manipulability), $m(\boldsymbol{q})$, shown in Eq. (1).

$$m(\boldsymbol{q}) = \sqrt{\det(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^T)} \tag{1}$$

Geometrically, manipulability is proportional to the volume of the manipulability ellipsoid of the manipulator, which is the volume of Cartesian space moved by the end effector for a unit ball of movement by the arm's joints. This metric can be useful when assessing kinematic dexterity between different configurations of the same robot. However, its scale and order dependencies make comparison between different robot morphologies challenging.

Other dexterity measures were developed that address some of the issues of using manipulability (Klein and Blaho 1987). Kim and Khosla (1991) proposed another local Jacobian-based metric that we refer to in this paper as kinematic isotropy, $\Delta(\boldsymbol{q})$, shown in Eq. (2).

$$\Delta(\boldsymbol{q}) = \frac{\sqrt[a]{\det(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^T)}}{\left(\frac{\mathrm{trace}(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^T)}{a}\right)} \tag{2}$$

Kinematic isotropy uses the manipulability term (shown in Eq. 1) with an alteration to remove order dependency and divided by a term to remove scale dependency. Order is the degrees of freedom (DoF) of the Cartesian space of interest [e.g., SE(2) or SE(3)]. For a planar robot, the order would be three [the degrees of freedom of SE(2)] if translations are all in a 2D plane and in-plane rotations are considered. For the case of tasks in 6D space [position and orientation: SE(3)], the order is six [the degrees of freedom of SE(3)].

Unlike manipulability, the values of kinematic isotropy always range from 0 to 1. This may allow methods that use kinematic isotropy to be applied to new robot platforms more easily. In our work we modified kinematic isotropy, adding a weighting term to create what we call joint-limit-weighted kinematic isotropy (JLWKI). We use JLWKI in our task-centric manipulability (TC-manipulability) defined in Sect. 3.7. TC-manipulability represents the kinematic dexterity of the robot for a task (a set of goal poses) from a set of one or more positions of the robot.

An important limitation to some Jacobian-based measures of dexterity is their ignorance of many relevant features of the workspace, such as joint limits and collisions. The manipulability ellipsoid calculated from the Jacobian suggests that the end effector can move in ways that may be constrained by joint limits. Many researchers have proposed various ways to include these features [joint limits: Tsai (1986) and Chan and Dubey (1995); velocity limits: Lee (1997); torque limits: Hammond III and Shimada (2009)] into weighting terms. Vahrenkamp et al. (2012) created what they called an extended manipulability measure by modifying the Jacobian to include weights on joint limits, on proximity to self-collision, and on proximity to collision with the environment. Many of these methods apply their weighting terms directly to manipulability or indirectly by modifying the Jacobian, which is used in manipulability. JLWKI differs from other measures of dexterity, using a distinct weighting function on joint limits and applying it to kinematic isotropy. JLWKI, does not include costs on proximity to collision because we found that calculating these costs increases the computation time of TOC excessively.

Zacharias et al. (2007) introduced a method for representing manipulator dexterity without using Jacobians, which they use to score the workspace of a robot, creating what they call a capability map. To create the capability map they discretize space around the robot into 3D points and discretize the range of possible orientations around each 3D point. The capability score (also known as reachability score) is the number of orientations for which the robot has a valid IK solution. A way to interpret the meaning of this score is, if a goal pose is located at the 3D location, the capability score is similar to the probability that the manipulator can achieve the pose.

## 2.3 Selecting robot configurations for mobile manipulation

Prior research has investigated how to select configurations for a mobile robot. A common method is to address the problem using IK solvers (Diankov 2010; Beeson and Ames 2015; Kumar et al. 2010; Smits 2006). The entire kinematic chain from end effector to the robot's base location may be solved using IK (Gienger et al. 2005; Grey et al. 2016). Alternatively, sampling-based methods may be used to find robot base poses that have valid IK solutions, often as part of motion planning (Elbanhawi and Simic 2014; Stilman and Kuffner 2005; Lindemann and LaValle 2005; Garrett et al. 2015; Diankov et al. 2008). TOC also uses a sampling-based search method, specifically CMA-ES, to perform its optimization of robot configurations.

By relying solely on IK to ensure that the robot can reach the goals, these prior methods are dependent on accurate models. Many of these methods are fast, but may fail if there is modeling or state estimation error. Additionally, there are often many robot configurations with valid IK solutions that cannot be distinguished using only IK, as shown in Fig. 3 (left). All locations in green have collision-free IK solutions to all goals, but some may result in higher success rates than others. Success is the robot being able to reach all goals despite state estimation error. TOC uses task-centric manipulability to differentiate those configurations, as shown in Fig. 3 (right). We show in Sect. 4.4 that higher TOC score is correlated with improved performance for configurations that have collision-free IK solutions to all goals. Those results provide evidence that, using task-centric manipulability, TOC is more robust to state estimation error. Additionally, TOC can find more than one robot configuration for a task. We implemented a standard IK sampling-based method as a baseline for comparison, as described in Sect. 4.2.

A body of work is based on the capability map from Zacharias et al. (2007). Capability-map-based methods are robot-centric and task-agnostic; they are generated offline for the robot's manipulator and applied to tasks online. They typically select the robot base position by overlapping the capability map with end-effector goal poses and maximizing the average capability score (Zacharias et al. 2009; Porges et al. 2014; Leidner et al. 2014). Dong and Trinkle (2015) altered the capability map by creating an orientation-based capability map and extending the map for tools on the robot's end effector.

In contrast with our method, existing capability-map-based methods do not consider collisions with the environment in their offline computations because they do not model the environment. Collisions are only considered at runtime to eliminate robot base locations in collision or that lack collision-free IK solutions. Simply selecting the robot base location with highest capability map score is fast, but searching for a collision-free location can take more time. Capability maps are often used to provide a sample of base poses that can each reach all goal poses without consideration of collisions. The base poses can then be filtered based on task constraints. Capability-map-based methods typically only find a single location for a robot for a task, but our method can find multiple robot configurations. We implemented two capability-map-based methods as baselines for comparison, based on Zacharias et al. (2009), as described in Sect. 4.2.

Another body of work extends the capability map by inverting it, creating an inverse-reachability map. While a capability map scores end-effector poses with respect to a robot base pose, the inverse-reachability map scores robot base poses with respect to an end-effector pose. As with

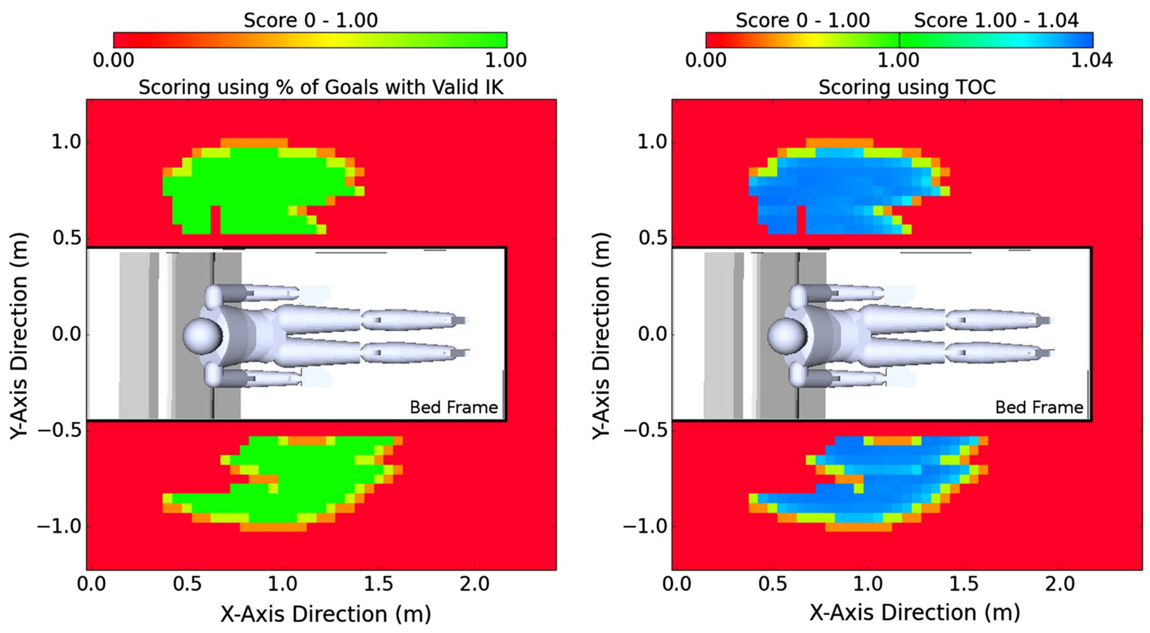


**Fig. 3** TOC differentiates between robot configurations that have collision-free IK solutions to all goal poses. This differentiation allows TOC to better select good robot configurations. The figure visualizes two scoring methods, showing the score for discretized PR2 base poses. Z-rotation is sampled every 45° from 0° to 360°, and the best score is shown (left) using the percentage of goals with collision-free IK solutions and (right) TOC, for the mouth wiping task in the robotic bed environment, with the bed raised 20 cm and at 45°. The color represents the best score for that 2D position of the PR2. 0 means no goals and $\geq 1$ means all goals have collision-free IK solutions

the capability map, the inverse-reachability map is generated offline for the robot's manipulator and can be used quickly online. For an end-effector 3D position, discretized robot base poses are scored based on the capability map score to that 3D position.

The inverse-reachability map is used to rapidly sample a robot base position that can reach a set of goal end-effector poses (Diankov and Kuffner 2008; Burget and Bennewitz 2015). Vahrenkamp et al. (2013) used an alternative representation of the robot's dexterity from their previous work (Vahrenkamp et al. 2012) that uses 6D poses in the workspace. They invert that workspace representation to create what they call an Oriented Reachability Map (ORM). ORM, like the inverse-reachability map, scores robot base poses based on the extended manipulability measure of each 6D pose in the manipulator's discretized workspace. When searching for a robot base pose for a task, they sample in series from the ORM using the map's score as a sampling weight. If the sampled robot base pose has collision-free IK solutions to the goal end-effector poses, they use that base pose; if not they re-sample. They propose methods to incorporate task-specific information in their extended manipulability measure (and thus into the ORM), and to calculate the ORM map online through what they call lazy-ORM.

Inverse-reachability-based methods and ORM differ from our work in a few ways. These methods are typically used in task-agnostic and robot-specific ways to facilitate applications of the robot to new tasks. Notably, these methods only find a single location for a robot for a given task, rather than multiple robot configurations. TOC also explicitly models features and parameters of the environment and user that may be important to assistive tasks. Details on this modeling is found in Sects. 3.5 and 3.6.

Most previous task-centric methods use simulation of the task, with explicit error modeling, to evaluate robot base poses. Hsu et al. (1999) presented a task-specific method for selecting a place for an industrial robot manipulator to perform a series of tasks amidst clutter. They used randomized path planners to generate collision-free paths for the arm and they randomly perturbed the robot position to find positions from which the tasks can be performed quickly.

Stulp et al. (2009) presented a task-centric method for finding areas in which to place a mobile manipulator where it can successfully perform a grasping task. They used Monte-Carlo simulation of error in the location of the object to be grasped to find base positions with high success rates. They simulated performance of the entire task including, navigation, motion planning and motion execution. For real-time base position selection, they convolved uncertainty in robot location with base position scores to provide an area of high-success probability. They used their method to select a 2D position of the robot base for a grasping task. These task-centric methods that explicitly model error and fully simulate task performance have only been used to select a few degrees of freedom in static environments, and can only select a single robot configuration for a task. In contrast, TOC uses faster, simpler simulations and implicitly handles error. TOC selects more degrees of freedom in configurable environments, and, again, can select multiple robot configurations for a task.

### 2.4 Human–robot proxemics

Several bodies of work have examined the proxemics of human–robot interactions (Walters et al. 2011, 2009; Mumm and Mutlu 2011; Takayama and Pantofaru 2009; Walters et al. 2005).

Proxemics is the study of the spatial requirements of humans (e.g., the amount of space that people feel they should have between themselves and others). These works look at acceptable interpersonal distances between humans and robots in social settings. Various works have used the concepts of human–robot proxemics to inform a robot when performing tasks. These works coupled task performance concepts with scoring methods based on proxemics to select base positions and paths for the robot and item handover locations (Sisbot et al. 2010; Mainprice et al. 2011; Sisbot et al. 2006, 2007). Proxemics might suggest that placing the robot in front of the person at some minimum distance is preferred over other locations.

Kruse et al. (2013) presented a thorough survey of human-aware robot navigation. In contrast, TOC does not consider proxemics or social factors; it instead focuses on kinematic aspects of the task. While proxemics is often used to consider navigation problems, TOC focuses exclusively on selecting the configuration for the task, which can serve as a goal pose for navigation.

### 2.5 Assistive robots

Researchers have investigated the use of mobile manipulators as assistive devices (Dario et al. 1999; Schaeffer and May 1999; Graf et al. 2009; Bien et al. 2004; Jain and Kemp 2010; Hawkins et al. 2014).

We seek to further empower assistive mobile manipulators by autonomously selecting configurations from which they can better provide assistance. This autonomy might improve task performance and decrease cognitive workload for teleoperated assistive systems, as from Grice and Kemp (2016). In this work, we have used a model of a robotic bed that matches Autobed, a robotic modified hospital bed from Grice et al. (2016). We have shown how TOC can optimize the configuration of the bed, allowing improved assistance from a mobile manipulator as part of a collaborative assistive system, as from Kapusta et al. (2016). This capability is demonstrated in our evaluations in the robotic bed environment in Sect. 4.2.

## 3 Task-centric optimization of robot configurations (TOC)

As mentioned in Sect. 1, key features of task-centric optimization of robot configurations (TOC) are its task-centric approach, representations of robot dexterity, selection of multiple configurations for a task, and framework that splits offline and online computation. TOC is suitable for situations when tasks and environment layouts are known beforehand and we would like to configure the robot for these tasks such that the robot is successful despite variations between models and reality. By taking a task-centric approach, TOC is able to use task-specific knowledge, such as explicit modeling of task-relevant parameters, to better select configurations. We will first explain the goal of TOC. Afterwards we describe the nomenclature used in the remainder the paper. We then explain the framework of TOC, details of its features, and specifics of our implementation.

### 3.1 TOC goal: selecting good configurations

The goal of TOC is to select a good set of one or two configurations for a robot to perform a task without additional adjustments. But what constitutes a *good* robot configuration? In this paper we use the term *robot configuration* as a more general term for the pose of the robot's base, so it can include additional relevant parameters. For example, for a PR2, the robot configuration might be the position and orientation of the robot's mobile base as well as the z-axis spine height of the robot. If the PR2 were operating in a room with a robotic bed, the degrees of freedom (e.g., the height of the bed) of the bed could be included in the robot configuration.

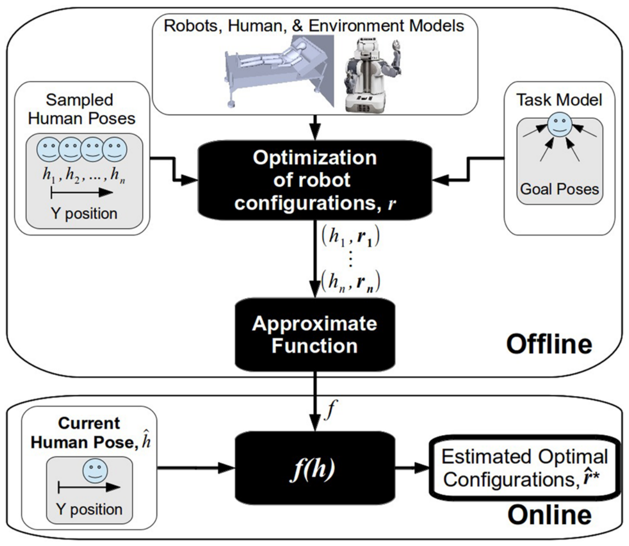
While all of the empirical results we present produce one or two configurations, for our formal treatment we allow the number of robot configurations to be greater than two. We consider robot configurations in sets that can be of cardinality 1 or greater; the robot can complete the entire task by adopting all configurations in the set in any order. With a *good* set of configurations, the robot is more likely to be able to complete the task successfully. We judge the robot's ability to perform the task from a set of robot configurations with one measure: if it can reach all goal poses collision-free with its end effector. Various forms of error, such as modeling error or state estimation error, may cause the robot to be unable to perform the task. Because the robot does not know how the modeling and state estimation error will manifest apriori, from a good set of robot configurations, the robot should be able to perform the task despite such error.

## 3.2 Nomenclature

| | |
|---|---|
| $t$: | A task identifier |
| $N_t$: | The number of goal poses for task $t$ |
| $\boldsymbol{q}$: | A joint configuration of the robot arm. $\boldsymbol{q} \in \mathbb{R}^n$, where $n$ is the number of DoF of the arm |
| $q_i$: | The value for joint $i$ in joint configuration $\boldsymbol{q}$, $q \in \boldsymbol{q}$ |
| $\boldsymbol{q}^-, \boldsymbol{q}^+$: | A list of the minimum and maximum values, respectively, for the joints of a robot's arm. |
| $q_i^-, q_i^+$: | The minimum and maximum values, respectively, for joint $i$ of a robot's arm. |
| $\boldsymbol{r}$: | A set of robot configurations of cardinality $\geq 1$, $\boldsymbol{r} = \{r_1, r_2, \ldots, r_n\}$, where $n$ is the number of robot configurations in set $\boldsymbol{r}$. We used $n \in \{1, 2\}$ in our implementation of TOC for our evaluation. |
| $\hat{\boldsymbol{r}}^*$: | The estimated optimal robot configurations given current observations, the output of the online portion of TOC. |
| $\boldsymbol{h}$: | The set of uncontrollable parameters, discretized into $\{h_1, h_2, h_3, \ldots\}$ |
| $\hat{h}$: | The uncontrollable parameters observed and estimated at run time, the input to the online portion of TOC. |
| $\boldsymbol{b}$: | The set of free parameters, discretized into $\{b_1, b_2, b_3, \ldots\}$ |
| $\boldsymbol{p}$: | Set of position and orientation end-effector goal poses $p \in \mathbb{R}^6$. $\boldsymbol{p}$ depends on $t$, $h$, and $b$, but we do not explicitly denote these dependencies in order to simplify our notation. $\boldsymbol{p} = \{p_1, p_2, \ldots, p_{N_t}\}$. |
| $\boldsymbol{s}_{r,x}$: | Set of IK joint configuration solutions to goal $p$ from robot configuration $r$, $\boldsymbol{s}_{r,x} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n\}$, where $n$ is the number of IK solutions |
| $a$: | The order of the robot arm. In our case, 6. |
| $\boldsymbol{J}(\boldsymbol{q})$ | The Jacobian of the arm in joint configuration $\boldsymbol{q}$ |
| $\Delta(\boldsymbol{q})$: | The kinematic isotropy for the arm in joint configuration $\boldsymbol{q}$ |
| $f$: | A function that takes $\hat{h}$ as input and outputs $\hat{\boldsymbol{r}}^*$. TOC generates $f$ offline and applies it online. |

## 3.3 Framework

Figure 4 shows the framework of TOC for our implementation described in Sect. 4 for a person on a robotic bed. TOC is split into an offline portion, which includes most of its computation, and an online portion. Offline, it performs the optimization of robot configurations and approximates a function that can be used online to select robot configurations for a task. The optimization takes as input task-relevant models (e.g., task, robot, user, and environment models) and a sample of the uncontrollable parameters (e.g., the position of the human model on the bed), $h$. It outputs an optimized robot configuration, $\boldsymbol{r}$, for that $h$. The $h$ inputs and $\boldsymbol{r}$ outputs



Robots, Human, & Environment Models
Sampled Human Poses
$h_1, h_2, \ldots, h_n$
Y position
Optimization of robot configurations, $r$
Task Model
Goal Poses
$(h_1, \boldsymbol{r}_1)$
$(h_n, \boldsymbol{r}_n)$
Approximate Function
Offline
$f$
Current Human Pose, $\hat{h}$
Y position
$f(h)$
Estimated Optimal Configurations, $\hat{\boldsymbol{r}}^*$
Online

**Fig. 4** This is a visualization of TOC's framework. The offline portion of TOC takes as input task-relevant models and samples of the uncontrollable parameters and outputs optimized robot configurations. It then approximates a function that is used online to estimate the optimal robot configurations given the current, observed uncontrollable parameters

of the optimization are used to approximate the function, $f$ that takes as input the observed estimated uncontrollable parameters, $\hat{h}$ and outputs the estimated optimal configurations, $\hat{\boldsymbol{r}}^*$. The function $f$ is applied online, at run time. The estimated optimal configurations, $\hat{\boldsymbol{r}}^*$ should allow the robot to perform the task despite errors in state estimation (e.g., estimating $\hat{h}$) and in modeling (e.g., models do not exactly match reality).

## 3.4 Task modeling

Our aim with task modeling is to create a representation that allows TOC to efficiently evaluate a robot's ability to perform a task. There are many tasks that consist of manipulation of small objects or tools around a person's body, that we expect can be well modeled by a set of goal poses (Cartesian positions and quaternion orientations) with respect to relevant reference frames. We manually model each task as a sparse set of poses for the robot's end effector. For example, Fig. 5 shows the eight goal poses with respect to the human model's head that make up our model of a shaving task. We assume that if the robot can reach all goal poses, it is likely to be able to perform the task. However, we expect there to be differences between the models and the real tasks. We consider these discrepancies to be a form of modeling error that TOC accounts for when selecting robot configurations.

## 3.5 Environment modeling

Using its environment model, TOC finds robot configurations that avoid collisions with obstacles, such as a bedside

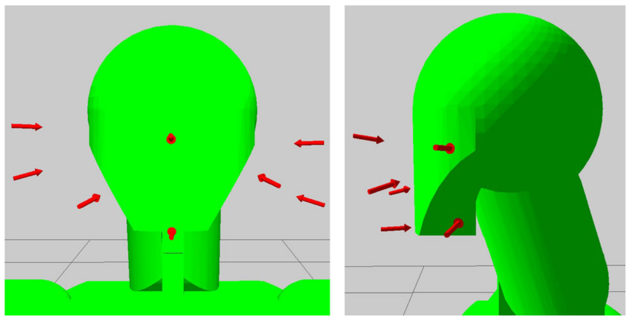


**Fig. 5** The manually selected goal poses for the shaving task. Each arrow represents a 6-DoF end-effector goal pose (a position and an orientation) with respect to the head. This shows views from the front and side

table or walls. TOC can use different resolutions for its environment model depending on the needs of the task. A room could be be represented simply as a wall behind the bed, as shown in Fig. 1, or it could contain models of furniture and other potential obstacles. The resolution of each object model could range from a block to a detailed mesh. TOC uses three types of objects to model the environment:

- Fixed objects
- Controllable movable/configurable objects
- Uncontrollable movable/configurable objects

The object types are distinguished by parameters that define them. Fixed objects are static objects in the world that cannot be moved and must be avoided by the robot. Their shapes, poses, and configurations are static. The other two types of objects are configurable, having some parameters that may be changed, such as their pose in the environment or their configuration (e.g., the height of an adjustable bed).

We add the configuration of the controllable movable/configurable objects to the robot configuration space that TOC optimizes, as the robot can control their configurations. An example of a controllable object is an adjustable bed.

For uncontrollable movable/configurable objects, TOC samples the possible configurations of the objects and then optimizes robot configurations for each sample. At run time, the robot observes the configurations of uncontrollable objects and uses these observations to select robot configurations. An example of an uncontrollable object is a bed, or more specifically, the position of a bed with respect to the walls of the room. The bed's position in the room could be moved by a person prior to the robot starting the task, but is not easily adjusted by the robot.

Using controllable and uncontrollable movable objects, TOC can suggest alterations to the environment that may improve task performance. Uncontrollable movable objects can also be used to generate robot configurations for possible states of the environment. This may be beneficial for environments, such as hospitals, where there are a few possible room layouts, but the robot may not know which layout will be relevant until it reaches the room.

Because TOC does not include a cost in proximity to collision between the robot and the environment in its objective function, we include a margin of safety in the environment model by expanding the environment model (we used $\sim 3$ cm in our evaluations). This safety margin reduces the risk of collision in the case of model or state estimation error, without having to explicitly include closeness to collision in the objective function.

## 3.6 User modeling

TOC's user model can be customized for a user to better locate relevant parts of the body, and to allow more accurate collision-checking. In our evaluation of TOC, we used a mesh model of a human designed around a 50 percentile male from Tilley (2002), shown in Fig. 1. TOC uses three types of parameters for the human model's joint configuration:

- Environment-driven parameters
- Uncontrollable parameters
- Free parameters

Environment-driven parameters are dependent on the state of the environment model. As such, the extent to which the robot can control them depends on the robot's control of the environment. For example, a human model's configuration would depend on the angle of the backrest of an adjustable bed, and the robot may be able to control the backrest.

Uncontrollable parameters of a human model are treated similarly to uncontrollable movable objects. For uncontrollable parameters, TOC samples the possible configurations of the human model and then optimizes robot configurations for each sample. At run time, the robot observes the configuration of the human body and uses this observations to select robot configurations. An example of an uncontrollable parameter is the position of the human model on the bed, if the robot is unable to shift the body on the bed.

Free parameters represent degrees of freedom that TOC assumes the person will cooperatively and voluntarily control to help the robot reach the goal poses. The robot cannot control these parameters, but can assume that the person will modify them cooperatively. An example of a free parameter we used is the user's head rotation. Figure 2 shows a configuration of the PR2 in which the cooperative motion of the human's head allows the robot to reach all goals for the shaving task in a wheelchair.

Just as with the environment model, we include a margin of safety in the human model by expanding the model.

#### 3.6.1 Additional user customization

TOC can consider additional customizations for the user's needs or preferences. For example, a user may prefer certain angles of the bed's head rest for feeding tasks. This preference can be represented as limitations or costs on the robot's configuration space.

## 3.7 Configuration scoring

Implicitly handling variation and error is a key aspect of TOC, because its heavy computation is performed offline for models that may differ from reality. TOC uses two metrics that we have developed to estimate how well the robot will be able to perform the task from a set of configurations: task-centric reachability (TC-reachability) and task-centric manipulability (TC-manipulability).

#### 3.7.1 Task-centric reachability

Task-centric reachability (TC-reachability), $R_{\text{reach}}$, is the percentage of goal poses to which the robot can find a collision-free IK solution from robot configurations, $\boldsymbol{r}$, for a task $t$ and uncontrollable parameters $h$, as shown in Eq. (3).

$$R_{\text{reach}}(\boldsymbol{r}, t, h) = \left(\frac{1}{N_t}\right) \sum_{k=1}^{N_t} \max_{r \in \boldsymbol{r}, b \in \boldsymbol{b}} F(r, p_k), \tag{3}$$

where

$$F(r, p) = 1 \quad \forall \boldsymbol{s}_{r,p} \neq \emptyset,$$
and (4)
$$F(r, p) = 0 \quad \forall \boldsymbol{s}_{r,p} = \emptyset.$$

Recall that $\boldsymbol{p}$ depends on $t$, $h$, and $b$, but we omit those for simplicity in writing and $N_t$ is the number of goal poses for task $t$. Note that $\boldsymbol{s}_{r,p_k} \neq \emptyset$ means that the IK solver can find a collision-free solution to the goal pose $p_k$ from robot configuration, $r$.

TC-reachability is related to using an IK solver with collision checking, but with the additional functionality of evaluating sets of robot configurations.

#### 3.7.2 Task-centric manipulability

Task-centric manipulability (TC-manipulability), $R_{\text{manip}}$, is related to the average kinematic dexterity of the arm when reaching the goal poses. It is defined here differently from our previous works, such as in Kapusta et al. (2015). After computing TC-reachability, computing TC-manipulability is very efficient, since it uses a computationally simple dexterity measure and the same collision-free IK solutions found while computing TC-reachability.

The TC-manipulability score is based on kinematic isotropy (Kim and Khosla 1991), shown in Eq. (2). Kinematic isotropy only considers the Jacobian of the arm in a configuration, ignoring potentially relevant properties of the robot arm, such as joint limits. When at a joint limit, the arm cannot move in one direction, effectively halving the movement of that joint. Hammond III and Shimada (2009) used torque-weighted global isotropy index and torque-weighted kinematic isotropy to estimate the dexterity of a robotic arm given joint torques and torque limits. Vahrenkamp et al. (2012) investigated configuration-based weighting functions to create what they call an augmented Jacobian that they use in manipulability. We have similarly modified kinematic isotropy to consider joint limits by scaling the manipulator's Jacobian by an $n \times n$ diagonal joint-limit-weighting matrix $\boldsymbol{T}$, where

$$\boldsymbol{T}(\boldsymbol{q}) = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_n \end{bmatrix}, \tag{5}$$

and where $n$ is the number of joints of the manipulator. The transmission weight, $w_i$, in $\boldsymbol{T}$ is defined as

$$w_i = 1 - \phi^{\kappa}$$
where
$$\kappa = \frac{q_i^r - |q_i^r - q_i + q_i^-|}{\lambda q_i^r} + 1 \tag{6}$$
and
$$q_i^r = \frac{1}{2}(q_i^+ - q_i^-).$$

We set $w_i = 1$ for infinite roll joints. The variable $\phi$ is a scalar that determines the maximum penalty incurred when joint $q_i$ approaches its maximum and minimum values, $q_i^+$ and $q_i^-$, and $\lambda$ determines the shape of the penalty function. We used a value of 0.5 for $\phi$ and 0.05 for $\lambda$. This weighting function and the values for $\phi$ and $\lambda$ were selected to halve the value of the kinematic isotropy at joint limits, have little effect in the center of the joint range, to begin exponentially penalizing joint values beyond 75% of the range, and to operate as a function of the percentage of the joint range. Figure 6 shows the value of $w_i$ as a function of the joint value as a percentage of its joint range.

We then define joint-limited-weighted kinematic isotropy (JLWKI) as

$$\text{JLWKI}(\boldsymbol{q}) = \frac{\sqrt[a]{\det(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{T}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^T)}}{\left(\frac{1}{a}\right)\text{trace}(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{T}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})^T)}. \tag{7}$$

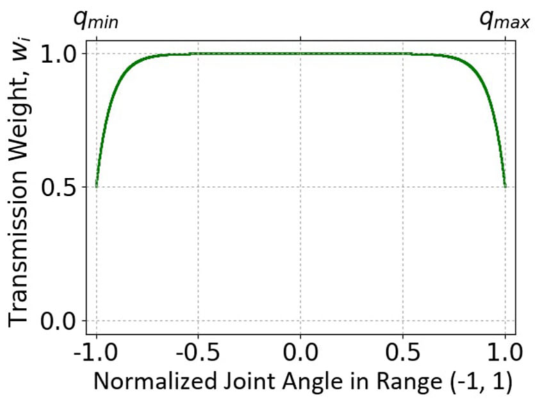


**Fig. 6** A plot of the joint-limit weighting function ranging from the maximum joint value to the minimum joint value

We use a function, $G$, to find the maximum value of JLWKI($\boldsymbol{q}$) for robot configuration $r$ and goal pose $p$, where

$$G(r, p) = \max_{\boldsymbol{q} \in \boldsymbol{s}_{r,p}} \text{JLWKI}(\boldsymbol{q}) \quad \forall \boldsymbol{s}_{r,p} \neq \emptyset,$$

and (8)

$$G(r, p) = 0 \quad \forall \boldsymbol{s}_{r,p} = \emptyset.$$

We finally define TC-manipulability, $R_{\text{manip}}$, as

$$R_{\text{manip}}(\boldsymbol{r}, t, h) = \left(\frac{1}{N_t}\right) \sum_{k=1}^{N_t} \max_{r \in \boldsymbol{r}, b \in \boldsymbol{b}} G(r, p_k). \tag{9}$$

## 3.8 Optimization method

TOC's optimization takes as input task-relevant models for task, $t$, and samples of the uncontrollable parameters, $h$. It outputs an optimized set of robot configurations, $\boldsymbol{r}$. TOC runs this optimization for samples of the uncontrollable parameters for each task.

### 3.8.1 Objective function

Each discretized value, $h_i$, of the uncontrollable parameters, $\boldsymbol{h}$, has an associated solution set of robot configurations, $\boldsymbol{r}_i$. To find $\boldsymbol{r}_i$, TOC searches the robot configuration space to maximize its objective function, a linear combination of TC-reachability and TC-Manipulability, shown in Eq. (10).

$$\arg\max_{\boldsymbol{r}_i} \quad \alpha R_{\text{reach}}(\boldsymbol{r}_i, t, h_i) + \beta R_{\text{manip}}(\boldsymbol{r}_i, t, h_i) \tag{10}$$

Both TC-reachability and TC-manipulability range from 0 to 1, allowing them to be directly compared in the objective function. We selected a value of 1 for $\alpha$. We chose to define $\beta$ as

$$\beta(\boldsymbol{r}) = (0.1)(0.95)^{n-1} \tag{11}$$

where $n$ is the cardinality of $\boldsymbol{r}$.

In our implementation of TOC the cardinality of $\boldsymbol{r}$ was 1 or 2. These definitions of $\alpha$ and $\beta$ emphasize the importance of reaching goals over having high dexterity at goals, and includes a small penalty in the objective function's value for using more configurations. For tasks with fewer than 10 goals (as in all tasks we implemented), the objective function's value for a set of configurations that can reach more goals will always be greater than one that can reach fewer goals. In other word, reaching another goal is always more important than increased dexterity.

TC-manipulability serves to impose preferences over configurations that reach the same number of goals. There are often many configurations that can reach all goals. As an example, we compare TOC to a standard method from the literature: using an IK solver with a collision checker to find a robot configuration that can reach all goals. Figure 3 shows the difference in scoring between using the existence of IK solutions for scoring and using TOC for scoring for a task for a user in bed. Figure 3 (left) shows that many poses of the robot's base have the same score, each having collision-free IK solutions to all goals. Figure 3 (right) shows scoring using TOC, where TC-manipulability allows additional differentiation between robot base poses that can reach all goals. Higher TC-manipulability is correlated with mean accuracy (percentage of goals that are reachable), as we show in Sect. 4.4.

### 3.8.2 Optimization algorithm

The space of the objective function can be highly nonlinear and challenging to search. The objective function does not have an analytical gradient and estimating its gradients can be computationally expensive. There are several derivative-free optimization algorithms that could be applied to this problem. A simple method would be to uniformly sample the space and select the configuration with highest objective function value. However, uniform sampling performs poorly as the dimensionality of the space increases, and we found that coarse sampling of the space often resulted in poor configurations.

Tan et al. (2011) used Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (CMA 2018; Hansen 2006) to design a controller for articulated bodies moving in a hydrodynamic environment, which inspired our use of CMA-ES. TOC optimizes the objective function using CMA-ES, which works well for derivative-free optimization. TOC jointly optimizes the set of configurations, so the dimensionality of the search space is $m \times n$, where $m$ is number of configurations and $n$ is the number of degrees of freedom being optimized. We used a heuristic when both $R_{\text{reach}}$ and $R_{\text{manip}}$ are zero that pushes the search toward configurations that may have non-zero $R_{\text{reach}}$ and $R_{\text{manip}}$. If the distance from the robot to all goals is larger than the maximum reach of the robot, the heuristic subtracts a value proportional to the extra distance. If the robot's base is in collision with the bed or the wheelchair, the heuristic subtracts a value proportional to the depth of penetration of the base with the object.

### 3.9 Approximate function

Offline, TOC generates a function that takes as input an estimation of the uncontrollable parameters, $\hat{h}$, such as the location of the person on the bed, and outputs estimated optimal robot configurations, $\hat{\boldsymbol{r}}^*$. At run time, TOC applies this function to the observed, estimated uncontrollable parameters. For this paper, we used K-nearest neighbor (K-NN) with $K = 1$ (hence, 1-NN) as the function, $f$. We trained the 1-NN algorithm from Pedregosa et al. (2011) with a set of ($h$, $r$) pairs and it returns as $\hat{\boldsymbol{r}}^*$ the $r$ for the $h$ that is closest to $\hat{h}$. The algorithm creates a rapidly explorable data structure fit to the data for faster computation of the nearest neighbor. In our implementations of TOC we trained the 1-NN on fewer than 20 ($h$, $r$) pairs for each task and found that the 1-NN would return $\hat{\boldsymbol{r}}^*$ in less than 1 s running in a single thread on a a 64-bit, 14.04 Ubuntu operating system with 8 GB of RAM and a 3.40 GHz Intel Core i7-3770 CPU. The use of other methods to approximate this function could merit further investigation.

## 4 Evaluation

### 4.1 Implementation

We manually created models for 9 assistive tasks associated with activities of daily living (ADLs): shaving, feeding, wiping the mouth, cleaning both arms, cleaning both legs, and scratching the left/right upper arm, and left/right knee (each scratching task was considered separately). We chose these tasks as representative of various activities for which a robot like the PR2 could provide assistance to a user with motor impairments. Previous work has noted that these types of tasks may be useful for those with severe motor impairments (Wiener et al. 1990; Chen et al. 2013). As described in Sect. 3.4, task models consisted of a set of end-effector goal poses, each of which specified a position and orientation. We defined each goal pose with respect to a relevant reference frame (e.g., the head for shaving, or the shoulder for scratching the upper arm), so they move appropriately as the model parameters change (e.g., the height of the bed or the pose of the human body).

As an example, Fig. 5 shows the eight goal poses with respect to the human model's head that we selected to model the shaving task. For simplicity, we limited tasks to one-handed tasks and only allowed the robot to use its left arm in our evaluations. In our implementation, we allowed TOC to search for sets of robot configurations of cardinality 1 or 2. When exploring multiple robot configurations for a task, we assume the robot can move from one configuration to another.

We ran all simulations in OpenRAVE (OpenRAVE 2018; Diankov and Kuffner 2008), for which we created environment models with a PR2 robot and a model of an average male human placed either in a wheelchair or in a robotic bed. The robotic bed is a controllable configurable object (see Sect. 3.5) and TOC includes the bed's configuration as part of the robot's optimized configuration, $\boldsymbol{r}$. The bed's configuration is 2-DoF, the height of the bed and the angle of the backrest. The human model dimensions come from Tilley (2002). The PR2 is a mobile manipulator made by Willow Garage with two 7-DoF arms. The models we created for the robotic bed and the wheelchair match Autobed, a modified Invacare 5401IVC full electric hospital bed (Grice et al. 2016) and a Sunrise Medical Quickie 2 wheelchair with overlap table, respectively. The casters on the bed and wheelchair are represented by swept volumes. For the wheelchair, we removed the part of the casters' swept volumes that extends to the sides of the chair to increase free space around the chair. We assume that the user would ensure that the casters are not pointing out from the chair.

Figure 7 shows the configurations selected by our implementation of TOC for each task, given the observation, $\hat{h}$, that the human model was positioned in the center of the bed or wheelchair.

The robotic bed environment has the bed in front of a wall, to emulate how beds are often positioned in rooms. The robotic bed can raise up to 25 cm and can increase the angle of its head rest up to 75°. For the wheelchair environment we gave the human model the ability to rotate its head up to 45° in either direction about the Z-axis. Figure 2 shows the head rotated 45°.

The robotic bed environment and the wheelchair environment demonstrate many of the functionalities of the TOC framework. The robotic bed environment demonstrates how TOC can select configurations for multiple robots in the environment, since each optimized configuration includes the configuration for the PR2 and the robotic bed. TOC treats additional robots the same as controllable objects, adding their parameters to the robot configuration, as it does with the robotic bed's parameters. TOC considers the position in the X–Y plane (see Fig. 3) of the human on the bed as an uncontrollable parameter (2-DoF) and considers the joint configuration of the human model as environment-driven parameters. The wheelchair environment demonstrates how TOC can make use of free parameters in the user model. TOC considers the rotation of the human model's head as a free parameter, which results in solutions that assume the human will move his or her head to help the robot reach goal poses. The remainder of the human model's joint configuration are

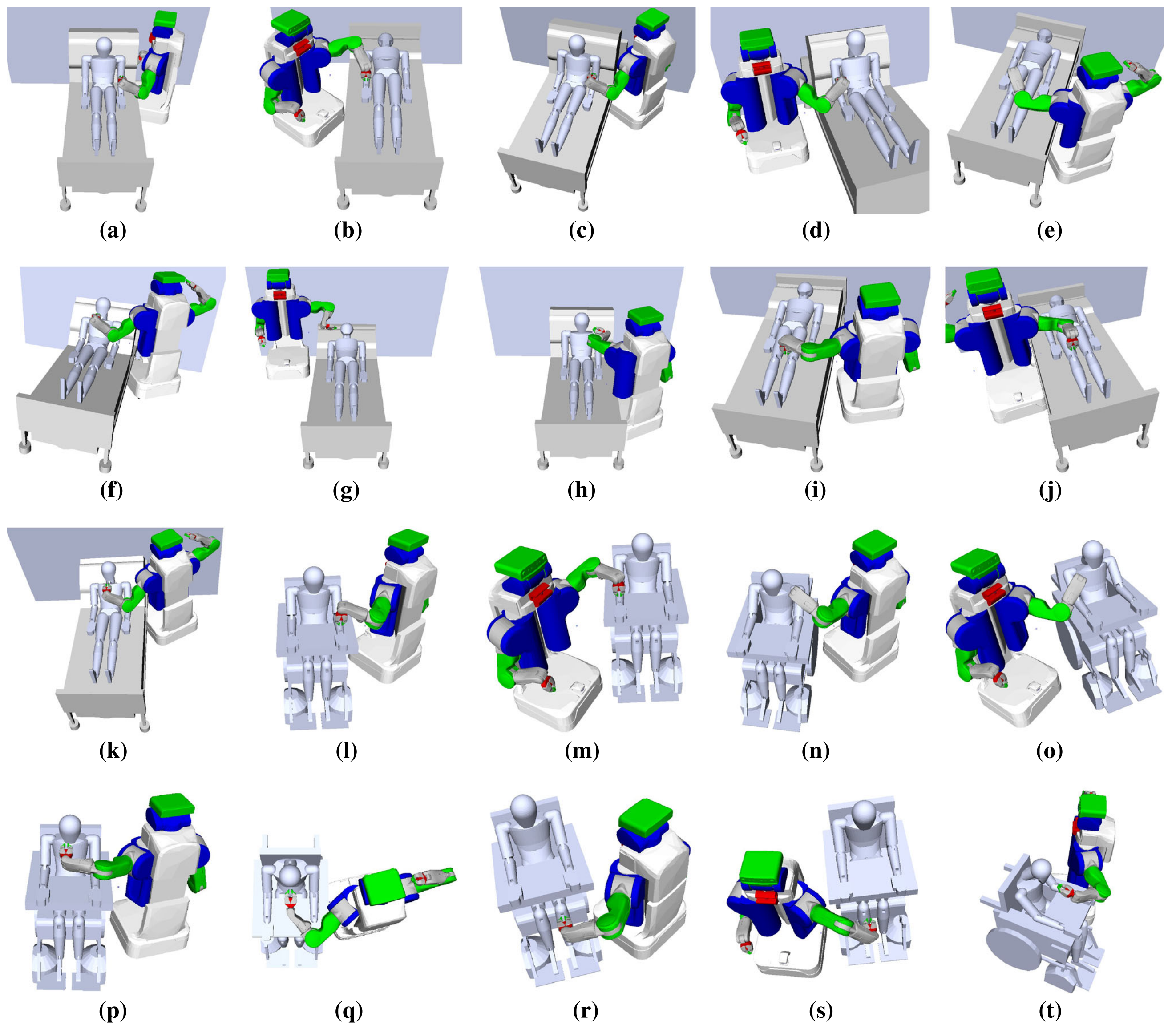


**Fig. 7** Visualization of the robot configurations selected by TOC for each task. Two images are used when TOC selected two configurations for a task. Images of configurations for the robotic bed environment: **a** cleaning arms config #1, **b** cleaning arms config #2, **c** scratching left upper arm, **d** scratching right upper arm, **e** cleaning legs, **f** wiping mouth, **g** shaving config #1, **h** shaving config #2, **i** scratching left knee, **j** scratching right knee, **k** feeding. Images of configurations for the wheelchair environment: **l** arm cleaning config #1, **m** arm cleaning config #2, **n** scratching left upper arm, **o** scratching right upper arm, **p** wiping mouth, **q** shaving, **r** scratching left knee, **s** scratching right knee, **t** feeding

environment-driven and set the human in a seated pose in the chair.

## 4.2 Evaluation against baselines

We compared the performance of TOC against three baseline algorithms based on state-of-the-art methods in Monte Carlo simulations that added Gaussian error to the human model's position and the robot's base pose. Error in the human model's position is comparable to a robot having error in its estimate of a person's body pose, which is a form of state estimation error. Error in the robot's base pose is comparable to errors that could occur when a robot attempts to navigate to a configuration, which could be due to a variety of causes.

Each method in our evaluation estimated an optimal set of robot configurations, $\hat{\boldsymbol{r}}^*$, given the observation, $\hat{h}$, that the human model was positioned in the center of the bed or wheelchair. Since the human model was always perceived to be in the same place, a function that maps observations to robot configurations was not required. The simulated PR2 was moved to the chosen base configuration and error was added to its position and orientation. From that pose, OpenRAVE attempted to find a collision-free IK solution to each goal pose defined with respect to the true pose of the human model's body.

The goals are sparse, so each goal is important to a task. We considered a trial successful if, from the robot configurations selected by the method, OpenRAVE could find a collision-free IK solution to all goal poses despite the error introduced in the Monte Carlo simulation. Otherwise, the trial was deemed a failure. We performed this evaluation for all 9 modeled tasks in both environments, except for the cleaning legs tasks with the wheelchair environment, since the table blocks access to the human model's thighs.

All introduced error was normally distributed around 0. For the robotic bed environment, the standard deviation for the human's pose was 2.5 cm translation in the global X direction and 5.0 cm translation in the global Y direction. These axes match those shown in Fig. 3. We used smaller X error because for larger negative X error the human model would penetrate the backrest of the bed and some goals would become unreachable. Rotations of the human in bed were not considered.

For the wheelchair environment, the standard deviation for the human's pose was 2.5 cm translation in the global X direction, 5.0 cm translation in the global Y direction, and 5° rotation about the human head's Z axis. These axes match those shown in Fig. 12. We used smaller X error because for larger negative X error the human model would penetrate the backrest of the bed and some goals would become unreachable. The standard deviation for the PR2's position was 1.0 cm in the global X and Y directions and 5° rotation about the robot's Z axis. We selected these error distributions from typical error in human pose estimation and PR2 servoing in our previous work (Kapusta et al. 2016). As described in Sects. 3.5 and 3.6, models used while optimizing robot configurations had a small safety margin of $\sim$ 3 cm. Models used during testing had no safety margin.

To more fairly compare the methods, we used the same seeds for all of the CMA-ES optimizations and the Monte-Carlo simulations. For the CMA-ES optimization, all methods were given a population size of 40, a maximum number of iterations of 1000, and the opportunity to restart with double the population if the optimization ran out of iterations before converging within some tolerance. All methods also had the same heuristics for driving the search towards configurations that may have collision-free IK solutions. Additionally, all methods used the same IK solver from OpenRAVE using the same default parameters. The dimensionality of the search space for TOC was 12 (6-DoF $\times$ 2 configurations) and 8 (4-DoF $\times$ 2 configurations) for the bed and the wheelchair environment, respectively. The dimensionality of the search space for the baseline algorithms was 6 (6-DoF $\times$ 1 configuration) and 4 (4-DoF $\times$ 1 configuration) for the bed and the wheelchair environment, respectively.

We assigned appropriate bounds on parameters based on the environment (e.g., slightly beyond reach of the bed). We initialized the parameters to aid coverage in the search, giving two initial locations, one position on one side of the bed or wheelchair and one on the other side. Baseline methods were allowed two searches, one for each initialization, and we used the single best configuration. TOC jointly optimized its two configurations from their respective initialization locations. Thus, all methods were given comparable initializations and bounds.

### 4.2.1 Baselines

To compare against TOC, we implemented three baseline algorithms based on state-of-the-art methods from the literature, one based on IK and two based on the robot capability map. An overview of these and other related methods from the literature can be found in Sect. 2. These methods used optimization to select a single configuration for both the PR2 and the robotic bed and made use of the human model's free parameter (head rotation) in the wheelchair environment. The main way in which the baseline methods differed from TOC are their objective functions and their use of only a single robot configuration.

**Inverse-kinematics (IK) solver-based baseline** IK solver-based methods to select a robot base pose for a task are common in the literature. The method we implemented used CMA-ES to search for a robot configuration from which the robot has a collision-free IK solution to all goal end-effector poses. We used the IKFast module within the OpenRAVE simulation environment to determine if a collision-free IK solution existed for each robot configuration.

**Capability map-based baselines** Various methods from the literature use the capability map Zacharias et al. (2007). We implemented two baseline methods roughly based on Zacharias et al. (2009). For these methods we first created a capability map using OpenRAVE's kinematic reachability module using its default parameters (OpenRAVE 2018). To create the capability map, the module discretized 3D space around the robot's arm into 3D points and discretized the range of possible orientations around each 3D point. The map was created using OpenRAVE's default parameters, discretizing 3D space into 4 cm intervals and sampling orientation at an average quaternion distance of 0.5. The capability score (also known as reachability score) for each point is the percentage of orientations for which the robot has a valid IK solution. These scores are calculated offline and saved. These two methods use CMA-ES to search for a robot configuration that maximizes the average capability score for all goal poses. The score of a goal pose is the score of the closest 3D point from the capability map. The first capability map-based baseline considered capability scores without regard to the environment. The second gave goal poses a 0 score if a collision-free IK solution could not be found to that pose in the environment. Although the method with collision-checking might be expected to dominate the method without collision checking, performance can vary due to random sampling in the capability map, in the optimization algorithm (CMA-ES), and in the Monte Carlo simulation.

#### 4.2.2 Results

The results for each task for the robotic bed and wheelchair are shown in Figs. 8 and 9, respectively. TOC's average success rate was higher than or comparable to baseline methods in all tasks. Numbers in boldface were significantly different from the TOC result ($p < 0.01$ in a Wilcoxon Rank-Sum test).

TOC had an overall average success rate of 90.6%, compared to 50.4% for the IK solver, 43.5% for the capability map, and 58.9% for the capability map with collision checking. The overall differences between the baseline results and TOC were statistically significant ($p < 0.01$ in a Wilcoxon Rank-Sum test). TOC chose to use a single configuration for all tasks other than the shaving and cleaning arms tasks in this environment in the bed environment and it used a single configuration for all but the cleaning arms task in the wheelchair environment.

TOC achieved significantly higher success rates ($p < 0.01$ in a Wilcoxon Rank-Sum test) for some tasks for which it used only one configuration and for tasks for which it used two configurations. This result suggests that the benefit from TOC comes from more than using two configurations over one. For tasks that seemed to require more than one robot configuration, baseline methods failed; they could only select a single configuration. TOC jointly optimizes 2 configurations, allowing it to succeed in these challenging tasks.

### 4.3 Quantifying robustness

In Figs. 10, 11 and 12 We visualize the robustness of robot configurations selected by TOC for the shaving and cleaning arms tasks in the two environments. These figures show the percentage of goal poses that have collision-free IK solutions (indicated by the color) for varying error in the human model's position on the bed or wheelchair (the X–Y axes). The goal poses are update to the true human model's position. Notable in these figures is the success region in blue, where all goals are reachable, as well as how the two configurations combine to reach all goals. For pose estimation error in the success region, the PR2 would still be able to successfully perform the task. TOC opted to use two configurations for each of the tasks shown. The success region is large and surrounds the origin for shaving and cleaning arms in bed, which is why 100% of the trials were successful for these tasks in Fig. 8. The success region is less centered around the origin for the cleaning arms task in the wheelchair, hence its lower percentage of successful trials in Fig. 9. The Monte Carlo simulations randomly sampled in these, as well as other, degrees of freedom and sampling outside the success region results in a failed trial. These figures suggest that the task may be easier for the robot to perform for a person in bed. Closer observation of the task shows that, because the wheelchair is tall, the goal poses for the cleaning arms task are vertically higher in the PR2's workspace and the arm has relatively low JLWKI when reaching those goals.

### 4.4 Evaluation of TOC objective function

TOC searches for a set of robot configurations that maximizes its objective function, which we will call its score for simplicity. The assumption therein is that higher values of the score are correlated with better robot configurations that are more robust to error. To test this assumption, we evaluated the relationship between the TOC score and the accuracy (the percentage of goals that are reachable) for robot configurations in the same evaluation described in Sect. 4.2. Note that we chose to compare against accuracy in this evaluation because it can convey more information than success, which is binary. A similar correlation can be seen for success. For the wiping mouth task in the robotic bed, we sampled robot configurations with TC-reachability of 1 (i.e., all goal poses have collision-free IK solutions) and compared their mean and variance in accuracy over 200 Monte Carlo simulations with their TOC score. Figure 13 shows the results of the analysis. Higher TOC score is correlated with accuracy and inversely correlated with variance in accuracy.

## 5 Discussion

In our evaluation, we used simulations to provide empirical evidence that TOC outperforms baseline methods based on state-of-the-art methods from the literature in the context of robotic assistance with ADLs. Each trial was only considered successful if IK solutions were found for all of the end-effector poses associated with a task. Consequently, a failure implied that a simulated robot would fail to find trajectories to reach all the locations and also makes it unlikely that a real robot would find trajectories to reach all the locations. The main weakness of our model for evaluation is that a simulated or real robot might not be able to find a trajectory to an end-effector pose for which an IK solution exists. So,

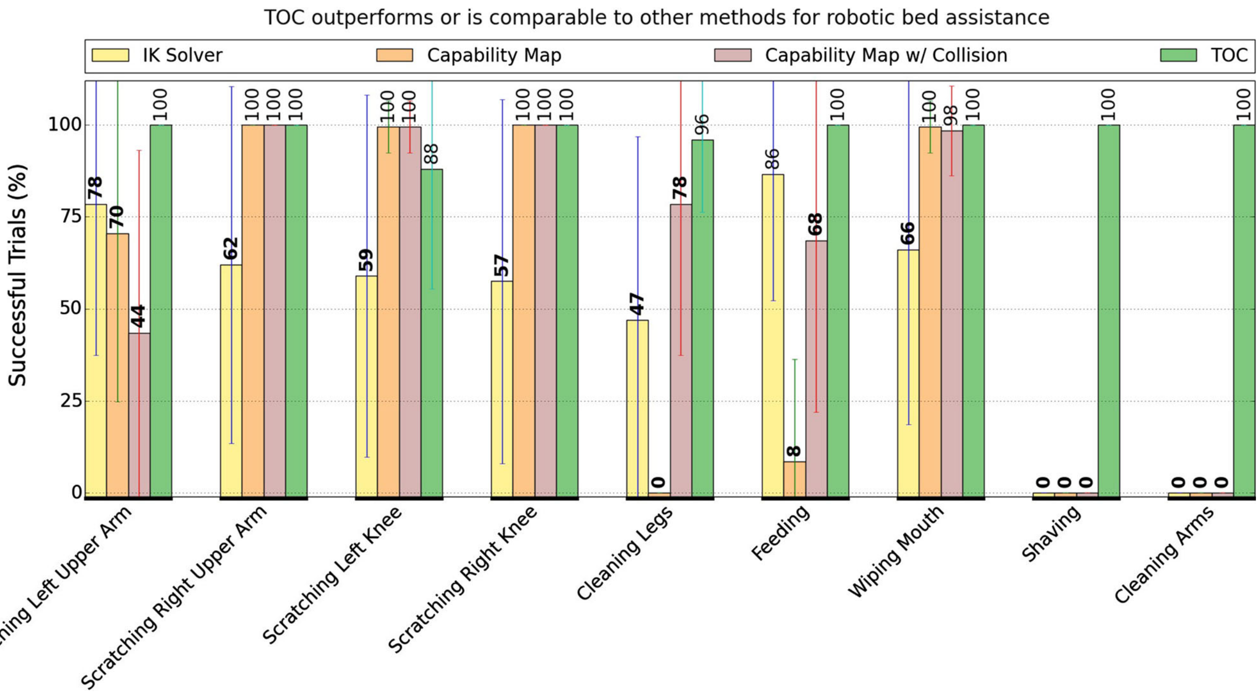


**Fig. 8** Comparison of performance between TOC and three baseline methods averaged over 200 Monte-Carlo simulations of error in the human model's position and the robot's base pose for tasks in the robotic bed environment. Numbers in boldface were significantly different from the TOC result ($p < 0.01$ in a Wilcoxon Rank-Sum test). Error bars show one standard deviation. TOC chose to use a single configuration for all tasks other than the shaving and cleaning arms tasks in this environment. Baseline methods could only select a single configuration

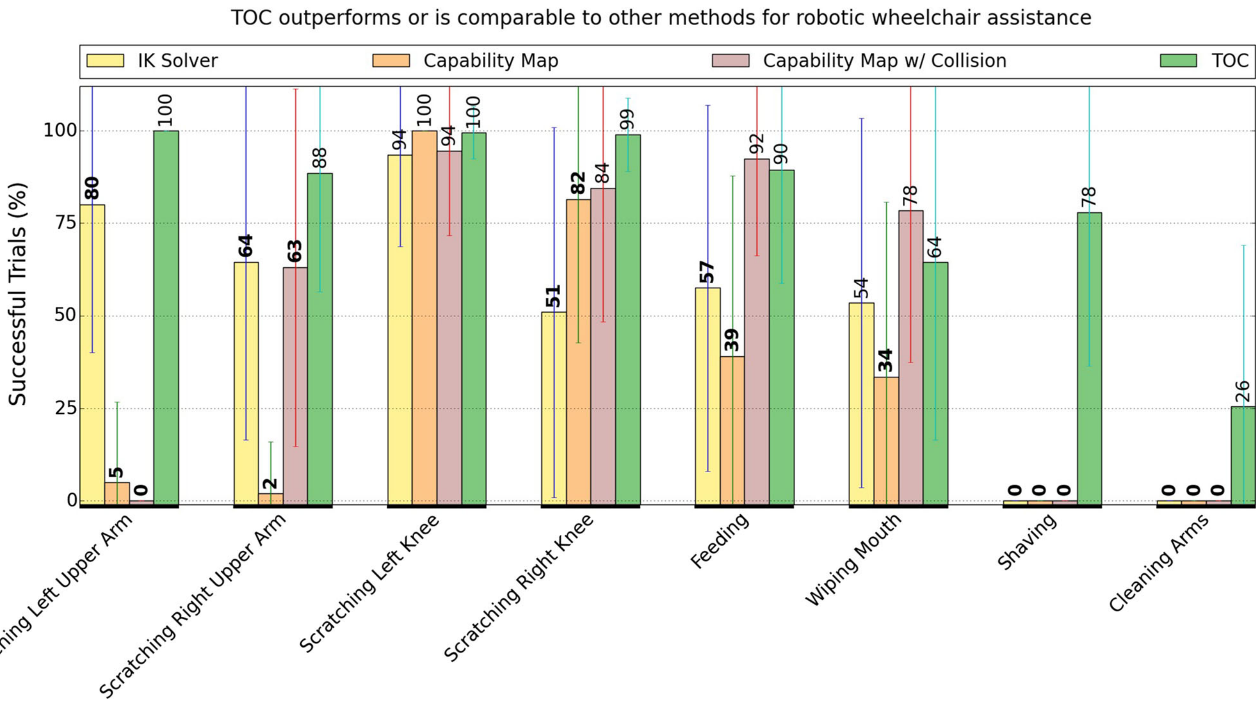


**Fig. 9** Comparison of performance between TOC and three baseline methods averaged over 200 Monte-Carlo simulations of error in the human model's position and the robot's base pose for tasks in the wheelchair environment. Numbers in boldface were significantly different from the TOC result ($p < 0.01$ in a Wilcoxon Rank-Sum test). Error bars show one standard deviation. TOC chose to use a single configuration for all tasks other than the cleaning arms task in this environment. Baseline methods could only select a single configuration

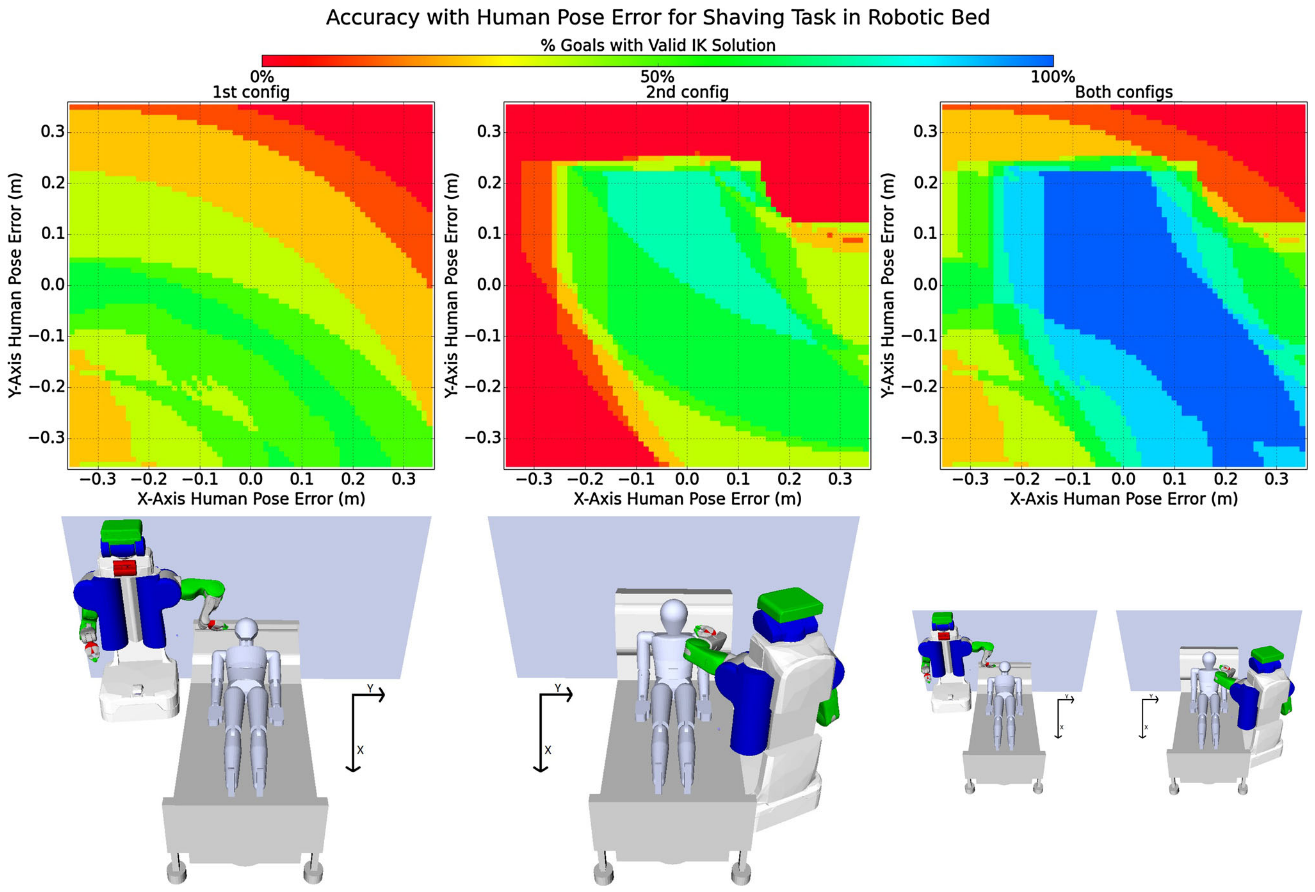


**Fig. 10** Visualization of the robustness of TOC's selected configurations for the shaving task in the robotic bed. (Top) Percentage of goals reached from the first, second, and both configurations for error in 1 cm increments in the x–y position of the human model. (Bottom) The first and second configurations of the PR2, and the two configurations combined on the right. Color is necessary to interpret this figure. The blue region represents when all goals can be reached (Color figure online)

while a failure strongly implies a failure in reality, a success only implies a high likelihood of success in reality.

One can construct situations that would result in the robot not being able to reach IK solutions. One might also be able to construct a contrived situation that would unfairly bias the results in favor of TOC via IK solutions that are unreachable. However, the nature of ADLs makes this type of bias in our results unlikely due to the geometric structure of the tasks. When the human body is sitting or reclining on a wheelchair or bed, there is typically substantial free space above the body and furniture permitting solutions that involve the arm reaching from above or reaching out directly to obtain end-effector poses with associated IK solutions.

When working with real robots in the context of providing assistance with ADLs, as in Hawkins et al. (2014) and Grice and Kemp (2018), we have encountered challenges when attempting to find poses for the mobile base by trial and error. Consequently, we pursued research that led to TOC, which addresses this issue. In contrast, we have not encountered difficulties when selecting an initial configuration of the robot's arm, and we chose not to address optimization of the arms initial configuration. Incorporating the robot arm's initial configuration and tests for the existence of trajectories with which to reach IK solutions would be possibilities for future work, although in the context of ADLs, the additional computation involved might not be justified.

## 5.1 Limitations

Although the framework of TOC allows it to jointly optimize more than two robot configurations, we limited our evaluation to two configurations. We made this choice because we found that more than two robot configurations were not needed for any of the tasks. Jointly optimizing more configurations would increase TOC's computational requirements, since the dimensionality of the search space is $m \times n$, where $m$ is number of configurations and $n$ is the number of degrees of freedom. For our bed environment $m = 2$ and $n = 6$. For our wheelchair environment $m = 2$ and $n = 4$.

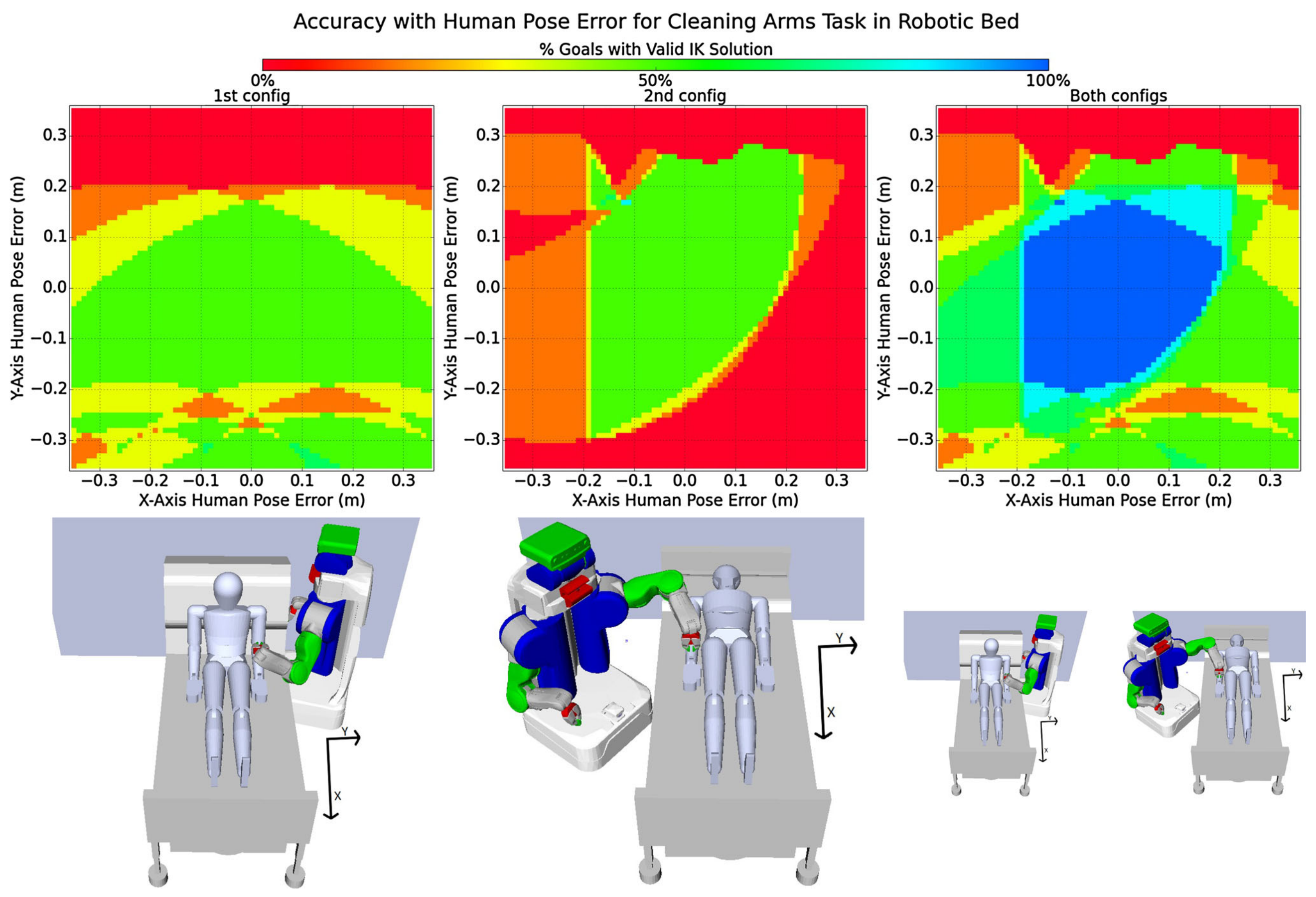


**Fig. 11** Visualization of the robustness of TOC's selected configurations for the arm cleaning task in the robotic bed. (Top) Percentage of goals reached from the first, second, and both configurations for error in 1 cm increments in the x–y position of the human model. (Bottom) The first and second configurations of the PR2, and the two configurations combined on the right. Color is necessary to interpret this figure. The blue region represents when all goals can be reached (Color figure online)

We hand designed the task models for our evaluation in simulation based on our real-world experience with assistive robots. While we expect them to be sufficient to support real-world assistance, we have not formally evaluated them. We modeled all of the tasks using sets of 6-DoF goal poses, yet some tasks, such as a sponge bath, could be less sensitive to the end effector's orientation.

Although we only implemented and evaluated TOC on a PR2 in simulation, we expect that the method may be transferable and generalizable to other robots. A feature of kinematic isotropy, used in TOC's objective function, is that it is independent of scale and order. $R_{\text{reach}}$ and $R_{\text{manip}}$ should remain between 0 and 1 for all robots. As a result, the values of the terms in the objective function may be used on new robots without modification.

## 5.2 Relevance to other tasks

We specifically developed TOC to support robotic assistance with ADLs. We expect that it could be useful for other types of tasks, but the extent to which it would be suitable for other domains is an open question. Also, TOC is not well suited for all ADLs. For example, it is unclear how one would model some dressing tasks using a set of end-effector poses with respect to the person's body. TOC and this type of model might be useful for assistance with a small article of somewhat rigid clothing like a baseball cap or a slipper. However, this type of model appears insufficient for active assistance with a larger article of compliant clothing, such as a sweatshirt, which depends on end-effector trajectories and the physics of garments interacting with the human body.

## 5.3 Collisions

Joint-limit-weighted kinematic isotropy, the foundation of our TC-manipulabilty score, does not account for environmental constraints. There may be value in preferring joint configurations away from obstacles. Explicitly penalizing proximity to collisions in the objective function, as was done by Vahrenkamp et al. (2012), might be a way to mitigate this

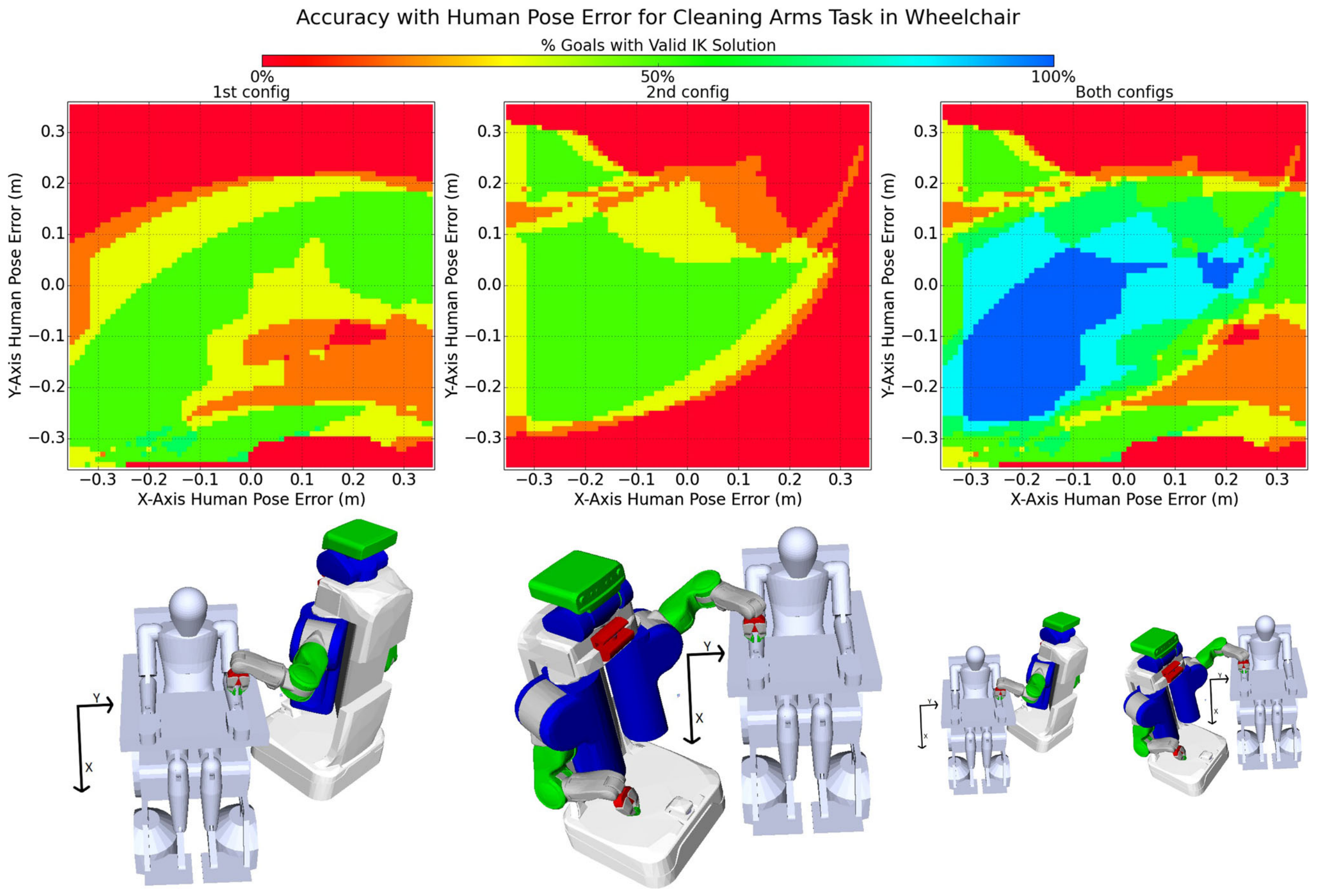


**Fig. 12** Visualization of the robustness of TOC's selected configurations for the arm cleaning task in the wheelchair. (Top) Percentage of goals reached from the first, second, and both configurations for error in 1 cm increments in the x–y position of the human model. (Bottom) The first and second configurations of the PR2, and the two configurations combined on the right. Color is necessary to interpret this figure. The blue region represents when all goals can be reached (Color figure online)

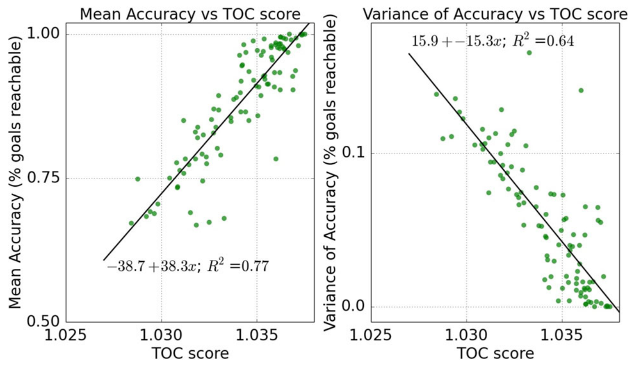


**Fig. 13** The TOC score is positively correlated with accuracy and negatively correlated with variance in accuracy. For this figure, collision-free IK solutions were found to all goals results in TOC scores above 1.0. The amount above 1.0 is the weighted TC-manipulability score for the configuration

issue at the cost of additional computation time. More generally, the inclusion of additional terms in TOC's objective function, such as to represent distance from obstacles, proxemics, or user preference, should be possible and may merit future work.

In our evaluation, TOC's offline optimization mitigated the risk of collisions by using a safety margin on the environment and user models. This comes at the cost of decreasing the valid search space and eliminating valid solutions. For example, Grice et al. (2013) found that contact can be both beneficial and acceptable during robotic assistance. Allowing contact can increase the space of reachable poses, and there are methods for controlling contact safely (Killpack et al. 2016).

## 5.4 Computational efficiency

On average, TOC took 70 minutes, the IK solver baseline took 6 minutes, the capability map took 1.5 minutes, and the capability map with collision took 34 minutes on average. All computations were performed using a single thread on a 64-bit, 14.04 Ubuntu operating system with 8 GB of RAM and a 3.40 GHz Intel Core i7-3770 CPU. These particular durations would all be excessively long for a robotic system intended to provide assistance with ADLs. With more efficient software and improved hardware, the IK solver could potentially be fast enough for online use, although its performance in our tests was relatively poor. The other methods require substantially more computation, but perform significantly better. All of the baseline methods could be used with the TOC framework to generate training pairs offline in order to generate an approximate function for efficient online use. Also, all algorithms in our evaluation used CMA-ES to optimize the robot configurations. Other optimization algorithms could result in different performance and timing.

### 5.5 Design application

TOC could potentially be used to assist in the design of environments. For example, in our evaluations with the robotic bed, TOC selected a configuration for the bed by including the bed's DoF in the robot configuration. Although we considered the bed to be a robotic bed under control of the robot, one could instead allow TOC to control features of the environment that can be altered by humans, such as the placement of the bed in the room. For example, we have found that for the shaving task, if there is sufficient space behind the bed, the PR2 can perform the task from a single configuration instead of requiring two configurations. Figure 14 shows the configuration found by TOC for shaving in the robotic bed without a wall and visualizes the robustness to error in the pose of the human model.

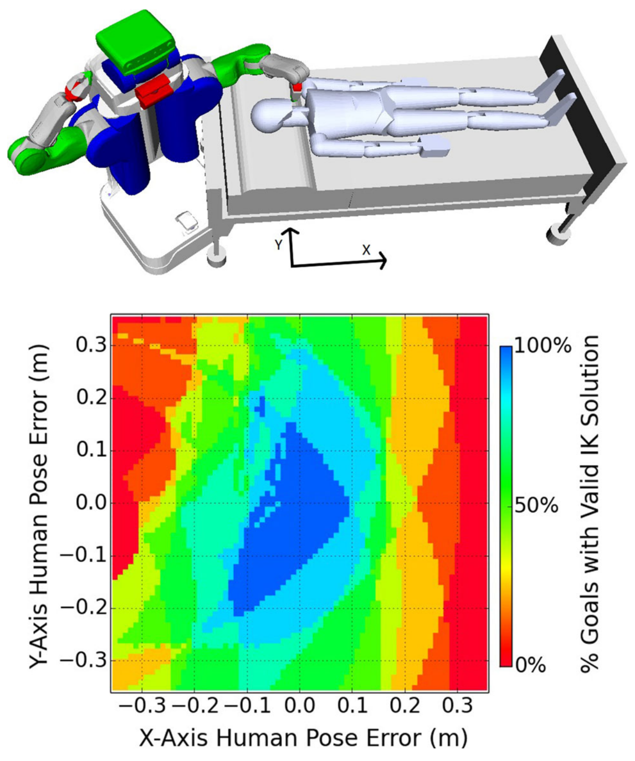


**Fig. 14** The PR2 can perform the shaving task on the human model in bed from a single location if there is no wall behind the bed. (Top) the configuration TOC selected (bottom) a visualization of the robustness to error in the human model's pose on the bed for this configuration

## 6 Conclusion

In this work, we have presented task-centric optimization of robot configurations (TOC), a method to select one or two configurations for robots to assist with tasks around a person's body. TOC uses TC-reachability and TC-manipulability, metrics that we have developed, to represent the robot's dexterity, and implicitly handle error. TOC is particularly suitable for robotic assistance with activities of daily living (ADLs) that can be well-modeled as a set of end-effector poses attached to the human body. We have shown that TOC can determine a set of one or two robot configurations from which the robot can perform a task well. TOC performs the bulk of its computation offline using models of the task, robot, environment, and person to generate a function that rapidly ($\leq$ 1 s) estimates the optimal set of robot configurations for a task given observations at runtime. We provide evidence that configurations selected by TOC are robust to errors between the models used offline and observations at run time. We created 9 models of assistive tasks to test our system in simulation and showed that for each task TOC's average success rate was higher than or comparable to three baseline algorithms based on state-of-the-art methods from the literature. TOC had an overall average success rate of 90.6% compared to 50.4%, 43.5%, and 58.9% for the baseline methods.

## References

Andersen, C. K., Wittrup-Jensen, K. U., Lolk, A., Andersen, K., & Kragh-Sørensen, P. (2004). Ability to perform activities of daily living is the main factor affecting quality of life in patients with dementia. *Health and Quality of Life Outcomes*, *2*(1), 52.

Beeson, P., & Ames, B. (2015). Trac-ik: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International conference on humanoid robots (humanoids)* (pp. 928–935). IEEE.

Bien, Z., Chung, M. J., Chang, P. H., Kwon, D. S., Kim, D. J., Han, J. S., et al. (2004). Integration of a rehabilitation robotic system (KARES II) with human-friendly man–machine interaction units. *Autonomous Robots*, *16*(2), 165–191.

Burget, F., & Bennewitz, M. (2015). Stance selection for humanoid grasping tasks by inverse reachability maps. In *2015 IEEE International conference on robotics and automation (ICRA)* (pp. 5669–5674). IEEE.

Canadian Partnership for Stroke Recovery. (2015). *Stroke Engine*. http://www.strokengine.ca/. Accessed 01 Jan 2018.

Chan, T. F., & Dubey, R. V. (1995). A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Transactions on Robotics and Automation*, *11*(2), 286–292.

Chen, T. L., Ciocarlie, M., Cousins, S., Grice, P. M., Hawkins, K., et al. (2013). Robots for humanity: A case study in assistive mobile manipulation. *IEEE Robotics & Automation Magazine, Special issue on Assistive Robotics*, *20*(1), 30–39.

CMA. (2018). *The CMA evolution strategy*. https://pypi.python.org/pypi/cma. Accessed 01 Jan 2018.

Dario, P., Guglielmelli, E., Laschi, C., & Teti, G. (1999). MOVAID: A personal robot in everyday life of disabled and elderly people. *Technology and Disability*, *10*(2), 77–93.

Diankov, R. (2010). *Automated construction of robotic manipulation programs*. PhD thesis, Carnegie Mellon University.

Diankov, R., & Kuffner, J. (2008). *Openrave: A planning architecture for autonomous robotics*. Robotics Institute, Pittsburgh, PA, Tech Rep CMU-RI-TR-08-34 79.

Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S., & Kuffner, J. (2008). Bispace planning: Concurrent multi-space exploration. In *Proceedings of robotics: Science and systems IV* (Vol. 63).

Dong, J., & Trinkle, J. C. (2015). Orientation-based reachability map for robot base placement. In *2015 IEEE/RSJ International conference on intelligent robots and systems (IROS)* (pp. 1488–1493). IEEE.

Elbanhawi, M., & Simic, M. (2014). Sampling-based robot motion planning: A review. *IEEE Access*, *2*, 56–77.

Garrett, C. R., Lozano-Pérez, T., & Kaelbling, L. P. (2015). Backward–forward search for manipulation planning. In *2015 IEEE/RSJ International conference on intelligent robots and systems (IROS)* (pp. 6366–6373). IEEE.

Gienger, M., Janssen, H., & Goerick, C. (2005). Task-oriented whole body motion for humanoid robots. In *2005 5th IEEE-RAS International conference on humanoid robots* (pp. 238–244). IEEE.

Graf, B., Reiser, U., Hägele, M., Mauz, K., & Klein, P. (2009). Robotic home assistant Care-O-bot®3-product vision and innovation platform. In *2009 IEEE Workshop on advanced robotics and its social impacts (ARSO)* (pp. 139–144). IEEE.

Grey, M. X., Garrett, C. R., Liu, C. K., Ames, A. D., & Thomaz, A. L. (2016). Humanoid manipulation planning using backward–forward search. In *2016 IEEE/RSJ International conference on intelligent robots and systems (IROS)* (pp. 5467–5473). IEEE.

Grice, P., Chitalia, Y., & Rich, M., et al. (2016). Autobed: Open hardware for accessible web-based control of an electric bed. In *RESNA*.

Grice, P., & Kemp, C. C. (2016). Assistive mobile manipulation: Designing for operators with motor impairments. In *Proceedings of robotics: Science and systems (RSS 2016) workshop on socially and physically assistive robotics for humanity*.

Grice, P. M., & Kemp, C. C. (2018). *In-home and remote use of robotic body surrogates by people with profound motor deficits*. arXiv preprint arXiv:1803.01477.

Grice, P. M., Killpack, M. D., Jain, A., Vaish, S., Hawke, J., & Kemp, C. C. (2013). Whole-arm tactile sensing for beneficial and acceptable contact during robotic assistance. In *2013 IEEE International conference on rehabilitation robotics (ICORR)* (pp. 1–8). IEEE.

Hammond, F., & Shimada, K. (2011). Multi-objective weighted isotropy measures for the morphological design optimisation of kinematically redundant manipulators. *International Journal of Mechatronics and Manufacturing Systems*, *4*(2), 150–170.

Hammond III, F. L., & Shimada, K. (2009). Improvement of kinematically redundant manipulator design and placement using torque-weighted isotropy measures. In *2009 International conference on advanced robotics, ICAR 2009* (pp. 1–8). IEEE.

Hansen, N. (2006). The CMA evolution strategy: A comparing review. In *Towards a new evolutionary computation* (pp. 75–102). Springer.

Hawkins, K. P., Grice, P. M., & Chen, T. L., et al. (2014). Assistive mobile manipulation for self-care tasks around the head. In *2014 IEEE Symposium on computational intelligence in robotic rehabilitation and assistive technologies (CIR2AT)* (pp. 16–25). IEEE.

Hsu, D., Latcombe, J. C., & Sorkin, S. (1999). Placing a robot manipulator amid obstacles for optimized execution. In *Proceedings of the 1999 IEEE international symposium on assembly and task planning (ISATP'99)* (pp. 280–285). IEEE.

Institute of Medicine. (2008). Retooling for an aging America: Building the health care workforce. Washington, DC: The National Academies Press.

Jain, A., & Kemp, C. C. (2010). EL-E: An assistive mobile manipulator that autonomously fetches objects from flat surfaces. *Autonomous Robots*, *28*(1), 45–64.

Janiszewski Goodin, H. (2003). The nursing shortage in the United States of America: An integrative review of the literature. *Journal of Advanced Nursing*, *43*(4), 335–343.

Kapusta, A., Chitalia, Y., Park, D., & Kemp, C. C. (2016). Collaboration between a robotic bed and a mobile manipulator may improve physical assistance for people with disabilities.

Kapusta, A., & Kemp, C. C. (2016). Optimization of robot configurations for assistive tasks. In *Proceedings of robotics: Science and systems (RSS 2016) workshop on planning for human–robot interaction: Shared autonomy and collaborative robotics*.

Kapusta, A., Park, D., & Kemp, C. C. (2015). Task-centric selection of robot and environment initial configurations for assistive tasks. In *2015 IEEE/RSJ International conference on intelligent robots and systems (IROS)* (pp. 1480–1487). IEEE.

Killpack, M. D., Kapusta, A., & Kemp, C. C. (2016). Model predictive control for fast reaching in clutter. *Autonomous Robots*, *40*(3), 537–560.

Kim, J. O., & Khosla, K. (1991). Dexterity measures for design and control of manipulators. In *IEEE/RSJ International workshop on intelligent robots and systems' 91.'Intelligence for mechanical systems, proceedings IROS'91* (pp. 758–763). IEEE.

Klein, C. A., & Blaho, B. E. (1987). Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, *6*(2), 72–83.

Kruse, T., Pandey, A. K., Alami, R., & Kirsch, A. (2013). Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, *61*(12), 1726–1743.

Kumar, S., Sukavanam, N., & Balasubramanian, R. (2010). An optimization approach to solve the inverse kinematics of redundant manipulator. *International Journal of Information and System Sciences (Institute for Scientific Computing and Information)*, *6*(4), 414–423.

Lee, J. (1997). A study on the manipulability measures for robot manipulators. In *Proceedings of the 1997 IEEE/RSJ international conference on intelligent robots and systems, IROS'97* (Vol. 3, pp. 1458–1465). IEEE.

Leidner, D., Dietrich, A., Schmidt, F., Borst, C., & Albu-Schaffer, A. (2014). Object-centered hybrid reasoning for whole-body mobile manipulation. In *2014 IEEE International conference on robotics and automation (ICRA)* (pp. 1828–1835). IEEE.

Lindemann, S. R., & LaValle, S. M. (2005). Current issues in sampling-based motion planning. In *Robotics research* (pp. 36–54).

Mainprice, J., Sisbot, E. A., Jaillet, L., Cortes, J., Alami, R., & Simeon, T. (2011). Planning human-aware motions using a sampling-based costmap planner. In *2011 IEEE International conference on robotics and automation (ICRA)* (pp. 5012–5017). IEEE.

Mumm, J., & Mutlu, B. (2011). Human–robot proxemics: Physical and psychological distancing in human–robot interaction. In *Pro-*

*ceedings of the 6th international conference on Human–robot interaction* (pp. 331–338). ACM.

OpenRAVE. (2018). *Open robotics automation virtual environment (OpenRAVE)*. http://openrave.org. Accessed 01 Jan 2018.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Porges, O., Stouraitis, T., Borst, C., & Roa, M. A. (2014). Reachability and capability analysis for manipulation tasks. In *ROBOT2013: First Iberian robotics conference* (pp. 703–718). Springer.

Schaeffer, C., & May, T. (1999). Care-o-bot-a system for assisting elderly or disabled persons in home environments. In *Assistive Technology on the Threshold of the New Millenium* (pp. 340–345). IOS Press.

Sisbot, E. A., Marin, L. F., Alami, R., & Simeon, T. (2006). A mobile robot that performs human acceptable motions. In *2006 IEEE/RSJ International conference on intelligent robots and systems* (pp. 1811–1816). IEEE.

Sisbot, E. A., Marin-Urias, L. F., Alami, R., & Simeon, T. (2007). A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, *23*(5), 874–883.

Sisbot, E. A., Marin-Urias, L. F., Broquere, X., Sidobre, D., & Alami, R. (2010). Synthesizing robot motions adapted to human presence. *International Journal of Social Robotics*, *2*(3), 329–343.

Smits, R. (2006). *KDL: Kinematics and dynamics library*. http://www.orocos.org/kdl. Accessed 01 Jan 2018.

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control* (Vol. 3). New York: Wiley.

Stilman, M., & Kuffner, J. J. (2005). Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, *2*(04), 479–503.

Stocco, L., Salcudean, S., & Sassani, F. (1998). Matrix normalization for optimal robot design. In *1998 IEEE International conference on robotics and automation. Proceedings* (Vol. 2, pp. 1346–1351). IEEE.

Stulp, F., Fedrizzi, A., & Beetz, M. (2009). Learning and performing place-based mobile manipulation. In *2009 IEEE 8th International conference on development and learning, ICDL 2009* (pp. 1–7). IEEE.

Takayama, L., & Pantofaru, C. (2009). Influences on proxemic behaviors in human–robot interaction. In *2009 IEEE/RSJ International conference on intelligent robots and systems, IROS 2009* (pp. 5495–5502). IEEE.

Tan, J., Gu, Y., Turk, G., & Liu, C. K. (2011). Articulated swimming creatures. *ACM Transactions on Graphics (TOG)*, *30*, 58.

The Wright Stuff, Inc. (2017) . *The Wright Stuff healthcare products that make life easier*. https://www.thewrightstuff.com/. Accessed 01 Jan 2018.

Tilley, A. R. (2002). *The measure of man and woman: Human factors in design*. New York: Wiley.

Topping, M. (1999). The development of handy 1. A robotic system to assist the severely disabled. In *International conference on rehabilitation robotics* (pp. 244–249)

Topping, M., & Smith, J. (1998). The development of Handy 1, a rehabilitation robotic system to assist the severely disabled. *Industrial Robot*, *25*(5), 316–20.

Tsai, M. J. (1986). *Workspace geometric characterization and manipulability of industrial robots*. PhD thesis, The Ohio State University.

Vahrenkamp, N., Asfour, T., & Dillmann, R. (2013). Robot placement based on reachability inversion. In *2013 IEEE International conference on robotics and automation (ICRA)* (pp. 1970–1975). IEEE.

Vahrenkamp, N., Asfour, T., Metta, G., Sandini, G., & Dillmann, R. (2012). Manipulability analysis. In *2012 12th IEEE-RAS International conference on humanoid robots (humanoids)* (pp. 568–573). IEEE.

Vest, M. T., Murphy, T. E., Araujo, K. L., Pisani, M. A., et al. (2011). Disability in activities of daily living, depression, and quality of life among older medical ICU survivors: A prospective cohort study. *Health and Quality of Life Outcomes*, *9*, 9. https://doi.org/10.1186/1477-7525-9-9.

Walters, M. L., Dautenhahn, K., Te Boekhorst, R., Koay, K. L., Kaouri, C., Woods, S., et al. (2005). The influence of subjects personality traits on personal spatial zones in a human–robot interaction experiment. In *2005 IEEE International workshop on robot and human interactive communication, ROMAN 2005* (pp. 347–352). IEEE.

Walters, M. L., Dautenhahn, K., Te Boekhorst, R., Koay, K. L., Syrdal, D. S., & Nehaniv, C. L. (2009). An empirical framework for human–robot proxemics. In *Procs of new frontiers in human–robot interaction*.

Walters, M. L., Oskoei, M. A., Syrdal, D. S., & Dautenhahn, K. (2011). A long-term human–robot proxemic study. In *2011 IEEE RO-MAN* (pp. 137–142). IEEE.

Wiener, J. M., Hanley, R. J., Clark, R., & Nostrand, J. F. V. (1990). Measuring the activities of daily living: Comparisons across national surveys. *Journal of Gerontology: Social Sciences*, *45*(6), S229–237.

Yoshikawa, T. (1984). Analysis and control of robot manipulators with redundancy. In *Robotics research: The first international symposium* (pp. 735–747). Cambridge, MA: MIT Press.

Zacharias, F., Borst, C., & Hirzinger, G. (2007). Capturing robot workspace structure: representing robot capabilities. In *2007 IEEE/RSJ International conference on intelligent robots and systems, IROS 2007* (pp. 3229–3236). IEEE.

Zacharias, F., Sepp, W., Borst, C., & Hirzinger, G. (2009). Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories. In *2009 9th IEEE-RAS International conference on humanoid robots, humanoids 2009* (pp. 55–61). IEEE.

**Ariel Kapusta** received his Ph.D. in robotics from the Georgia Institute of Technology in 2018 through his research in the Healthcare Robotics Lab. He earned a B.S. degree in Mechanical Engineering from Rice University in 2009. His research interests include assistive robotics, mobile manipulation, control systems for robotic manipulation, human–robot interaction, multi-modal sensing, and machine learning.

**Charles C. Kemp** is an Associate Professor at the Georgia Institute of Technology in the Department of Biomedical Engineering with adjunct appointments in the School of Interactive Computing and the School of Electrical and Computer Engineering. He earned a doctorate in Electrical Engineering and Computer Science (2005), an M.Eng., and B.S. from MIT. In 2007, he joined the faculty at Georgia Tech where he directs the Healthcare Robotics Lab (http://healthcare-robotics.com). His research has primarily been in the areas of mobile manipulation, human–robot interaction, assistive robotics, and haptic sensing. He is a member of Georgia Tech's Institute for Robotics and Intelligent Machines (IRIM) and its multidisciplinary Robotics Ph.D. program. He has received a 3M Non-tenured Faculty Award, the Georgia Tech Research Corporation Robotics Award, a Google Faculty Research Award, and an NSF CAREER award. He was a Hesburgh Award Teaching Fellow in 2017 and received the Class of 1940 Course Survey Teaching Effectiveness Award in 2018.