Auton Robot (2016) 40:537–560
DOI 10.1007/s10514-015-9492-6

# Model predictive control for fast reaching in clutter

**Marc D. Killpack**[1,2] · **Ariel Kapusta**[1] · **Charles C. Kemp**[1]

Received: 20 October 2014 / Accepted: 18 July 2015 / Published online: 25 September 2015


**Abstract** A key challenge for haptically reaching in dense clutter is the frequent contact that can occur between the robot's arm and the environment. We have previously used single-time-step model predictive control (MPC) to enable a robot to slowly reach into dense clutter using a quasistatic mechanical model. Rapid reaching in clutter would be desirable, but entails additional challenges due to dynamic phenomena that can lead to higher forces from impacts and other types of contact. In this paper, we present a multi-time-step MPC formulation that enables a robot to rapidly reach a target position in dense clutter, while regulating whole-body contact forces to be below a given threshold. Our controller models the dynamics of the arm in contact with the environment in order to predict how contact forces will change and how the robot's end effector will move. It also models how joint velocities will influence potential impact forces. At each time step, our controller uses linear models to generate a convex optimization problem that it can solve efficiently. Through tens of thousands of trials in simulation, we show that with our dynamic MPC a simulated robot can, on average, reach goals 1.4 to 2 times faster than our previous controller, while attaining comparable success rates and fewer occurrences of high forces. We also conducted trials using a real 7 degree-of-freedom (DoF) humanoid robot arm with whole-arm tactile sensing. Our controller enabled the robot to rapidly reach target positions in dense artificial foliage while keeping contact forces low.

**Keywords** Clutter · Haptic · MPC · Multi-contact

## List of symbols

| | |
|---|---|
| $\boldsymbol{A}_d$, $\boldsymbol{B}_d$ | State-space matrices that are discrete time linear approximations of the dynamics for a robot in contact |
| $\boldsymbol{C}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ | Coriolis and centrifugal matrix |
| $\boldsymbol{d}_{grav}$ | An integral term added to the cost function to counter errors in gravity compensation |
| $\boldsymbol{e}_{int}$ | Approximate integral of end effector position error |
| $\boldsymbol{e}_o$ | Current error in end effector position |
| $\boldsymbol{f}_i^{ext}$ | External contact force vector on robot links |
| $\boldsymbol{f}_i^{measured}$ | Measured normal force for contact $i$ |
| $f_{threshold}$ | User-defined allowable contact force threshold |
| $\boldsymbol{F}(\dot{\boldsymbol{q}})$ | Coulomb and viscous joint friction |
| $\boldsymbol{G}(\boldsymbol{q})$ | Configuration dependent gravity joint torques |
| $\hat{\boldsymbol{G}}(\boldsymbol{q})$ | Estimate of configuration dependent gravity joint torques used for gravity compensation |
| $H_u$ | Number of time steps in the prediction model where there is control authority |
| $H_y$ | Number of time steps in the prediction model with the control input set to zero |
| $\boldsymbol{J}_{ee}$ | Geometric Jacobian at the end effector |
| $\boldsymbol{J}_{c_i}$ | Geometric Jacobian at contact $i$ |
| $k$ | Discrete time index |
| $\boldsymbol{K}_{c_i}$ | Cartesian stiffness matrix for contact $i$ |
| $k_i$ | Gain on the error integrator, $\boldsymbol{e}_{int}$ |
| $\boldsymbol{K}_p$ | Diagonal joint stiffness matrix |
| $\boldsymbol{K}_d$ | Diagonal joint damping matrix |

**Electronic supplementary material** The online version of this article (doi:10.1007/s10514-015-9492-6) contains supplementary material, which is available to authorized users.

✉ Marc D. Killpack
marc_killpack@byu.edu

[1] Georgia Institute of Technology, Atlanta, GA, USA

[2] Present Address: Brigham Young University, Provo, UT, USA

| | |
|---|---|
| $\boldsymbol{M}(\boldsymbol{q})$ | Configuration dependent joint-space inertia matrix |
| $m$ | Number of degrees of freedom of given robot linkage |
| $\hat{\boldsymbol{n}}_{c_i}$ | Unit vector normal to the surface of the robot at the location of contact $i$ |
| $N$ | The number of contacts at any given time instant |
| $\boldsymbol{q}, \dot{\boldsymbol{q}}$ | State variables of joint angle and velocity |
| $\boldsymbol{q}_{eq}$ | Commanded equilibrium joint angles that are sent to the joint impedance controller |
| $\boldsymbol{q}_{min}$ | Minimum allowable joint angle limits |
| $\boldsymbol{q}_{max}$ | Maximum allowable joint angle limits |
| $\boldsymbol{q}_o$ | Initial joint configuration at current time |
| $t_o$ | Current time which is always the starting point for the predictive model |
| $\boldsymbol{x}_{c_i}$ | Cartesian position for contact $i$ |
| $\boldsymbol{x}_{ee}$ | Cartesian position of the end effector |
| $\alpha, \beta, \mu, \zeta$ | Scalar weighting terms for the multi-objective cost function |
| $\Delta f_{rate,i}$ | Maximum desired rate at which the contact force should be allowed to change at contact $i$ |
| $\Delta \boldsymbol{q}_{eq}$ | Change in equilibrium joint angles, this is the output of MPC |
| $\Delta \boldsymbol{q}_{max,eq}$ | Maximum allowable change in commanded joint angles |
| $\Delta t_{impact}$ | Time duration of an unexpected impact |
| $\Delta \boldsymbol{x}_{des}$ | Desired change in position at the end effector |
| $\Delta t_d$ | Size of continuous time step used to generate discrete-time difference equations for prediction and dynamic MPC |
| $\boldsymbol{\tau}_{ext}$ | Joint torques that result from the sum of all external forces due to contact |
| $\boldsymbol{\tau}_{control}$ | Commanded joint torque that results from "simple joint impedance control" and gravity compensation calculations |
| $\boldsymbol{\tau}_{impact}$ | Average torque due to impact forces occurring during an unexpected collision |
| $x_{th}$ | The integral term $\boldsymbol{d}_{grav}$ becomes effective when $\boldsymbol{e}_0$ is below this threshold value |

## 1 Introduction

The current state of capabilities for robot manipulation in domains such as in-home assistance, search and rescue, and natural or military disasters lags behind human capability and speed. One particular capability at which humans (and other animals) excel is using haptic feedback to operate effectively in clutter. Robots with this capability could potentially perform tasks better in constrained and dynamic scenarios such as reaching into containers or cupboards without line-of-sight, performing search and rescue in debris, or working alongside human co-workers.

In this paper, we specifically consider the problem of a robot arm (i.e., a serial manipulator) reaching into clutter in order to move its end effector to a target position. We define success to be when all contact forces that occur are low and the end effector attains a position close to the target. Success does not depend on the orientation of the end effector. In contrast to our previous research, we focus on enabling the robot to reach the target position in a short amount of time. As such, dynamic phenomena, such as inertia and impact forces, play an important role.

Model predictive control (MPC) offers a promising approach to handling the multiple objectives and constraints associated with reaching in clutter. As we have previously discussed in Jain et al. (2013), while reaching in dense clutter, a robot is likely to make contact between its arm and the environment at multiple locations. Moreover, the robot is unlikely to anticipate each contact event before it occurs. In Jain et al. (2013), we presented a single-time-step model predictive controller that used a quasistatic mechanical model of the robot in contact with the environment. This controller, which we refer to as quasistatic MPC, achieved good empirical performance across a variety of circumstances in terms of enabling robots to successfully reach target locations while keeping contact forces low. Quasistatic MPC assumed that the robot had whole-arm tactile sensing and compliant joints. Our results also provided evidence that, when reaching in clutter, whole-arm tactile sensing and joint-torque sensing together enable superior performance compared to joint-torque sensing alone or joint-torque sensing with a force–torque sensor on each link for contact force sensing.

However, the quasistatic model used by quasistatic MPC did not account for dynamic properties, such as link inertia and joint damping. As expected, the robot performed best when moving at low velocities for which the dynamics become negligible. Many tasks would benefit from faster robots, but effectively controlling forces from multiple intermittent and unexpected contacts presents a substantial challenge. Faster end effector velocities tend to result in both higher impact forces and higher forces from persistent contact. Even intentionally slow motion with compliant joints can result in dynamic phenomena, such as when an arm with a preloaded compliant joint slips off of an object, allowing the joint to release its stored energy and thereby accelerate the robot links.

In this paper, we present a controller that enables a robot manipulator to rapidly move its end effector and simultaneously control contact forces in the presence of multiple contacts along the entire robot arm. We present a multi-time-step model predictive controller that models dynamic phenomena relevant to the task of reaching in clutter. This controller, which we refer to as dynamic MPC, models the

dynamics of the arm in contact with the environment in order to predict how contact forces will change and how the robot's end effector will move. Among other factors, the model considers link inertia and damping at the robot's joints. Dynamic MPC also models how joint velocities will influence potential impact forces based on a collision model. This enables the robot to select joint velocities that mitigate potential impact forces. At each time step, dynamic MPC uses linear models that locally approximate the nonlinear dynamics in order to generate a convex optimization problem that it can solve efficiently.

As with our previous work, we assume the following:

- Contact that results in forces below a user defined threshold is acceptable.
- The robot has some form of compliance at its joints (either active or passive).
- Tactile sensors cover the entire surface of the robot arm.

We evaluate our controller empirically. Through tens of thousands of trials in simulation, we show that with dynamic MPC a simulated robot can, on average, reach goals 1.4 to 2 times faster than the required time for our previous controller, while attaining comparable success rates and fewer occurrences of high forces. As expected, dynamic MPC performed better in low-density clutter, where it could use the open space to accelerate. Likewise, it performed better when the controller allowed the robot to apply higher forces (25 N instead of 5 N) to the world, which increased the chance that the robot could slip off of one object and hit another. Interestingly, dynamic MPC also performed better in higher clutter with only low forces allowed, which is a situation that should be well-matched to quasistatic MPC.

We also conducted extensive trials with a real 7 degree-of-freedom (DoF) humanoid robot arm. Throughout our tests, dynamic MPC enabled the robot to rapidly reach locations in dense clutter while keeping contact forces low (see Fig. 1).

We have organized the paper as follows: Sect. 2 discusses related prior research. Section 3 covers the mathematical formulas and assumptions for formulating our controller. Section 4 describes our software, hardware and experimental setup. In Sect. 5, we show evidence for the accuracy of our simplified dynamic model over a short, but useful, time horizon. In Sect. 6 we present and discuss results from comparing quasistatic MPC and dynamic MPC in our software simulation testbed. We present results from a global reaching task with a real robot in Sect. 7. We present results from local tests that emphasize aspects of contact force control on a real robot with ground-truth force–torque data in Sect. 8. We use the term global reaching task to refer to a trial for which the robot's end effector reaching a target location is an important measure of success. We use the term local test to refer to a trial that involves the robot maneuvering for a short period of time over a short distance without consideration of whether or not the robot's end effector attains a target position. We conclude with a discussion of applications and an overview of our results in Sects. 9 and 10.

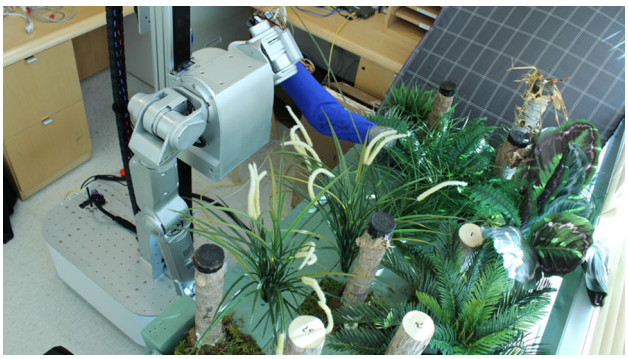

**Fig. 1** Front view of the robot Darci reaching into a cluttered environment

## 2 Related work

We first presented our dynamic controller in a conference paper published at Humanoids 2013 (Killpack and Kemp 2013). The conference paper only conveyed results from two simulated three-link planar robots. In this journal article, we present extensive results from dynamic MPC running on a real 7 DoF humanoid robot arm. In addition, we provide a more thorough formulation of the controller, details about how we altered the controller to run on the real robot, and an empirical evaluation of the fidelity of the controller's approximate linear models.

Many common approaches to robotic manipulation are poorly matched to the challenges of reaching in dense clutter. Methods often rely on collision-free arm motion, line-of-sight sensing of the volume to be traversed, or detailed geometric models prior to reaching (Dogar and Srinivasa 2011; Hornung et al. 2012; Kavraki and laValle 2008; Leeper et al. 2013a, b; Saxena et al. 2008, 2011; Srinivasa et al. 2009; Stilman et al. 2007). Most robotic manipulation research has emphasized avoiding contact except at the end effector (Guo and Hsia 1993; Katz et al. 2013; Latombe 1990; Stilman 2010) or other single point contact locations (De Luca et al. 2006; De Luca and Ferrajoli 2008), even as the robot attempts to operate in unstructured environments. Such restrictions on contact unnecessarily limit a robot's actions when low-force contact is benign and allowable for a given task. Our approach allows multiple contacts along the entire arm surface and uses only haptic feedback to reach through unknown environments.

Work in Erez and Todorov (2012) shows that contact along the arm may be permissible when performing a task, but requires detailed geometric models of the environment, assumes rigid contact, and does not use sensor feedback during the tasks. While results in Mordatch et al. (2012), and

Mordatch et al. (2012) use an optimal control formulation and explicit contact modeling to perform multi-contact tasks, but produce open-loop trajectories and do not use online haptic feedback.

The majority of robotics research on unwanted or unmodeled collisions has focused on reacting to impacts after collision has occurred (De Luca et al. 2006; De Luca and Mattone 2004; Haddadin et al. 2008). This includes work that quantifies forces during impact (Phan et al. 2011) using novel sensing technology and work that models the instantaneous stiffness effects during collision (Shin et al. 2011). Extensive research has aimed to quantify the potential for personal injury from robot-human collisions (Haddadin et al. 2011) and some work has been done to limit robot joint velocities accordingly (Haddadin et al. 2012). In our work, we use an impact-momentum model in our cost function to regulate joint velocities to mitigate contact forces from unexpected impacts without limiting all joint velocities uniformly.

In regards to our approach for control, one of the earliest application areas for MPC was chemical process control (Garcia et al. 1989). MPC is also often referred to as receding horizon control and has been used in work on the control of aerial vehicles (Abbeel et al. 2010; Bellingham et al. 2002). MPC has also been used in robot locomotion research (e.g, Erez et al. 2012; Manchester et al. 2011; Wieber 2006). In terms of robot manipulation, MPC has recently been used in applications such as bouncing a ball (Kulchenko and Todorov 2011), generating manipulator trajectories to compensate for inertial forces on a boat (From et al. 2011), controlling a 6 DoF cable-driven parallel manipulator (Duchaine et al. 2007), and reaching in free space (Ivaldi et al. 2010).

## 3 Model predictive control formulation

In this section, we first explain the architecture of our low-level control framework. We then present the mathematical models our controller uses for predicting the motion and contact forces for our robot. For the prediction step of the controller, we use a forward, discrete-time prediction model. We subsequently show the form of our model predictive controller. Details about our previous work with quasistatic MPC, against which we compare performance with dynamic MPC, can be found in Jain et al. (2013).

The list of symbols at the beginning of the paper summarizes the nomenclature we will use. Lower case variables that are bold face are vectors, while upper case variables that are bold face are matrices, and any non-bold face variable is a scalar.

### 3.1 Overall controller structure

In our system, MPC runs on top of simple joint impedance control (see Hogan 1985; Hogan and Buerger 2005). We use simple joint impedance control due to its compliance and stability when in contact. Equation 1 defines the joint torque control vector, $\boldsymbol{\tau}_{control}$, given a joint space stiffness matrix $\boldsymbol{K}_p$, damping matrix $\boldsymbol{K}_d$, equilibrium angles $\boldsymbol{q}_{eq}$, joint angles $\boldsymbol{q}$, and joint velocities $\dot{\boldsymbol{q}}$. $\hat{\boldsymbol{G}}(\boldsymbol{q})$ provides a gravity-compensating torque.

$$\boldsymbol{\tau}_{control}\left(\boldsymbol{K}_p, \boldsymbol{K}_d, \boldsymbol{q}_{eq}\right) = \boldsymbol{K}_p\left(\boldsymbol{q}_{eq} - \boldsymbol{q}\right) - \boldsymbol{K}_d\dot{\boldsymbol{q}} + \hat{\boldsymbol{G}}\left(\boldsymbol{q}\right) \tag{1}$$

For our work, $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are nonsingular diagonal matrices, so the robot can be thought of as moving its arm by changing the equilibrium angles, $\boldsymbol{q}_{eq}$, of viscoelastic torsional springs located at its joints. We use compliant springs that are virtual, but they could potentially be implemented as real physical springs. In the absence of disturbances such as gravity or other externally applied forces, the robot's arm will eventually settle such that its joint angles equal the commanded equilibrium angles, $\boldsymbol{q} = \boldsymbol{q}_{eq}$. Hogan *et al* refer to a set of equilibrium angles over time as a virtual trajectory.

While our implementation is mathematically analogous to proportional derivative control (PD control), the semantics are different. For example, the equilibrium angles, $\boldsymbol{q}_{eq}$, are not desired joint angles, and a difference between the equilibrium angles and the current joint angles, $\boldsymbol{q}$, is not necessarily error. Likewise, $K_p$ and $K_d$ are more aptly thought of as defining a desirable mechanical impedance for the joints (e.g., high compliance for our system) rather than as proportional and derivative gains to reduce errors over time.

MPC runs on top of this simple joint impedance controller commanding the equilibrium angles ($\boldsymbol{q}_{eq}$) based on the measured external forces, joint angles, joint velocities, current end effector position, and the desired end effector position (see Fig. 2). The model predictive controller uses a model that explicitly incorporates a model of the underlying simple joint impedance controller. In our implementation, the higher bandwidth simple joint impedance controller defined in Eq. 1 and shown in Fig. 2 runs at 1 kHz. MPC runs in an outer control loop at approximately 20–100 Hz (depending on the robot platform). Between updates of the equilibrium angles, $\boldsymbol{q}_{eq}$, the simple joint impedance controller holds them constant.

### 3.2 Robot equations of motion

In order to use MPC, we need models to predict the change in the robot's state given control inputs and disturbances. In the following sections, we derive nonlinear contact and robot state models for a serial manipulator in contact with its environment. We then present linear contact and robot state models that approximate these nonlinear models and which can be used for dynamic MPC.

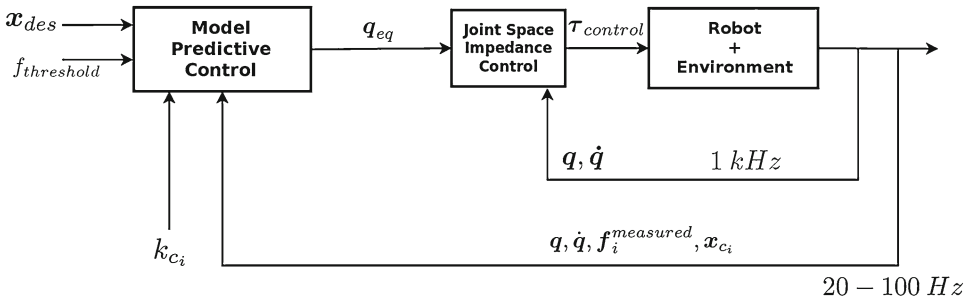


**Fig. 2** This is the block diagram of major components for dynamic MPC

We start with the joint space Lagrangian formulation for a serial torque controlled robot manipulator found in robotics textbooks such as Craig (2005) and Siciliano et al. (2011):

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau}_{ext} + \boldsymbol{\tau}_{control} \tag{2}$$

The variable $\ddot{\boldsymbol{q}}$ represents the joint accelerations and all terms in Eq. 2 are in joint space. On the left hand side of the equation, $\boldsymbol{M}(\boldsymbol{q})$ is the configuration dependent mass matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ represents the Coriolis and centrifugal terms, $\boldsymbol{G}(\boldsymbol{q})$ is the configuration dependent gravity term, and $\boldsymbol{F}(\dot{\boldsymbol{q}})$ is the resultant joint torque vector due to both viscous and Coulomb friction. The terms on the right hand side of Eq. 2 represent the control torques, $\tau_{control}$, and the external torques $\boldsymbol{\tau}_{ext}$ due to external forces applied by the environment. These external torques can be defined as follows if we assume that all contact forces are point contacts:

$$\boldsymbol{\tau}_{ext} = \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T(\boldsymbol{q}) \boldsymbol{f}_i^{ext} \tag{3}$$

In Eq. 3, $N$ is the number of total contacts at the current instant in time. $\boldsymbol{f}_i^{ext}$ is the current contact force at the $i$th contact and $\boldsymbol{J}_{c_i}(\boldsymbol{q})$ is the configuration dependent geometric contact Jacobian at that contact location.

Figure 3, illustrates our dynamic model. The model incorporates link inertia, as well as stiffness and damping at the robot's joints, and contact as bi-linear springs. The springs and dampers at the joints model the underlying joint-space impedance control.

### 3.2.1 Contact model

Our system models each contact as a linear spring that only applies force normal to the surface of the robot's arm. Any forces tangent to the surface of the arm are ignored. At each time step, the system creates a new mechanical model with a linear spring at each location on the arm at which contact has been detected. In our implementation, the robot detects contact when a measured contact force exceeds a threshold.

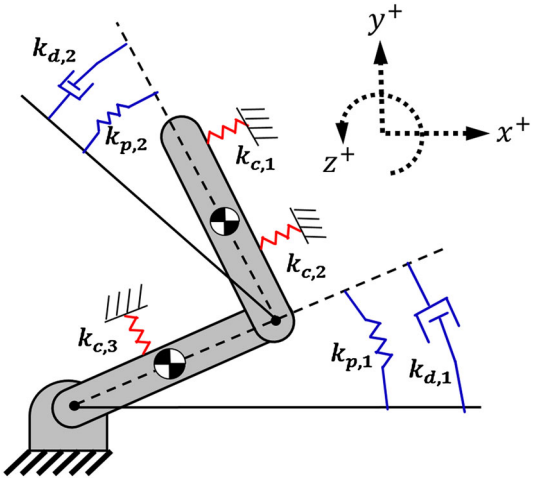


**Fig. 3** Graphical representation of the dynamics that are included in our robot model. The links have mass and rotational inertia. The *blue* elements represent the simple joint impedance control and the *red* springs represent contact with the world (Color figure online)

The model predicts that the force, $\boldsymbol{f}_i^{ext}$, applied to the robot's arm at contact location $i$ will change by $\Delta \boldsymbol{f}_i^{ext}$ when the associated location on the arm moves by $\Delta \boldsymbol{x}_{c_i}$, where

$$\Delta \boldsymbol{f}_i^{ext} = -\boldsymbol{K}_{c_i} \Delta \boldsymbol{x}_{c_i} \tag{4}$$

$\boldsymbol{K}_{c_i}$ is a positive semidefinite Cartesian stiffness matrix, which has the form $\boldsymbol{K}_{c_i} = \hat{\boldsymbol{n}}_{c_i} k_{c_i} \hat{\boldsymbol{n}}_{c_i}^T$. The variable $\hat{\boldsymbol{n}}_{c_i}$ is a unit vector normal to the robot's arm at contact $i$, and $k_{c_i}$ is a positive scalar representing the stiffness associated with motions parallel to this surface normal. Motion in directions orthogonal to the surface normal have zero stiffness and hence result in no predicted change in contact force. Notably, this contact model predicts adhesive forces when the robot moves away from a contact location. As such, unilateral contact models present an interesting direction for future work (Erez et al. 2012; Posa and Tedrake 2013; Stewart and Trinkle 2000).

Our model approximates the translation of contact location $i$ as

$$\Delta \boldsymbol{x}_{c_i} \approx \boldsymbol{J}_{c_i}(\boldsymbol{q_0})(\boldsymbol{q} - \boldsymbol{q_0}) \tag{5}$$

where $\boldsymbol{q_0}$ is the initial joint configuration, $\boldsymbol{q_0} = \boldsymbol{q}(t_0)$, and $\boldsymbol{q}$ is a new joint configuration, $\boldsymbol{q} = \boldsymbol{q}(t)$ with $t > t_0$. Hence,

$$\Delta \boldsymbol{f}_i^{ext} \approx -\boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i}(\boldsymbol{q_0})(\boldsymbol{q} - \boldsymbol{q_0}) \tag{6}$$

and

$$\boldsymbol{f}_i^{ext} \approx \boldsymbol{f}_i^{measured} - \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i}(\boldsymbol{q_0})\,(\boldsymbol{q} - \boldsymbol{q_0}) \tag{7}$$

where $\boldsymbol{f}_i^{measured}$ is the force normal to the robot arm's surface measured at contact $i$ for every time step of the controller.

Combining the predicted contact forces, $\boldsymbol{f}_i^{ext}$, with Eq. 3, gives the following prediction for the external torque, $\boldsymbol{\tau}_{ext}$:

$$\boldsymbol{\tau}_{ext} \approx \Sigma_{i=0}^{N} \boldsymbol{J}_{c_i}^T(\boldsymbol{q})\big(\boldsymbol{f}_i^{measured} - \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i}(\boldsymbol{q_0})\,(\boldsymbol{q} - \boldsymbol{q_0})\big) \tag{8}$$

The controller also regulates the change in the contact forces using the following signed scalar:

$$\Delta f_i^{ext} = \hat{\boldsymbol{n}}_{c_i}^T \Delta \boldsymbol{f}_i^{ext} \approx -\hat{\boldsymbol{n}}_{c_i}^T \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i}(\boldsymbol{q_0})\big(\boldsymbol{q} - \boldsymbol{q_0}\big) \tag{9}$$

### 3.2.2 Linear robot model

For our dynamic robot model, we neglect joint friction ($\boldsymbol{F}(\dot{\boldsymbol{q}}) = 0$). If viscous joint friction were significant, we could incorporate it into our model of joint damping in the simple impedance controller.

We combine Eqs. 1, 2 and 8, and rearrange them to obtain a state-space representation which gives the following:

$$\begin{bmatrix} \ddot{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{q} \end{bmatrix} + \boldsymbol{B} \begin{bmatrix} \boldsymbol{q}_{eq} \\ \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T(\boldsymbol{q}) \boldsymbol{f}_i^{measured} \\ \boldsymbol{q}_0 \end{bmatrix} \tag{10}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A_{11}} & \boldsymbol{A_{12}} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \tag{11}$$

$$\boldsymbol{A_{11}} = -\boldsymbol{M}(\boldsymbol{q})^{-1}(\boldsymbol{K}_d + \boldsymbol{C}(\dot{\boldsymbol{q}}, \boldsymbol{q})) \tag{12}$$

$$\boldsymbol{A_{12}} = -\boldsymbol{M}(\boldsymbol{q})^{-1}\left(\boldsymbol{K}_p + \sum_{i=1}^{N}(\boldsymbol{J}_{c_i}^T(\boldsymbol{q}) \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i}(\boldsymbol{q_0}))\right) \tag{13}$$

$$\boldsymbol{B} = \boldsymbol{M}(\boldsymbol{q})^{-1} \begin{bmatrix} \boldsymbol{K}_p & \boldsymbol{I} & \sum_{i=1}^{N}(\boldsymbol{J}_{c_i}^T(\boldsymbol{q}) \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i}(\boldsymbol{q_0})) \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{14}$$

The matrix $\boldsymbol{I}$ is an $m$ by $m$ identity matrix, where $m$ is the number of joints. $\boldsymbol{M}(\boldsymbol{q})$ is, by definition, a positive definite matrix and therefore invertible. We derived the symbolic form of the mass matrix, Coriolis, and gravity terms using a symbolic Python library (see Sousa 2014). Our linear model approximates $\boldsymbol{M}(\boldsymbol{q}), \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and all contact Jacobians ($\boldsymbol{J}_{c_i}(\boldsymbol{q})$) as constants for the short time horizon of length $H$ over which dynamic MPC makes predictions, such that:

$$\boldsymbol{M}(\boldsymbol{q}(t)) \approx \boldsymbol{M}(\boldsymbol{q}(t_0)) \quad t_0 \le t \le t_0 + H \tag{15}$$

$$\boldsymbol{C}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) \approx \boldsymbol{C}(\boldsymbol{q}(t_0), \dot{\boldsymbol{q}}(t_0)) \quad t_0 \le t \le t_0 + H \tag{16}$$

$$\boldsymbol{J}_{c_i}(\boldsymbol{q}(t)) \approx \boldsymbol{J}_{c_i}(\boldsymbol{q}(t_0)) \quad t_0 \le t \le t_0 + H \tag{17}$$

The variable $H$ is a continuous amount of time and can be defined such that $H = (H_u + H_y)\Delta t_d$. $\Delta t_d$ is the duration of the continuous time step used for the discretization of our prediction model. Using the approximations of Eqs. 15 through 17, we can discretize our system with the matrix exponential (see Brogan 1991). This entails performing the following calculation:

$$\boldsymbol{A}_d = e^{\boldsymbol{A}\Delta t_d} \tag{18}$$

$$\boldsymbol{B}_d = \left(\int_o^{\Delta t_d} e^{\boldsymbol{A}\lambda} d\lambda\right) \boldsymbol{B} \tag{19}$$

To approximate the matrix exponential, we use a Padé approximation (see Moler and Van Loan 2003), which has favorable computational performance and stability with respect to other methods. We then formulate the following discrete-time state-space equations:

$$\begin{bmatrix} \dot{\boldsymbol{q}}[k+1] \\ \boldsymbol{q}[k+1] \end{bmatrix} = \boldsymbol{A}_d \begin{bmatrix} \dot{\boldsymbol{q}}[k] \\ \boldsymbol{q}[k] \end{bmatrix} + \boldsymbol{B}_d \begin{bmatrix} \boldsymbol{q}_{eq}[k] + \Delta \boldsymbol{q}_{eq}[k] \\ \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T \boldsymbol{f}_i^{measured}[0] \\ \boldsymbol{q}[0] \end{bmatrix} \tag{20}$$

In these discrete-time equations, $[k]$ and $[k+1]$ represent the current and next time steps. The equilibrium angles for the next time step are $\boldsymbol{q}_{eq}[k+1]$ with $\boldsymbol{q}_{eq}[k+1] = \boldsymbol{q}_{eq}[k] + \Delta \boldsymbol{q}_{eq}[k]$. Variables with the time index $[0]$ represents the value for that variable at time $t_0$. The state-space matrices $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ are constant over the given prediction horizon $H$. These equations are a linear and discretized approximation of the original nonlinear and continuous dynamics.

Section 6 shows that using a horizon of 5 with dynamic MPC outperformed quasistatic MPC across 4800 simulated trials in terms of speed, success rates and regulation of maximum contact forces. Section 5 shows that our linear model can predict the future state of a robot with low error.

In Sect. 5, we provide empirical support for the fidelity of our linear model, and our evaluations of dynamic MPC demonstrate that this model can be used effectively. The approximations we made result in a computationally efficient formulation that is straightforward to implement. Conventional linearization would require the calculation of numerous derivatives at each time step, which would be complicated by the number of contacts changing over time.

### 3.2.3 Impulse–momentum Impact Model

One important motivation for introducing dynamics was to control the joint velocities explicitly. Although we could limit all the joint velocities to remain below constant values, this is unnecessarily stringent. There are many configurations where some links can move faster than others without incurring high contact forces during unexpected impact. To capture this property, we used a joint-space impulse model as shown in Featherstone and Orin (2008) to model the dynamics of impacts. Impulse–momentum models have been used for robotic control before, most often for work on walking robots (Grizzle et al. 2001). We express our impulse–momentum model as

$$\boldsymbol{M}(\boldsymbol{q})\left(\dot{\boldsymbol{q}}^{+}-\dot{\boldsymbol{q}}^{-}\right) \approx \boldsymbol{\tau}_{impact}\Delta t_{impact} \tag{21}$$

where $\dot{\boldsymbol{q}}^{-}$ represents the joint velocities just before impact, $\dot{\boldsymbol{q}}^{+}$ represents the joint velocities just after impact, and $M(\boldsymbol{q})$ is the joint-space mass matrix. $\boldsymbol{\tau}_{impact}$ is the average torque vector due to the force occurring during the collision and $\Delta t_{impact}$ is the duration of the impact. The left hand side of Eq. 21 represents the change in momentum due to impact and the right hand side represents average torques operating over a short period with no displacement or work occurring.

For an unexpected impact on a single link, we regulate the resulting joint torque $\tau_{j,impact}$ for joint $j$ by expressing it as a product of allowable contact force threshold, $f_{threshold}$, and a moment arm $d_{j,impact}$, giving

$$\boldsymbol{\tau}_{impact}\Delta t_{impact} = \begin{bmatrix} d_{1,impact} \\ d_{2,impact} \\ \vdots \\ d_{m,impact} \end{bmatrix} f_{threshold}\Delta t_{impact} \tag{22}$$

For all of the work in this paper, $d_{j,impact} = d_{impact}$ for all $j$, where $d_{impact}$ is a constant scalar (see Fig. 4).

With regard to the change in momentum, we assume that in the worst case scenario of a perfectly elastic collision

$$\dot{\boldsymbol{q}}^{+} = -\dot{\boldsymbol{q}}^{-} \tag{23}$$

and

$$\dot{\boldsymbol{q}}^{+} - \dot{\boldsymbol{q}}^{-} = 2\dot{\boldsymbol{q}} \tag{24}$$

We assume that a collision could happen at any time given the current joint velocities. In order to limit the contact forces from a collision to be below the force threshold, $f_{threshold}$, dynamic MPC uses the following constraint:

$$|2\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}| \leq \boldsymbol{d}_{impact} f_{threshold}\Delta t_{impact} \tag{25}$$

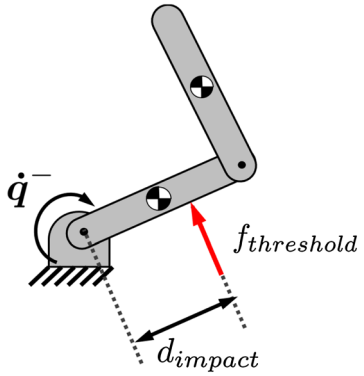


**Fig. 4** This is the visualization of the joint-space impulse–momentum constraint for the first joint of a two link arm

With this model of impulse–momentum between a contact force and joint torque, the worst case scenario for impact location for joint $j$ is when $d_{impact}$ approaches zero. Figure 4 shows a visual representation of how $\boldsymbol{\tau}_{impact}$ is calculated for the first joint. From Fig. 4, it is clear that as $d_{impact}$ goes to zero, $f_{threshold}$ must go to infinity to give the same average torque. However, letting $d_{impact}$ go to zero is unnecessarily conservative for formulating a joint velocity constraint. We wish to avoid the frequent occurrence of high impact forces, but we are not designing the controller for the worst case scenario. To achieve good empirical performance, we tuned $\Delta t_{impact}$ for a robot while holding $d_{impact}$ constant. Since $\Delta t_{impact}$ and $d_{impact}$ are both constant scalars that only appear in this equation, this is equivalent to tuning their product.

Our impulse–momentum constraint described in Eq. 25 does not take into account joint compliance. In addition, contact with some link geometries could cause $d_{impact}$ to be small and result in high forces. However, our evaluations suggest that this collision model is effective in practice. This model also has the advantage of being linear.

## 3.3 Limits and saturation in robot model

A benefit of using MPC is that the robot can plan over a short time horizon, adhering to physical constraints on state and input variables in addition to other user defined constraints (such as force thresholds using models in Sect. 3.2.1). In terms of physically meaningful constraints, we include limits on physical joint angles and high joint velocities (as presented in Sect. 3.2.3). We also include a model of actuator saturation. This takes one of two forms, either (1) limiting the amount that the desired joint angle can be changed at each time step (see Eq. 34), or (2) limiting the total torque each joint can apply (see Sect. 7.2).

### 3.4 Dynamic model predictive controller

Using the equations of motion from Eq. 20, along with the contact models and other constraints, we created a model predictive controller that accounts for dynamics (dynamic MPC). A major difference from quasistatic MPC (see Jain et al. 2013), which used a single time step time horizon, is that dynamic MPC has a horizon that consists of multiple time steps. Dynamic MPC uses a horizon of length $H_u$ during which the controller can apply control effort and a subsequent horizon of $H_y$ time steps during which the controller applies no control effort. This is a common approach to improve stability and robustness and can be found in Rossiter (2003). The values we used for $H_u$ and $H_y$ are presented in relation to specific tests in subsequent sections. The actual physical time that the controller predicts into the future depends on the rate ($\frac{1}{\Delta t_d}$) at which the controller runs. For example, using a total time horizon of ten discrete steps, a time step $\Delta t_d = 0.01$ s would result in the controller predicting up to 0.1 s into the future, and would imply that the controller expects to run at 100 Hz.

The procedure for MPC is to define an optimization problem with a cost function and constraints that are functions of the state variables and that include a model of the system dynamics as an equality constraint. After solving the current convex optimization problem to find a sequence of commanded changes to the equilibrium angles, $\Delta \boldsymbol{q}_{eq}[k]$ for $k = 0, 1, \ldots, H_u + Hy$, the controller only uses $\Delta \boldsymbol{q}_{eq}[0]$ before solving a new convex optimization problem based on updated information. Equations 26 through 35 show the complete optimization problem that dynamic MPC creates and solves at each time step.

$$\underset{\Delta \boldsymbol{q}_{eq}}{\textbf{minimize}}$$

$$\alpha \left\| \Delta \boldsymbol{x}_{des} - \boldsymbol{J}_{ee} \left( \boldsymbol{q}[H_u + H_y + 1] - \boldsymbol{q}[0] \right) \right\|^2 \tag{26}$$

$$+\beta \sum_{k=0}^{H_u} \sum_{i=1}^{N} \max \left( \hat{\boldsymbol{n}}_{c_i}^T \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i} (\boldsymbol{q}[k+1] - \boldsymbol{q}[0]) \right.$$

$$\left. - \left( f_{threshold} - \left\| \boldsymbol{f}_i^{measured}[0] \right\| \right), \boldsymbol{0} \right) \tag{27}$$

$$+\zeta \sum_{k=0}^{H_u} \sum_{i=1}^{N} \max$$

$$\times \left( |\hat{\boldsymbol{n}}_{c_i}^T \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i} (\boldsymbol{q}[k+1] - \boldsymbol{q}[k])| - \Delta f_{rate,i}, \boldsymbol{0} \right) \tag{28}$$

$$+\mu \sum_{k=0}^{H_u} \left\| \Delta \boldsymbol{q}_{eq}[k] \right\|^2 \tag{29}$$

$$\textbf{subject to}: \quad (\forall k = 0, 1, \ldots, H_u + H_y)$$

$$\begin{bmatrix} \dot{\boldsymbol{q}}[k+1] \\ \boldsymbol{q}[k+1] \end{bmatrix} = \boldsymbol{A}_d[k] \begin{bmatrix} \dot{\boldsymbol{q}}[k] \\ \boldsymbol{q}[k] \end{bmatrix} + \boldsymbol{B}_d[k] \begin{bmatrix} \boldsymbol{q}_{eq}[k] + \Delta \boldsymbol{q}_{eq}[k] \\ \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T \boldsymbol{f}_i^{measured}[0] \\ \boldsymbol{q}[0] \end{bmatrix} \tag{30}$$

$$\boldsymbol{q}_{eq}[k+1] = \boldsymbol{q}_{eq}[k] + \Delta \boldsymbol{q}_{eq}[k] \tag{31}$$

$$\boldsymbol{q}[k+1] \leqq \boldsymbol{q}_{max} \tag{32}$$

$$\boldsymbol{q}[k+1] \geqq \boldsymbol{q}_{min} \tag{33}$$

$$|\Delta \boldsymbol{q}_{eq}[k]| \leqq \Delta \boldsymbol{q}_{max,eq} \tag{34}$$

$$|2\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}[k+1]| \leqq d_{impact} f_{threshold} \Delta t_{impact} \tag{35}$$

Our cost function for dynamic MPC consists of a terminal cost (Eq. 26) which attempts to move the end effector towards a desired goal position, a cost on non-adhesive forces above a desired force threshold (Eq. 27), a cost on changing the contact force faster than a specified rate (Eq. 28), and a cost on control effort (Eq. 29). Notably, Eq. 27 does not penalize predicted adhesive forces nor predicted contact forces with magnitudes below $f_{threshold}$. Non-adhesive forces with magnitudes above $f_{threshold}$ incur a linear penalty.

In our original formulation, we defined constraints on the allowable contact force. However, due to feasibility issues with convergence of the optimization, we removed these constraints and added terms related to contact forces to the cost function. The constraints in our current formulation consist of the discrete dynamic equations (Eqs. 30 and 31), limits on joint actuation and angles (see Eqs. 32, 33, 34) and the joint velocity constraint described in Sect. 3.2.3, (see Eq. 35). The actuation limits described in Eq. 34 are for the simulation testbed. For the full-sized real robot DARCI, this actuator model takes a different form to limit the torques applied at the joints. This form is described in Sect. 7. The variable $\Delta x_{des}$ is a waypoint that is a fixed step size in a straight line towards the target position unless the distance to the target position is smaller than the nominal step size. Equation 35 describes the impulse–momentum constraint from Sect. 3.2.3. The result of this optimization at each time step is a series of $\Delta \boldsymbol{q}_{eq}$ values from which the controller executes only the first.

We used an optimization tool named CVXGEN to generate efficient C code that solves this optimization problem (see Mattingley and Boyd 2009, 2010, 2012 for details about this tool for web-based convex optimization code generation). Our system uses Python code to generate the data required by this C code. For the average number of contacts experienced during our trials, this Python code executed in around 1–4 ms. The C code solver requires arguments with fixed dimensions. It solved this optimization problem with multiple contacts and a multi-step horizon in around 4–10 ms, depending on the number of degrees of freedom of the robot being controlled.

### 3.5 Evaluation of controller

Our strategy in evaluating our controllers is based on empirical measures of success at reaching a specified goal in clutter

while keeping the contact forces low. Due to the nature of clutter, robots will not always succeed at reaching a target and the contact that occurs can vary widely. To represent this variability, we randomly generated simulated environments and representative real physical environments to represent the complexity of real clutter. We then used success rates, contact forces, and time-to-complete to evaluate dynamic MPC and compare it with our prior work. Details of quasistatic MPC, which we use as a base-line comparison in simulation can be found in Jain et al. (2013) and Killpack (2013).

Infrastructure and testbeds that we used to test our controller include two different software simulation testbeds, and a human-scale robot with torso, mobile base, and two 7 degree-of-freedom arms. Section 4 describes these testbeds in detail.

## 4 Testbeds and infrastructure

### 4.1 Software simulation testbeds

We developed two simulation testbeds for controller development and testing. We first performed tests using the MATLAB Robotics Toolbox (Corke 1996) which gives more control over the numerical integration methods and the contact models that are used (for more details see Killpack 2013; Killpack and Kemp 2013). We also used the Open Dynamics Engine (ODE Smith et al. 2011), which is an open source physics SDK. Both simulations included a simple joint impedance controller running at 1 kHz. They also both simulated the same three link arm described in Jain et al. (2013), Killpack (2013), and Killpack and Kemp (2013). This arm has kinematics, mass and joint limits similar to a human holding a hand outstretched and manipulating in a plane parallel to the ground at shoulder height (see Fig. 5). The three joints had stiffnesses of 30, 20, and 15 N-m/rad, and damping values of 15, 10, 8 N-m-s/rad, which correspond to the torso, shoulder, and elbow joints.

#### *4.1.1 MATLAB simulation using robotics toolbox*

We used our MATLAB simulation testbed to prototype model predictive controllers with the dynamic models presented in Sect. 3.4. This higher fidelity simulation was valuable for testing the fidelity of our dynamic model as we found that although ODE was better at simulating large numbers of contact, it had lower numerical stability and integration accuracy (see Killpack 2013).

Our MATLAB implementation used an explicit spring-damper model for simulating contact. Since there is no native geometry collision library with MATLAB, we also simulated discrete tactile sensing elements (taxels), that were spaced one centimeter apart along the center of each link to which our implementation assigned any simulated contact force. All objects were assumed to have a 1.5 cm radius and a spring-damper contact was simulated when the extent of the arm (which was assumed to have a 1.5 cm radius) was within the extent of the object. Objects were simulated as having a stiffness of 5000 N/m and damping value of 10 N-s/m for the tests in Sect. 5.

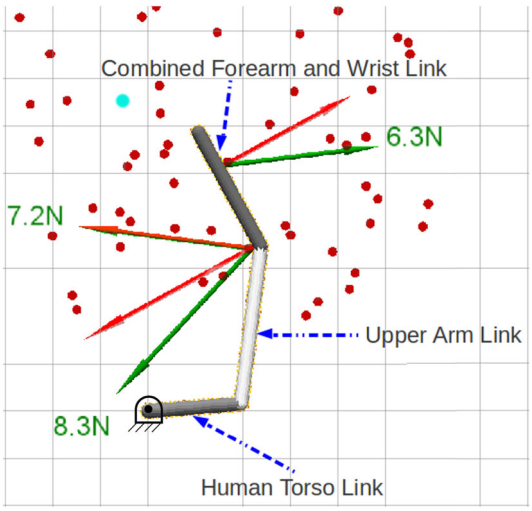


**Fig. 5** Visualization of our simulated robot arm showing the similarity in terms of kinematics to a human reaching in clutter. For the ODE software testbed, this shows the three link planar arm reaching to a goal location (*cyan*) in a volume consisting of rigid cylinders that are fixed (*red*). The base joint of the arm is rigidly fixed to the world. The *orange points* on the arm are 1 cm apart and represent the centers of each tactile sensor. The *green arrows* are the contact force vectors and each *red arrow* is the component of the contact force normal to the surface of the arm which our sensor can measure (Color figure online)

#### *4.1.2 Open source dynamics engine (ODE)*

The ODE software testbed allowed us to simulate a large number of trials and prototype our controllers in highly cluttered workspaces. This was particularly beneficial as we could generate large data sets to test controller performance.

For this platform we simulated tactile sensors covering the entire surface of the arm with the same default density of 100 taxels per meter as in MATLAB. Other specifics (e.g. taxel assignment for each contact force, link geometry, obstacle properties) about our ODE implementation are included in Killpack and Kemp (2013). Figure 5 shows a visualization of the simulated robot, force sensing taxels as well as an example of our simulated environments composed of fixed, rigid cylindrical obstacles (red circles).

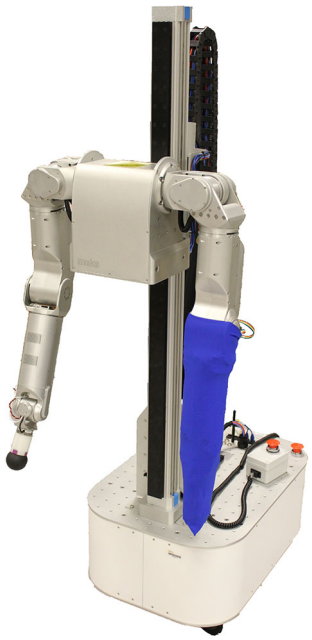

**Fig. 6** Mobile manipulator DARCI produced by Meka Robotics. DARCI has a tactile sensing sleeve which has an external *blue* covering that can be seen on the left arm (Color figure online)

### 4.2 Real robot platform: DARCI

We used a humanoid mobile manipulator called DARCI, a Meka M1 Mobile Manipulator (see Fig. 6). DARCI has two 7 degree-of-freedom arms that have series-elastic actuators (SEAs), are torque controlled and include gravity compensation. Joint stiffness and damping gains for the simple joint impedance controller are 43, 43, 43, 43, 2.6, 3.4, 3.4 N-m/rad and 2.6, 4.3, 0.64, 0.64, 0.064, 0.090, 0.090 N-m-s/rad where they are listed from the most proximal to the most distal joint. The first three values in each list correspond to the shoulder, the fourth value to the elbow, and the last three values to the wrist. We used ROS to send commanded joint angles ($\boldsymbol{q}_{eq}$) to the low-level impedance controller. The computer we used to run our MPC solver in real time had a 32-bit Ubuntu operating system with 16 GB of RAM and a 3.40 GHz Intel Core i7-3770 CPU. The solver only used a single core, running as a single process.

### 4.3 Tactile sensing hardware

We describe specifics of our tactile sensor implementation in Bhattacharjee et al. (2013). The tactile sensor we used on DARCI is a sleeve pulled over the end effector, wrist, and forearm, that consists of five layers of fabric. An inner and outer layer that protect and insulate the sensor's interior, and two layers of conductive fabric that go on either side of a layer of electrically resistive fabric that decreases resistance as the pressure on the cloth increases. One of the conductive layers is split into separate rectangles; where each rectangle is a taxel. The sleeve has 25 taxels total consisting of a single taxel at the tip of the end effector and an array of 24 taxels with 4 taxels around the arm's circumference and 6 along the arm's length. When a taxel detects contact, the system uses the geometric center of the taxel as the location of contact. Figure 6 shows the tactile sensor on the left arm of DARCI.

We calibrated the tactile sensor in order to convert the raw sensor measurements to forces in Newtons. We fit an exponential curve to a plot of tactile sensor readings versus ground truth from a force–torque sensor when pushing on a taxel with various forces. We used the calibration curve from one taxel on all taxels. Although the values reported by these sensors vary in complex ways based on other parameters such as contact area, pressure, and hysteresis, when quasistatic MPC (see Bhattacharjee et al. 2013) and dynamic MPC (see Sect. 8) have used these values they have also performed well with respect to ground-truth forces.

In this paper, our systems detected contact when a tactile sensor's measurements exceeded a threshold. This threshold was 0.5 N for the simulated robot and 0.2 N for the real robot.

## 5 Dynamic model prediction accuracy

The performance of MPC is limited by the quality of the model used to make predictions. For dynamic MPC, it uses a linear approximation of the nonlinear robot dynamics. This approximation assumes that over the prediction horizon changes to the mass and Coriolis matrices are negligible and can thus be treated as constants (see Sect. 3.2.2). In this section we examine the accuracy of our simplified dynamic model for the simulated three-link planar arm in MATLAB as described in Sect. 4.1.1.

### 5.1 Testing accuracy of open loop dynamic model prediction

We first randomly generated 10 different initial joint configurations that were within the physical joint limits of the robot arm. We then randomly generated 5 different goals that were within the same workspace dimensions as our ODE simulation tests in Sect. 6 and in Jain et al. (2013). Finally we also randomly generated 20 fixed objects within that same workspace. The initial joint configurations (differently colored linkages), the goal positions (green stars) and fixed objects (red circles) are shown in Fig. 7. For each initial configuration we ran two sets of tests. For the first set we removed all the objects so the arm reached in free space, while for the second set we used the objects and the arm made contact.

For both sets of trials, at every time step, we solved the model predictive control optimization problem. This solution results in a control sequence over the control horizon ($H_u$) as well as a prediction of how that control sequence will change

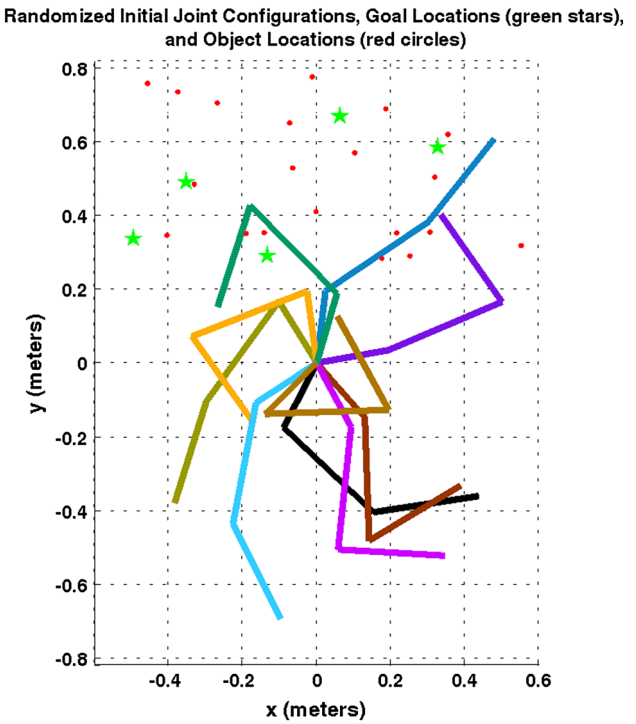


**Fig. 7** This graph shows the initial joint configurations for 10 different starting positions for a three link planar arm. Additionally, the *green stars* are the goals that the arm reached to for each starting configuration and *red circles* are fixed objects (Color figure online)

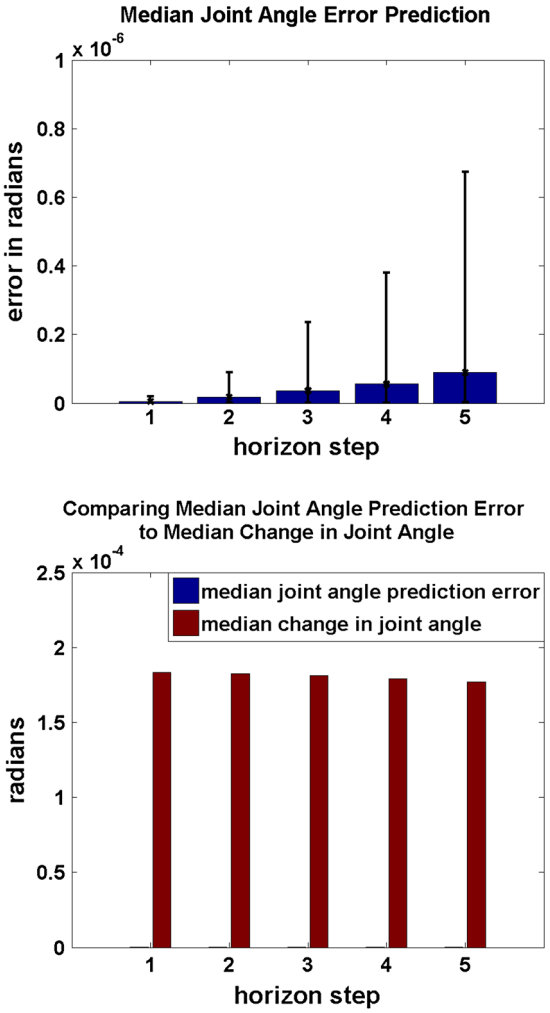


**Fig. 8** *Top*: This plot shows how the median error of the joint angle prediction changed with each step in the horizon. The *error bars* are the 25th and 75th percentile error. *Bottom*: This plot shows a comparison between the magnitude of the median joint angle error versus the median change in joint angle at every step in the horizon (Color figure online)

the joint velocities and angles over that same horizon. At each time step we simulated the motion of the arm using our linear model as if we were to apply all of the control steps over the entire control horizon instead of just the first step. Then, using the full nonlinear equations, we also simulated and compared the actual change in joint angles and velocities to the change predicted by our linear dynamic model. After this simulation over just the horizon ($H_u$), we would reset the state of the arm as if we had only taken the first control step and repeat the process for the next time step. This means that as the arm progressed towards the goal using MPC (executing only the first control input), we also simulated with both our linear and nonlinear models five steps into the future to check the error in our linear prediction model.

In order to quantify the error of the prediction model, we group measurements from all three joints and looked at statistics on the error from the actual and predicted joint angles across the horizon of 5 control steps ($H_u = 4$ since the implementation starts with an index of 0).

### *5.1.1 Prediction of arm moving in free space*

We first looked at the median of the absolute value of prediction errors over each of the five time steps in the horizon for the arm moving in free space. We plot this in Fig. 8. The error bars also show the 25th and 75th percentile error values. As expected, the error increases for predictions farther in the future. However, this does not convey the relationship between the magnitude of the prediction error and the magnitude of the actual change in the joint angles. In Fig. 8, we see the median of the actual change in the joint angles (in red) as compared to the median of the error (in blue). We can see that the median error is much smaller than the median change in the joint angles.

Another way to examine the error is to look at plotting the change in the actual joint angle versus the predicted change from one step to the next. If our prediction is perfect, then plotting the actual change versus the predicted change should be a line with a slope of one. In Fig. 9 we took the data from movement in free space for all three joints across all of the predictions for only the first and fifth steps in the horizon and

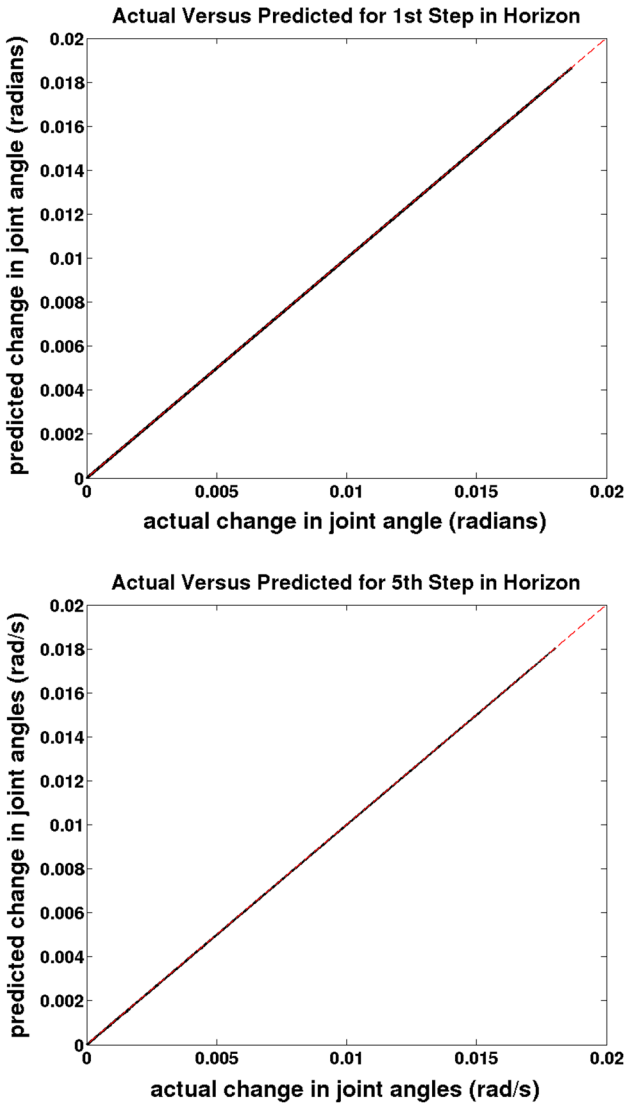


**Fig. 9** This is a scatter plot for free space motion showing the actual change in joint angles versus the predicted change. The *red dotted line* is the ideal if our prediction was perfect. *Top*: The data for the first step in the horizon. *Bottom*: The data for the fifth step in the horizon (Color figure online)

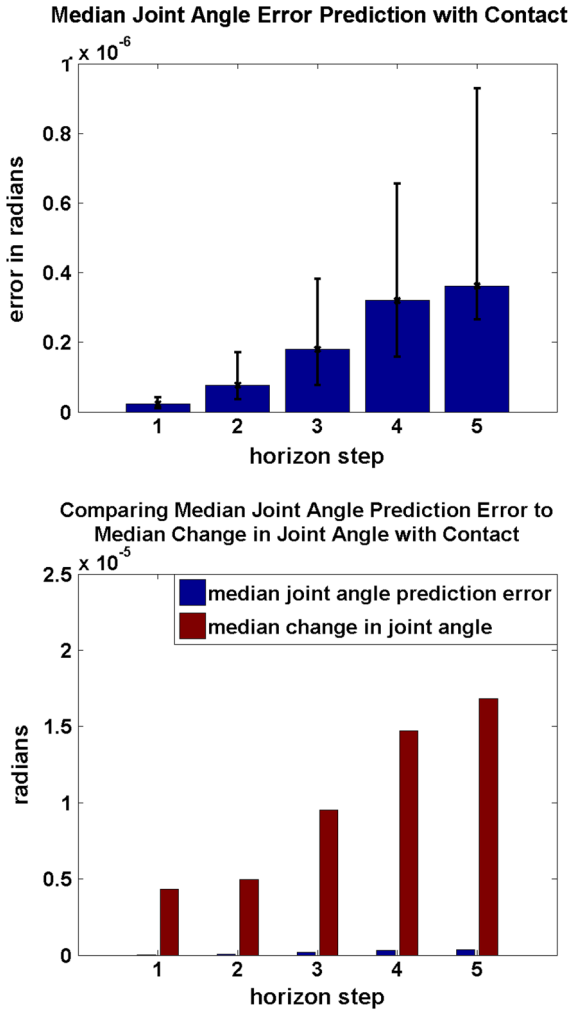


**Fig. 10** *Top*: If we allow the arm to make contact, this plot shows how the median error of the joint angle prediction changed with each step in the horizon. The *error bars* are the 25th and 75th percentile error. *Bottom*: This shows a comparison between the magnitude of the median joint angle error versus the median change in joint angle at every step in the horizon

plotted the actual change in joint angles against the predicted change in joint angles as scatter plots. The red dotted line is the ideal relationship (one to one). The plot in Fig. 9 is for all of the data from the first step in the horizon and the plot in Fig. 9 is for the fifth step in the horizon. We can again see that the prediction model performs well for free space motion.

### *5.1.2 Prediction of arm moving in clutter with contact*

We ran the same tests for the same initial joint configurations but this time included fixed objects so that the arm would make contact while reaching. For this test, in order to evaluate the prediction model, the simulation again executed the entire open loop control sequence over the horizon (5 steps). Note that for our normal implementation of the controller, when contact occurs the contact is included at the next time step when the optimization is reformulated. However, for this test of our linear model, when the robot first makes or breaks contact our model will be inaccurate for however many steps are left in the open loop control horizon the robot is executing. We again show the median of predicted errors and the direct comparison between the median of the prediction error and the median of the change for the joint angles in Fig. 10.

The error in the prediction is higher relative to the actual change in joint position than it was for movement in free space. However, the error is still much smaller than the actual change.

We show the scatter plot of the actual change in the joint angle versus the predicted change for movement with contact in Fig. 11.

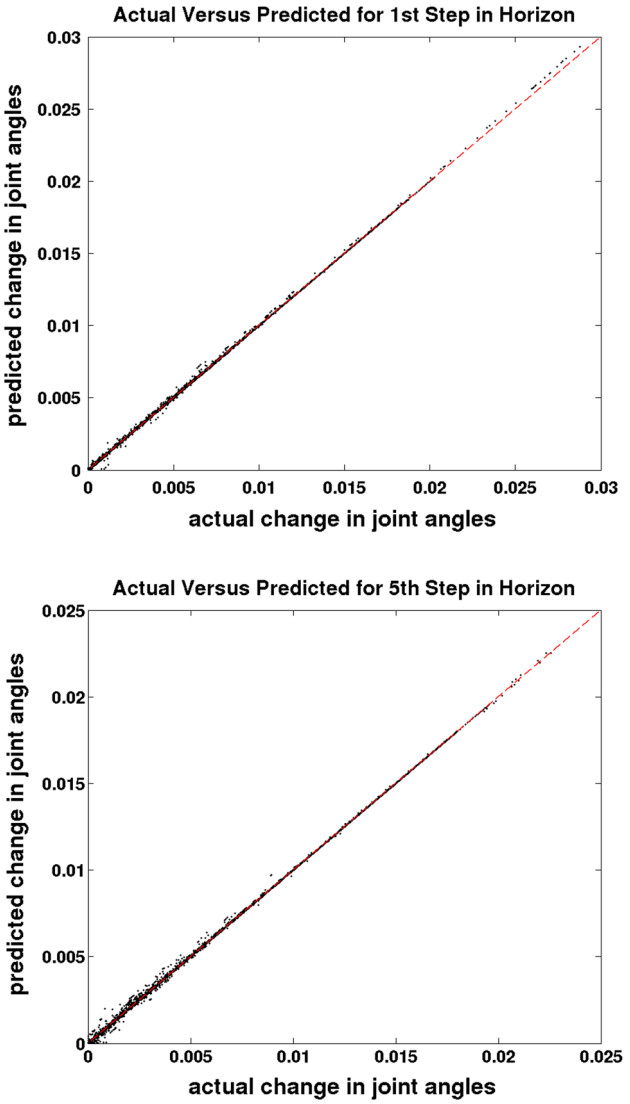


**Fig. 11** This is a scatter plot for when the arm is allowed to make contact, and shows the actual change in joint angles versus the predicted change. The *red dotted line* is the ideal if our prediction was perfect. *Top*: The data for the first step in the horizon. *Bottom*: The data for the fifth step in the horizon (Color figure online)

The error in the prediction is more apparent when moving in contact than when moving in free space. Although the error and the variance are larger at the the end of the horizon (step 5), when dynamic MPC is actually running it updates its contact information at every time step, which would reduce errors due to contact being made or broken.

In this section, we have shown that using our dynamic model, the change in predicted joint angles correlates strongly with the actual change in the joint angles. We also have shown that the error in our predictions due to our linear approximations is small relative to the actual change in the robot state.

## 6 Dynamic versus quasistatic MPC for reaching in simulated clutter

In this section we compare dynamic MPC against quasistatic MPC from Jain et al. (2013) in ODE using the testbed shown in Fig. 5.

In order to compare the performance of our controllers, we ran four sets of 1,200 trials each in ODE for our dynamic controller and for quasistatic MPC. We generated these four sets by varying the density of clutter between 20 fixed objects and 80 fixed objects, and the value of the force threshold between 5 and 25 N. For quasistatic MPC, we used the same data for the 5 N threshold that was reported in Jain et al. (2013). However, for the 25 N threshold, we also generated new results for quasistatic MPC.

Our initial attempts at tuning the parameters of dynamic MPC involved setting the gain on the position cost term first (Eq. 26) and then trying to manually vary the other parameters to achieve desirable performance. However, our multi-objective cost function along with other tunable parameters such as the waypoint magnitude size made tuning the controller difficult and sometimes unintuitive. In order to search the parameter space for our controller, we used simulated annealing on 15 randomly selected environments in ODE (5 with 20 fixed objects and 10 with 80 fixed objects) that were not part of the data set we used for evaluating the controllers. We ran these trials with two different maximum desired force thresholds of 5 and 15 N, and accumulated the cost across all 30 trials. Our cost function for simulated annealing was based on four terms: a cost on the time to complete a trial; a cost on forces over the threshold; a cost on the maximum force in a trial; and a cost on failure to reach the goal. This optimization resulted in a Pareto front that gave us a better intuition for the trade-off between control parameters while at the same time improving performance significantly. We tuned the weightings between the cost terms such that the trade off (especially between costs on force and time to complete) caused the optimization to more equally explore the Pareto front while searching mostly in the space where success rates remained on the same order of magnitude. More details on this tuning with a nine dimensional parameter space are discussed in Killpack (2013).

For quasistatic MPC, we used the nominal tuning from Jain et al. (2013) which is one limitation of our comparison since further tuning might improve performance.

For all of the simulation trials we set the control horizon $H_u = 4$ and prediction horizon $H_y = 1$ according to their description in Sect. 3.4. The parameter $d_{impact}$ as described in Sect. 3.2.3 was set to 0.02 m for all joints which is the diameter of a single cylinder from our simulation testbed. The task for all simulations was specified as reaching to a goal location and we used the

**Table 1** Summary statistics for comparing success rate, contact force control and time to reach the goal between dynamic and quasistatic MPC at different densities of clutter and for different force thresholds

| | Low clutter (20 objects) | | High clutter (80 objects) | |
|---|---|---|---|---|
| | Dynamic model | Quasistatic model | Dynamic model | Quasistatic model |
| *High force threshold (25 N)* | | | | |
| Success rate | **74.6 %** | 72.3 % | **24.6 %** | 23.3 % |
| 99 percentile contact force value (N) (1/100 chance of this force) | 28.3 | 27.6 | 28.5 | 27.8 |
| 99.9 percentile contact force value (N) (1/1000 chance of this force) | **32.3** | 36.1 | **34.1** | 39.8 |
| Maximum force in alltrials | **38.2** | 74.6 | **39.5** | 196.4 |
| Avg. time to complete (s) | **10.9** | 22.3 | **14.2** | 21.1 |
| *Low force threshold (5 N)* | | | | |
| Success rate | **80.9 %** | 77.3 % | **30.1 %** | 28.3 % |
| 99 percentile contact force value (N) (1/100 chance of this force) | **6.5** | 7.9 | **7.1** | 8.6 |
| 99.9 percentile contact force value (N) (1/1000 chance of this force) | **9.9** | 21.2 | **10.6** | 22.0 |
| Maximum force in all trials | **21.0** | 68.2 | **22.6** | 122.1 |
| Avg. time to complete (s) | **12.8** | 22.3 | **14.6** | 21.1 |

The larger rows correspond to a change in force threshold while the larger columns are a change in density of clutter. Boldfaced values show where dynamic MPC performed better

same stopping criterion for success or failure as used in Jain et al. (2013), and Killpack and Kemp (2013). The trial was deemed a success if the arm reached the goal within 2 cm. It was deemed a failure if the max forces were greater than 100 N or the time took longer than 100 s.

We used three main criteria for comparing dynamic MPC, which uses multiple time steps and a dynamic model, to the single-step quasistatic MPC. The first was overall success rate of reaching to a goal position through the simulated clutter (see Sect. 6.1.1). The second criterion was comparing the ability of the two controllers to keep their contact forces below or near the specified desired force threshold (see Sect. 6.1.2). The third criterion was the average time to complete the task for the intersection of successful trials for the two controllers (see Sect. 6.1.3). Table 1 summarizes the results of these comparisons.

## 6.1 Comparison results from ODE simulation

### *6.1.1 Success rates*

Dynamic MPC achieved equal or higher success rates than the quasistatic MPC for all force threshold and clutter settings. In addition, we used a standard significance test for comparing two proportions (McClave et al. 2008) and showed that for two of the four test settings, dynamic MPC's higher success rate was statistically significant with a p-value less than 0.05.

### *6.1.2 Contact force regulation*

For the 99th percentile contact force value (meaning that 1/100 forces measured above the noise threshold was this high or higher), the results were comparable for the two controllers. Figure 12 shows the calculated percentage of contact forces in the y-axis that were above the force value shown on the x-axis for both clutter and force threshold settings. Both controllers are able to successfully affect the measured contact forces according to the force threshold specification. Another way to compare the controllers is to look at events that occur comparatively infrequently, but can have catastrophic effects depending on the specific application. One example of this is extremely high forces. Table 1 shows the value for forces that occur at least once in 1000 force measurements (which at 100 Hz sampling rate is occurring every 10 s that the arm is in contact) as well as the maximum overall measured force. For these two forms of evaluation, dynamic MPC does better in all four settings (high/low clutter and high/low force threshold).

### *6.1.3 Comparing speed of robot end effector*

Table 1 also shows that in all cases dynamic MPC completes the task faster than quasistatic MPC. The speed increase using dynamic MPC for the task ranges from 1.45 to 2.04 times faster than quasistatic MPC. This means that while dynamic MPC is at times moving up to twice as fast as quasistatic MPC, it is still successfully controlling the force and reaching the goal as seen in the previous sections.

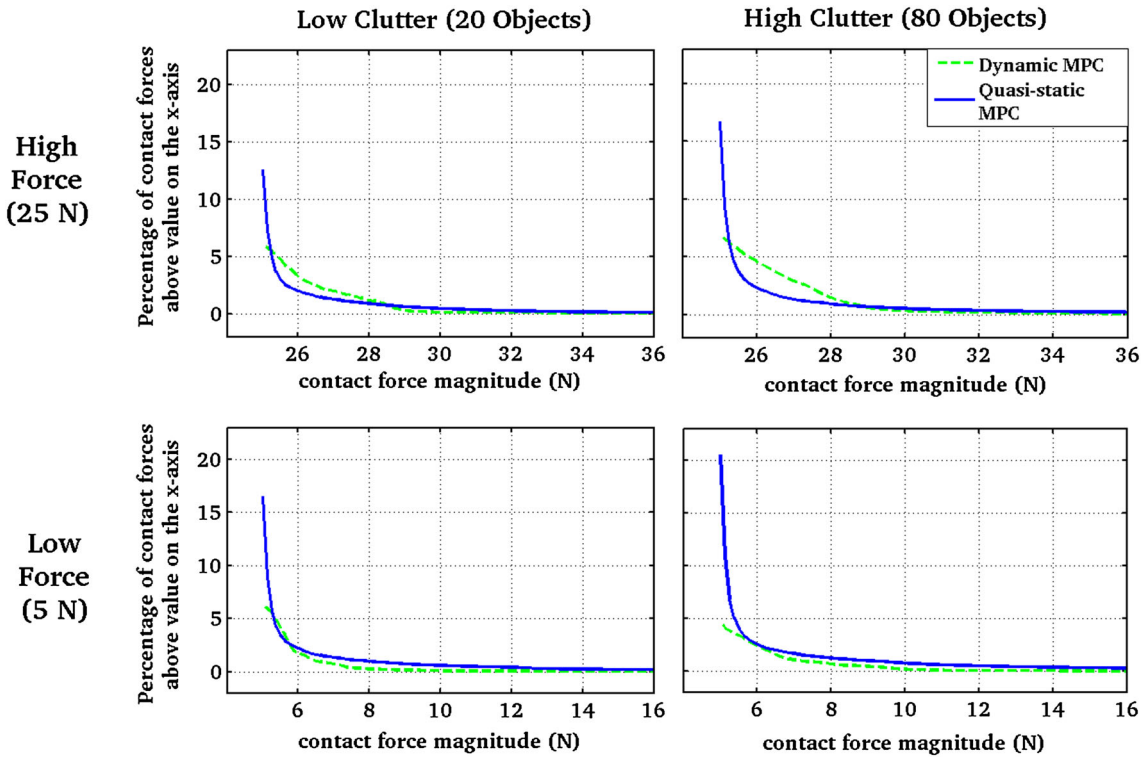


**Fig. 12** These graphs shows how the percent of overall contact forces above the force value in the x-axis decrease as that force value increases. The graphs are organized in *columns* and *rows* such that the *left column* corresponds to low clutter (20 objects), and the *right column* to high clutter (80 objects). The *top row* corresponds to a high allowable contact force threshold (25 N) while the *lower row* corresponds to a low force threshold (5 N)

### 6.2 Limitations of software simulation results

For the simulation results in this section, we have reported results on fixed, stiff objects only. This is in part because fixed, stiff objects initially proved more problematic for controlling the forces using a dynamic model. However, we noticed that for the movable objects that we had used in Jain et al. (2013), dynamic MPC also moved too slowly and conservatively. This is likely due to the contact model where we used a fixed, high stiffness value to represent each contact. Interestingly, in the experiments that we conducted with the real robot, DARCI, the controller performed in foliage environments successfully moving leaves and branches (see Sect. 7). This may indicate that this issue is less important in real world scenarios.

## 7 Results for global task of reaching in clutter using the dynamic model predictive controller with a real robot

In this section we present results using dynamic MPC on the real robot DARCI. We show the ability of dynamic MPC to reach randomly generated goal locations in the clutter. We also show that according to the tactile sensor measurements, the arm is able to control its forces.

### 7.1 Cluttered environment used for testing

The environment that we used for testing reaching in clutter was the same as in Bhattacharjee et al. (2013). This environment consists of plastic leaf-like vegetation and solid wooden trunks as can be seen in Fig. 14. Certain sections of the workspace with leaf-like vegetation can be difficult to push through if the robot respects an allowable contact force threshold. This is due to the density and stiffness of the plants. Figure 14 also shows the coordinate frame used, which is centered between the two robot arms and oriented as shown.

### 7.2 Dynamic MPC adaptation for DARCI

Adapting the controller used in simulation in Sect. 6 to run on DARCI involved a few changes. We used the same controller parameters that we identified for simulation except for $\Delta t_{impulse}$ and the waypoint magnitude size. These we tuned empirically by first setting the waypoint magnitude size without the joint velocity constraint being active. Then, we tuned $\Delta t_{impulse}$ until the impact forces, as measured by our tactile sensing skin, were close to the allowable force threshold.

We also used a different actuation constraint than the one described in Eq. 34. The new constraint describes a limit on

the allowable joint torque as follows:

$$\boldsymbol{\tau}_{min} \leq \boldsymbol{K}_p \left(\boldsymbol{q}_{eq}[k] + \Delta \boldsymbol{q}_{eq}[k] - \boldsymbol{q}[k]\right) - \boldsymbol{K}_d \dot{\boldsymbol{q}}[k] \leq \boldsymbol{\tau}_{max} \tag{36}$$

Because we used software provided with the robot and relied on the high-bandwidth, real time, low-level impedance control to do gravity compensation, gravity was not explicitly modeled in our dynamic model. The provided gravity compensation had significant errors that we compensated for by changing one term in the overall cost function from Eq. 26 to be as follows:

$$\alpha \left\| \Delta \boldsymbol{x}_{des} - \boldsymbol{J}_{ee} \left(\boldsymbol{q}[H_u + H_y + 1] - \boldsymbol{q}[0]\right) - \boldsymbol{d}_{grav} \right\|^2 \tag{37}$$

where $\boldsymbol{d}_{grav} \in \mathbb{R}^3$ acts as an integral term with saturation limits and anti-windup. The method for calculating $\boldsymbol{d}_{grav}$ is described in Algorithm 1 where $\boldsymbol{e}_o[k] \in \mathbb{R}^3$ is the error between the desired and current end effector position at the current time step, and $x_{th}$ is a user-defined scalar distance from the desired goal position. The variable $k_i$ is small and $d_{grav}$ is evaluated once at each time step. Finally the "for" loop shown in Algorithm 1 checks if the sign is different between the sum of the error and the current error for each Cartesian direction. If the sign is different, that term is set to zero to avoid overshoot due to the integral term.

**Algorithm 1** Calculation of Integral Term

**function** CALCINTEGRALTERM
  $\boldsymbol{e}_o[k] = \boldsymbol{x}_{des} - \boldsymbol{x}_{ee}[k]$
  **if** $\|\boldsymbol{e}_o[k]\| < x_{th}$ **then**
    $\boldsymbol{e}_{int}[k] = k_i \boldsymbol{e}_o[k] + \boldsymbol{e}_{int}[k-1]$
  **else**
    $\boldsymbol{e}_{int}[k] = 0$

  **if** $\|\boldsymbol{e}_{int}[k]\| > x_{th}$ **then**
    $\boldsymbol{e}_{int}[k] = \frac{\boldsymbol{e}_{int}[k]}{\|\boldsymbol{e}_{int}[k]\|} x_{th}$

  **for** $i = 1, 2, 3$ **do**
    **if** $\boldsymbol{e}_{int}[k][i] * \boldsymbol{e}_o[k][i] < 0$ **then**
      $\boldsymbol{e}_{int}[k][i] = 0$

  $\boldsymbol{d}_{grav}[k] = \boldsymbol{e}_{int}[k]$

  **return** $\boldsymbol{d}_{grav}[k]$

For this controller, we set the control horizon $H_u = 2$ and prediction horizon $H_y = 3$ which gives three time steps for control (since $H_u$ begins at zero) and then predicts the output for another four steps (since the controller predicts the state up to $H_y + 1$). The convex optimization problem was generated by CVXGEN (Mattingley and Boyd 2012). Although the optimization could solve an MPC problem at a rate between 50 and 100 Hz, we ran the controller at 25 Hz with 40 ms time steps. We did this because jitter in the control loop due to using a non-realtime operating system with Python and ROS required that the controller run at a lower rate to get consistent timing in the control loop. Note that as long as the dynamic model used in MPC does not go unstable during our experiments because of the size of the time step, our discretization incorporates the time step into the model implicitly. For testing on DARCI, we also moved the constraint that describes our impulse–momentum model (see Eq. 35) into the cost function because noise on our joint velocity signal was causing the optimization to be infeasible when any joint was operating near its constrained joint velocity value.

### 7.3 Controller evaluation for reaching in clutter

To generate goal locations for reaching in the simulated foliage with DARCI, we first estimated the workspace of the arm in the foliage. We extended the arm as far as possible forward in the x-y plane (according to Fig. 14) while moving through the workspace from left to right and recording the end effector positions. This traced out an approximate semi-circle that covered most of the foliage. We then randomly sampled from a uniform distribution over x and y using this approximate semi-circle in the x-y plane while also randomly sampling from a uniform distribution in z between 5 and 30 cm above the ground plane of our testbed in the z-direction to determine goal locations. For testing, the robot reached into the clutter and recorded data for 105 reaches. Figure 13 shows DARCI reaching into the foliage. Extension 1 is a video that shows multiple successful and unsuccessful reaches. For each trial, the arm attempted to reach the goal for 20 s before classifying the attempt as success or failure. Success was determined as when the end effector was within 4 cm (or approximately 1.5 inches) from the desired goal.

### 7.4 Results in terms of success rates, contact forces and speed

The success rate for reaching goals was 85.7 % (90 out of 105 trials). Over 95 % of the successful trials where the arm reached the goal within 4 cm, happened in under 7 s which was well before the maximum allowable time of 20 s. Figure 14 shows the distribution of the goals and the end-effector starting location.

The average time to complete for all of the successful trials was 3.0 s (noting that the end effector traveled an average distance of 35 cm across these same trials looking at the total end effector path traveled). The average velocity at the end effector for successful trials was 22.2 cm/s and for all trials was 12.5 cm/s. In these trials we found that the end effector moved on average 7.5 times faster for successful trials than velocities reported for quasistatic MPC on the robot Cody (Jain et al. 2013).

**Fig. 13** Sequence of images from one trial out of 105 trials where DARCI reached into simulated foliage while controlling estimated contact forces using dynamic MPC

The average force as sensed by our tactile sensor for all contact forces above the 0.2 N noise threshold was 1.6 N, while the average force for all forces above the threshold ($f_{thresh} = 5$ N) was 8.1 N. In terms of high forces, the average maximum contact force per trial across all trials was 0.9 N (since many trials had very low maximum forces) and for contact forces above the 5 N threshold the average maximum force was 8.8 N. A histogram for all contact forces is shown in Fig. 15. There is a sharp drop in the number of forces above the specified 5 N threshold in this histogram. However, there are still a number of forces up to the absolute maximum force sensed which was 13.9 N. In general, we would expect that our max forces when moving faster would be slightly higher. However, with the impulse–momentum constraint we have a way of explicitly trading between maximum forces and joint velocities.

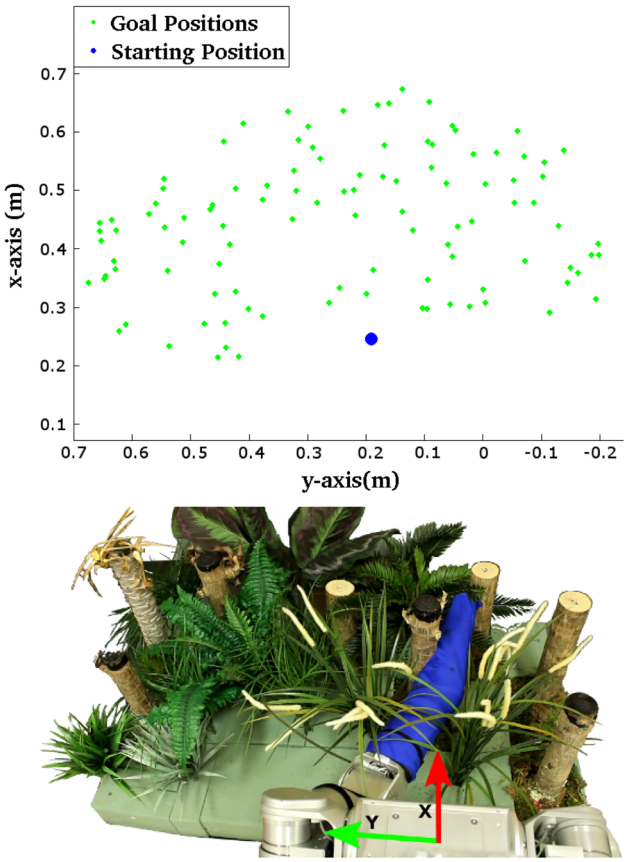


**Fig. 14** *Top:* Distribution of goals (*green*) for the 105 reaching trials as well as the starting end effector position for all trials (*blue*). *Bottom:* Image showing DARCI with its arm partially extended. The farthest goals from the starting position in the figure on the *left* were when the arm was almost fully extended (Color figure online)

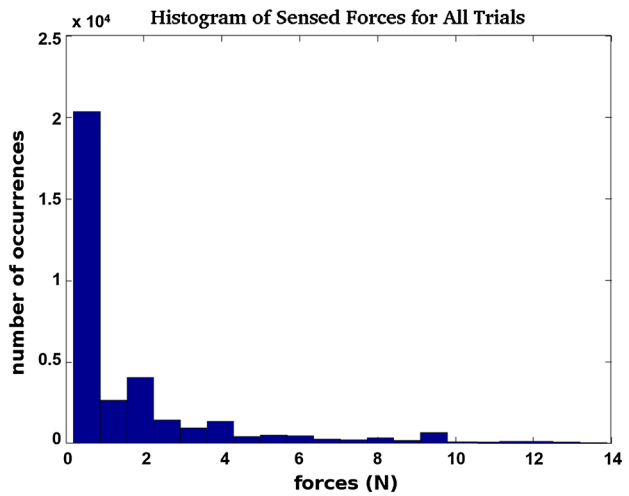


**Fig. 15** Histogram of measured contact forces across all trials using calibrated tactile sensing skin (Color figure online)

Table 2 summarizes the results and contains the values that are relevant to our task of reaching in clutter with dynamic MPC.

**Table 2** Results for dynamic MPC reaching in foliage

| | Dynamic MPC |
|---|---|
| Success rate | 85.7 % (90/105) |
| Exceeded safety threshold (15N) | 0 times |
| Avg. max. of all contact forces | 0.9 N |
| Avg. max. of contact forces over $f_{thresh}$ | 8.8 N |
| Avg. of all contact forces | 1.6 N |
| Avg. of contact forces over $f_{thresh}$ | 8.1 N |
| Avg. time to complete all trials | 5.5 s |
| Avg. time to complete successful trials | 3.0 s |

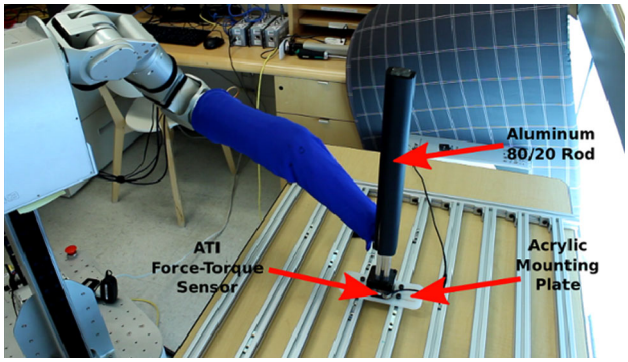


**Fig. 16** Force–torque setup that we used for gathering ground truth contact force data

## 8 Results with local tests for reaching in clutter

Section 7 showed that our system enabled a robot to do fast reaching in clutter while keeping forces low. The trials from previous sections entailed examining a global task of reaching a Cartesian goal location. For this section, we performed tests to analyze the effect of varying the allowable contact force while measuring ground-truth data from force–torque sensors. On the real robot, we performed short trials representative of situations that resulted in high forces or failure in simulation.

Figure 16 shows a typical setup for the robot and the force–torque sensors for these trials. The aluminum rod has a dense foam covering and is attached to the force torque sensor and table using laser cut acrylic. We recorded the data from the force–torque sensors at 100 Hz. For the results we report next, the terms "left" and "right" are in reference to the robot's frame of reference where left is in the positive y direction according to the coordinate frame in Fig. 14.

### 8.1 Evaluating single contact force control with tactile sensor

In these tests we started the arm on either the left or the right side of a single force–torque sensor post (see Fig. 16). For each trial, the robot reached to a pre-defined goal on the opposite side of the post and made initial contact somewhere along the forearm, wrist or end effector. After making initial contact, the arm continued to try reaching the goal using dynamic MPC. The arm was often successful at reaching from right to left, but from left to right would usually get stuck in a local minimum. We refer to a local minimum as a configuration in which our controller would require a different initial condition to be successful or would require a high-level planner to generate Cartesian way points around the obstacle. Figure 17 shows how this trial was executed. Extension 2 is a video of one trial where DARCI reaches from left to right and then right to left.

For these trials we varied the force threshold ($f_{threshold}$) between 5, 10 and 15 N. We ran 10 reaches for each direction (left to right and right to left) resulting in a total of 20 trials at each setting. The purpose of these tests was to verify that varying the force threshold ($f_{threshold}$) had the expected effect on the real robot.

Figure 18 shows the summary statistics for each set of 20 trials as we held $\Delta t_{impulse}$ fixed at 4 s while varying $f_{threshold}$. We calculated the resultant force from the force–torque measurement and used its magnitude as ground truth for a contact force. The statistics that we plotted include the maximum contact force, and the 99th and 50th percentile contact force for each value of $f_{threshold}$. The correlation coefficient between the force threshold value and the 99th percentile forces is 0.99975.

As expected, increasing $f_{threshold}$ for the controller is positively correlated with the maximum and 99th percentile forces that we measured while reaching in contact even if our tactile sensor calibration had some error in terms of ground-truth forces.

### 8.2 Performance of dynamic MPC for multi-contact scenarios

The local tests and results shown in Sect. 8.1 are for contact with a single object. However, multi-contact scenarios often occur in clutter. Phenomena that can occur during multi-contact and cause high forces include wedging and jamming (see Mason 2001). These situations are similar to what happens to our robot arm, for example when simultaneous contact on both sides of the arm cause the arm to quickly decelerate with high contact forces, at which point jamming can occur.

From a wide range of multi-contact scenarios we found in simulation trials from our previous work (see Jain et al. 2013), we developed a set of multi-contact trials. We examined the histograms of contact locations on the three link simulated arm shown in Fig. 5 and identified contact configu-

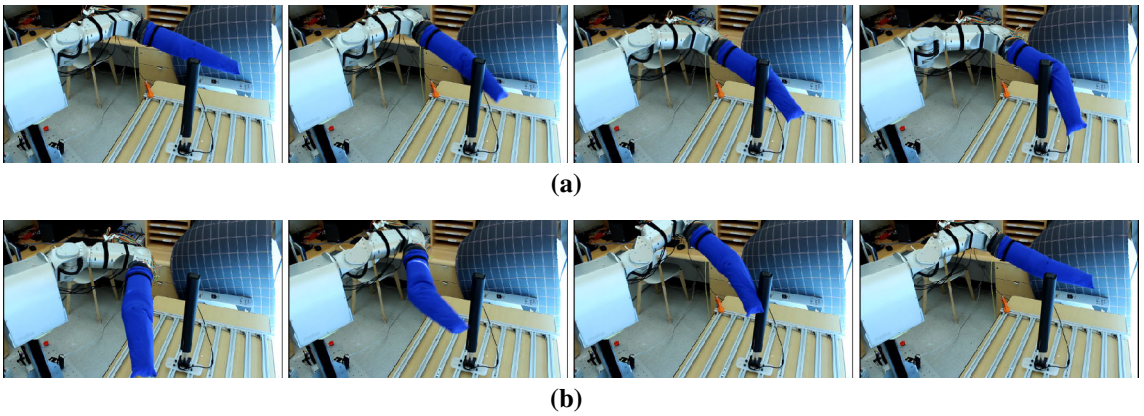


**Fig. 17** Sequence of images shows the arm reaching from *left* to *right* on *top*, and *right* to *left* on *bottom* while making contact with the post. **a** Reaching from left to right. **b** Reaching from right to left

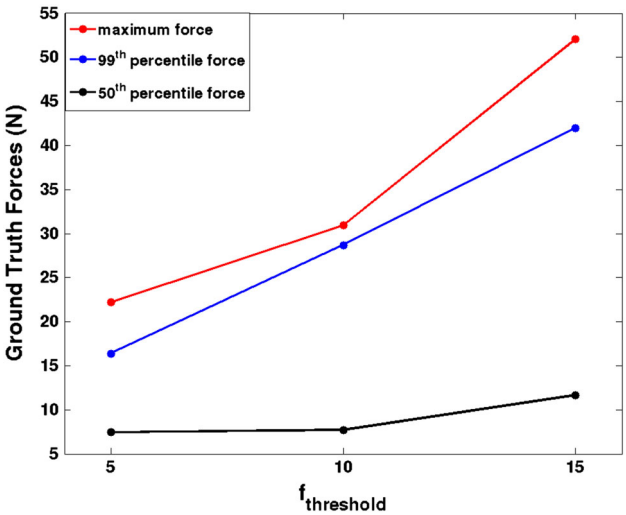


**Fig. 18** Maximum, 99th, and 50th percentile forces as $f_{threshold}$ increased. Each data point represents 20 trials worth of data measuring the ground truth forces with a force–torque sensor at 100 Hz

rations that occurred frequently or resulted in high force. We then manually confirmed that the multi-contact trials were representative of common configurations by randomly sampling and visualizing the actual contact configurations from the simulation trials represented in the contact histograms. These tests are thus subjectively defined, but based on objective data for thousands of tests in simulation.

For each set of trials, we present a figure showing a typical arm trajectory we observed and force data. The resulting force data from the force–torque sensors was the focus of these tests rather than success in reaching the goal.

For all of these multi-contact trials we used $f_{threshold} = 5\text{N}$ and $\Delta t_{impulse} = 4\text{s}$. We ran 20 trials for each multi-contact task. The termination criterion for each trial was successfully reaching the goal or a timeout of 15 s. Note that in these trials it was common for two taxels on our skin sensor to make contact with the force–torque sensor simultaneously (especially at the wrist joint). When this happened, the force–torque sensor value could be up to double the allowed force threshold and our controller would still be successfully controlling skin sensor forces since it was measuring and controlling with respect to two separate contacts.

### *8.2.1 Multi-contact Test 1*

Figure 19 shows the progression of multi-contact test 1 over time. In this test, the robot arm starts in contact the tip of the end effector on the left side and in the middle of the forearm on the right hand side. In the second image in the sequence of Fig. 19, the arm first pushes against both rods as it tries to move to the goal. In subsequent motions the arm uses out of plane motion to move around the post and still get to the

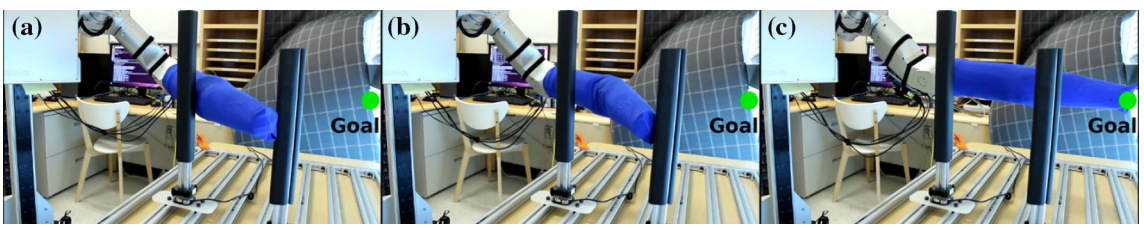


**Fig. 19** Multi-contact Test 1: DARCI reaching to the left while starting in contact at the end effector

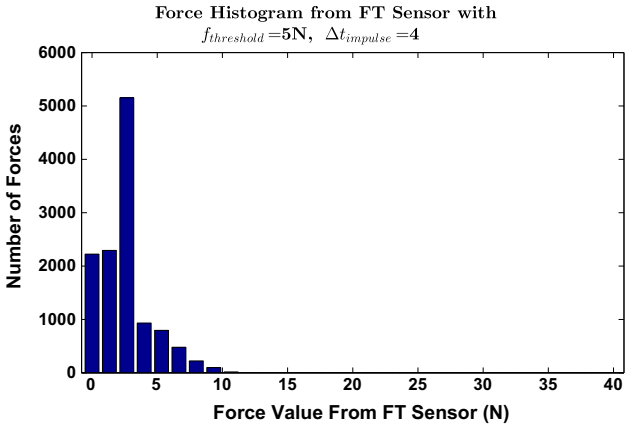


**Fig. 20** Multi-contact Test 1: Force results for reaching to the left

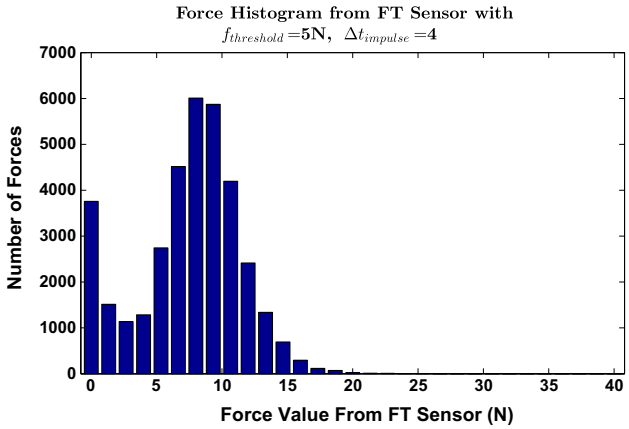


**Fig. 22** Multi-contact Test 2: Force results for reaching to the right while starting in contact at the middle of the forearm

goal. This is one of the trials that we expected to have large forces due to what we saw in the simulation. However, the 3D motion and extra degrees of freedom made it straight forward for our controller to reach the goal and control the forces at the same time. See Extension 3 for video of this trial.

The contact force results for this trial are in Fig. 20. These results have forces that agree with our expectation of force control where almost all of the forces are below 10 N and the majority are below 5 N.

### *8.2.2 Multi-contact Test 2*

Figure 21 shows the progression of multi-contact trial 2. For this test, the arm started in contact with two posts and reached to the right between them. The space between the two posts is much smaller in this trial when compared to the first trial. In this test, the arm could not reach the goal. See Extension 3 for video of this trial.

For the force results of this trial we had higher forces than for the first test. It should be noted that this configuration was more difficult than the first trial as the posts were placed closer together and we expected to have high forces due to jamming in this situation. Most forces were still limited to be below 15 N. This also matches what our test in Sect. 8.1 showed where the 99th percentile contact force for a single contact was about 15 N and corresponded to a 5 N allowable force threshold on the tactile sensor (see Fig. 22).

### *8.2.3 Multi-contact Test 3*

For multi-contact test 3, the arm started in contact near the wrist and was given a goal such that it kept contact with the first force–torque sensor while trying to wedge between two other force–torque sensors to reach the goal shown in Fig. 23. The goal was not reachable due to the small gap between the two distal force–torque sensors. see Extension 3 for video of this trial.

The contact forces for this trial are in Fig. 24. The majority of the forces are again below 15 N with a maximum force around 25 N.

### *8.2.4 Multi-contact results discussion*

Table 3 contains summary statistics of the forces for all of the multi-contact trials. These statistics include the mean of the maximum forces as well as the 99th percentile of all contact forces and the mean of all contact forces for each test.

In prior work (see Bhattacharjee et al. 2013), this type of Meka robot arm applied upwards of 40 N on a force–torque sensor while moving slowly and using the compliance of the simple impedance controller. In our case, it is clear that dynamic MPC is limiting contact forces to be within the same ranges we saw with quasistatic MPC in Bhattacharjee et al. (2013). In many of the trials on DARCI, there was

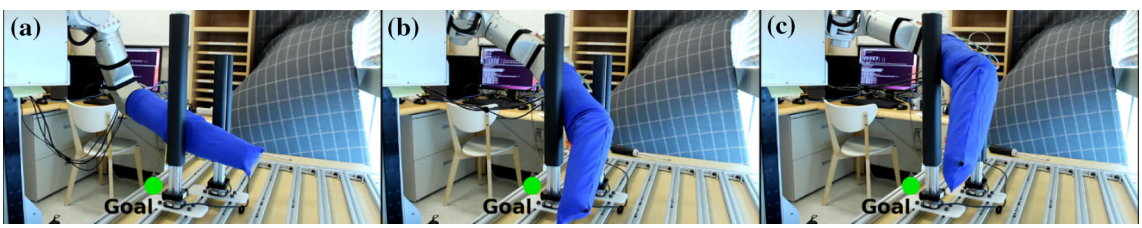


**Fig. 21** Multi-contact Test 2: DARCI in contact with two posts in the middle of its forearm and reaching to the right

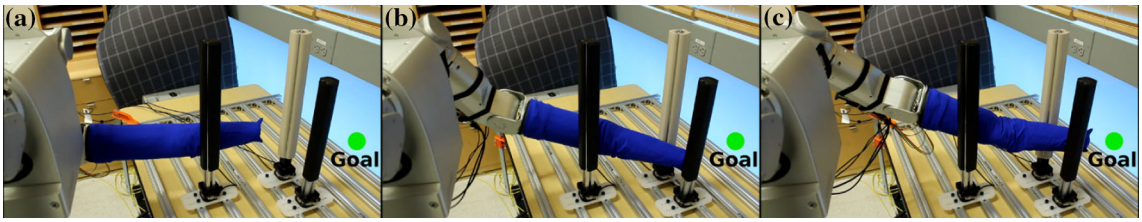


**Fig. 23** Multi-contact Test 3: DARCI reaching to a goal between two posts with the gap between them smaller than the diameter of the arm given the angle from which the arm is constrained to approach

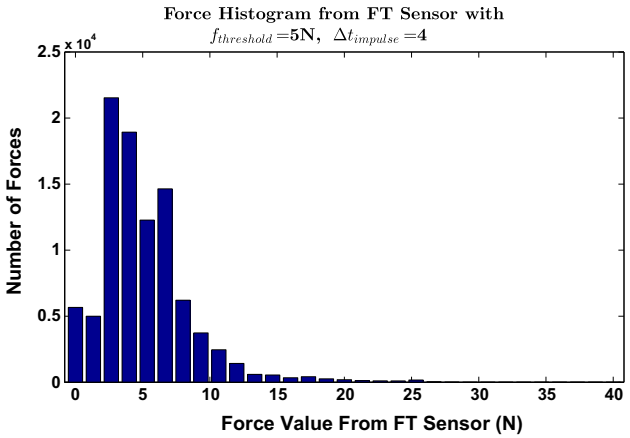


**Fig. 24** Multi-contact Test 3: Force results for reaching straight ahead while wedging between two posts and making contact with a third post

no way to reach the goal, yet dynamic MPC regulated the forces to be generally low despite the fact that as it is noted in Bhattacharjee et al. (2013) "... the relationship between contact forces and taxel output is complex."

### 8.3 Discussion of all local results

The results in this section show that the contact force trends we saw with measurements from only the tactile sensors while reaching in foliage (see Sect. 7) hold even when we use force–torque sensors to obtain ground truth.

## 9 Applications

An open question is the extent to which our controller can be used for other tasks, robots, and environments. We expect that controllers designed for fast reaching with whole-arm contact may eventually serve as general purpose controllers for interaction with people in situations such as in-home assistance or search and rescue.

An anecdotal example of the kind of complex tasks that we envision is shown in the video of Extension 4. Because of the speed of dynamic MPC we can teleoperate DARCI with a Phantom Omni haptic input device to reach into an unmodeled and non-rigid canvas bag with unknown objects inside. The Cartesian position of the haptic input device is measured, scaled and fed to dynamic MPC as a goal position at each time step. The robot is then able to move towards this goal location while locally controlling the contact forces so as to not damage itself or the contents of the bag. The bag is also fixtured at one point to the table with a clamp.

## 10 Conclusion

We have shown that across a large number of simulated trials, using dynamic MPC while incorporating a linear approximation of the nonlinear dynamics of a robot arm in contact gives significant performance improvements in comparison to quasistatic MPC. In particular, we have shown that for two different settings of clutter and two different force thresholds, dynamic MPC had better success rates, faster speeds and comparable force control. We were also able to run dynamic MPC on the real robot DARCI using the manufacturer's nominal dynamic parameters (i.e. no complex system identification was necessary). In addition, we have shown that running this controller in Python at around 25 Hz can result in good performance. We would expect that further engineering efforts would allow one to increase the control rate and further improve our force control. A variant of our dynamic MPC has even recently been used successfully in

**Table 3** These are numerical values that summarize the data presented in the force histograms for the multi-contact test cases in this section

| Test | 99th percentile force (N) | Mean of all forces | Mean of max forces | Max of max forces |
|---|---|---|---|---|
| 1 | 8.6 | 2.5 | 8.1 | 10.0 |
| 2 | 15.9 | 7.5 | 13.8 | 25.3 |
| 3 | 18.3 | 5.1 | 13.3 | 28.1 |

conjunction with haptic mapping, geometric planning, learning and tactile perception (Bhattacharjee et al. 2014).

Our results show the possibility of reaching at faster rates into cluttered environments while controlling velocities, forces, and mitigating the effects of unexpected impact.

**Acknowledgments** We gratefully acknowledge support from DARPA Maximum Mobility and Manipulation (M3) Contract W911NF-11-1-603. Support from NSF Emerging Frontiers in Research and Innovation (EFRI) 1137229 allowed the use of the robot DARCI. This work also benefited from discussions with Wayne J. Book, Magnus B. Egerstedt, Lena Ting, Karen Liu, and Jun Ueda. We also thank Tapomayukh Bhattacharjee, Philip Grice, Advait Jain, Daehyung Park, and Joshua Wade for technical discussions on this work and contributions to the forearm tactile sensor and foliage testbed.



## Appendix: Index to multimedia extensions

| Extension | Type | Description |
|---|---|---|
| 1 | Video | DARCI reaching in simulated foliage as described in Sect. 7.3 |
| 2 | Video | Single contact local tests as described in Sect. 8.1 |
| 3 | Video | Multi-contact local tests as described in Sect. 8.2 |
| 4 | Video | Example of using dynamic MPC for teleoperating DARCI to reach into a non-rigid canvas bag as described in Sect. 10 |

## References

Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, *29*, 1608–1639.

Bellingham, J., Richards, A., & How, J. P. (2002). Receding horizon control of autonomous aerial vehicles. In *American Control Conference, 2002. Proceedings of the 2002* (vol. 5, pp. 3741–3746). IEEE.

Bhattacharjee, T., Grice, P. M., Kapusta, A., Killpack, M. D., Park, D., & Kemp, C. C. (2014). A robotic system for reaching in dense clutter that integrates model predictive control, learning, haptic mapping, and planning. In *Proceedings of the 3rd IEEE/RSJ international conference on intelligent robots and systems (IROS) workshop on robots in clutter: perception and interaction in clutter*.

Bhattacharjee, T., Jain, A., Vaish, S., Killpack, M. D., & Kemp, C. C. (2013). *2013* . In IEEE world haptics conference : Tactile sensing over articulated joints with stretchable sensors.

Bhattacharjee, T., Kapusta, A., Rehg, J., & Kemp, C. C. (2013). Rapid categorization of object properties from incidental contact with a tactile sensing robot arm. In *IEEE international conference on humanoid robotics*, 2013.

Brogan, W. L. (1991). *Modern control theory*. New Jersey: Prentice Hall.

Corke, P. I. (1996). A robotics toolbox for MATLAB. *IEEE Robotics & Automation Magazine*, *3*(1), 24–32.

Craig, J. (2005). *Introduction to robotics: mechanics and control*. New Jersey: Prentice Hall.

De Luca, A., Albu-Schaffer, A., Haddadin, S., & Hirzinger, G. (2006) Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *IEEE/RSJ international conference on intelligent robots and systems*, 2006 (pp. 1623–1630). IEEE.

De Luca, A., & Ferrajoli, L. (2008) Exploiting robot redundancy in collision detection and reaction. In: *IEEE/RSJ international conference on intelligent robots and systems, 2008. IROS 2008* (pp. 3299–3305). IEEE.

De Luca, A., & Mattone, R. (2004). An adapt-and-detect actuator fdi scheme for robot manipulators. In *ICRA*.

Dogar, M., & Srinivasa, S. (2011). A framework for push-grasping in clutter. *Robotics: Science and systems VII*. Cambridge: MIT Press.

Duchaine, V., Bouchard, S., & Gosselin, C. M. (2007). Computationally efficient predictive robot control. *IEEE/ASME Transactions on Mechatronics*, *12*(5), 570–578.

Erez, T., Tassa, Y., & Todorov, E. (2012). Infinite-horizon model predictive control for periodic tasks with contacts. *Robotics: Science and systems VII* (p. 73). Cambridge: MIT Press.

Erez, T., & Todorov, E. (2012). Trajectory optimization for domains with contacts using inverse dynamics. In *IROS*.

Featherstone, R., & Orin, D. E. (2008). Dynamics. *Handbook of robotics (Chap. 2)*. Berlin: Springer.

From, P. J., Gravdahl, J. T., Lillehagen, T., & Abbeel, P. (2011). Motion planning and control of robotic manipulators on seaborne platforms. *Control Engineering Practice*, *19*(8), 809–819.

Garcia, C., Prett, D., & Morari, M. (1989). Model predictive control: Theory and practice–A survey. *Automatica*, *25*(3), 335–348.

Grizzle, J. W., Abba, G., & Plestan, F. (2001). Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *IEEE Transactions on Automatic Control*, *46*(1), 51–64.

Guo, Z., & Hsia, T. (1993). Joint trajectory generation for redundant robots in an environment with obstacles. *Journal of Robotic Systems*, *10*(2), 199–215.

Haddadin, S., Albu-Schaffer, A., De Luca, A., & Hirzinger, G. (2008). Collision detection and reaction: A contribution to safe physical human–robot interaction. In *IEEE/RSJ international conference on intelligent robots and systems, 2008. IROS 2008* (pp. 3356–3363). IEEE.

Haddadin, S., Albu-Schäffer, A., & Hirzinger, G. (2011). Safe physical human–robot interaction: Measurements, analysis and new insights. *Robotics research* (pp. 395–407). Berlin: Springer.

Haddadin, S., Haddadin, S., Khoury, A., Rokahr, T., Parusel, S., Burgkart, R., Bicchi, A., & Albu-Schaffer, A. (2012). A truly safely moving robot has to know what injury it may cause. In *IEEE/RSJ international conference on intelligent robots and systems (IROS), 2012* (pp. 5406–5413). IEEE.

Hogan, N. (1985). Impedance control-an approach to manipulation. I-Theory. II-Implementation. III-Applications. *ASME Transactions Journal of Dynamic Systems and Measurement Control*, *107*, 1–24.

Hogan, N., & Buerger, S. (2005). Impedance and interaction control (Chap. 9). *Robotics and automation handbook*. Boca Raton: CRC-Press.

Hornung, A., Phillips, M., Jones, E. G., Bennewitz, M., Likhachev, M., & Chitta, S. (2012) Navigation in three-dimensional cluttered environments for mobile manipulation. In *IEEE international conference on robotics and automation (ICRA)*, 2012 (pp. 423–429). IEEE.

Ivaldi, S., Fumagalli, M., Nori, F., Baglietto, M., Metta, G., & Sandini, G. (2010). Approximate optimal control for reaching and trajectory planning in a humanoid robot. In *IEEE/RSJ interna-*

*tional Conference on intelligent robots and systems (IROS)*, 2010 (pp. 1290–1296). IEEE.
Jain, A., Killpack, M. D., Edsinger, A., & Kemp, C. (2013). Reaching in clutter with whole-arm tactile sensing. *The International Journal of Robotics Research*, *32*, 458–482.
Katz, D., Venkatraman, A., Kazemi, M., Bagnell, J. A., & Stentz, A. (2013). Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Autonomous Robots*, *37*, 369–382.
Kavraki, L. E., & la Valle, S. M. (2008). Motion planning. In B. Siciliano & O. Khatib (Eds.), *Handbook of robotics*. New York: Springer.
Killpack, M. D. (2013). Model predictive control with haptic feedback for robot manipulation in cluttered scenarios. Dissertation, Institute of Technology, Georgia. http://hdl.handle.net/1853/50207.
Killpack, M. D., & Kemp, C. C. (2013). Fast reaching in clutter while regulating forces using model predictive control. In *IEEE-RAS international conference on humanoid robots (humanoids)*.
Kulchenko, P., & Todorov, E. (2011). First-exit model predictive control of fast discontinuous dynamics: Application to ball bouncing. In *IEEE international conference on robotics and automation (ICRA)*, 2011 (pp. 2144–2151). IEEE.
Latombe, J. C. (1990). *Robot motion planning*. Berlin: Springer.
Leeper, A., Hsiao, K., Ciocarlie, M., Sucan, I., & Salisbury, K. (2013). Arm teleoperation in clutter using virtual constraints from real sensor data. In *RSS workshop on robots in clutter: preparing robots for the real world*.
Leeper, A., Hsiao, K., Ciocarlie, M., Sucan, I., & Salisbury, K. (2013). *Methods for collision-free arm teleoperation in clutter using constraints from 3d sensor data*. Atlanta, GA: In IEEE international conference on humanoid robots.
Manchester, I. R., Mettin, U., Iida, F., & Tedrake, R. (2011). Stable dynamic walking over uneven terrain. *The International Journal of Robotics Research*, *30*, 3.
Mason, M. (2001). *Mechanics of robotic manipulation*. London: MIT Press.
Mattingley, J., & Boyd, S. (2009). *Automatic code generation for real-time convex optimization. Convex optimization in signal processing and communications*. Cambridge: Cambridge University Press.
Mattingley, J., & Boyd, S. (2010). Real-time convex optimization in signal processing. *IEEE Signal Processing Magazine*, *27*(3), 50–61.
Mattingley, J., & Boyd, S. (2012). Cvxgen: a code generator for embedded convex optimization. *Optimization and Engineering*, *13*, 1–27.
McClave, J., Benson, P., & Sincich, T. (2008). *Statistics for business and economics*. London: Pearson Education.
Moler, C., & Van Loan, C. (2003). Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, *45*(1), 3–49.
Mordatch, I., Popovic, Z., & Todorov, E. (2012). Contact-invariant optimization for hand manipulation. In *Eurographics/ACM SIGGRAPH symposium on computer animation* (pp. 137–144). The Eurographics Association.
Mordatch, I., Todorov, E., & Popović, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, *31*(4), 43.
Phan, S., Quek, Z., Shah, P., Shin, D., Ahmed, Z., Khatib, O., & Cutkosky, M. (2011). Capacitive skin sensors for robot impact monitoring. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2992–2997). IEEE.
Posa, M., & Tedrake, R. (2013). Direct trajectory optimization of rigid body dynamical systems through contact. *Algorithmic foundations of robotics X* (pp. 527–542). Berlin: Springer.
Rossiter, J. (2003). *Model-based predictive control: a practical approach*. Boca Raton: CRC.
Saxena, A., Driemeyer, J., & Ng, A. (2008). Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, *27*(2), 157.
Saxena, A., Wong, L., Quigley, M., & Ng, A. Y. (2011). A vision-based system for grasping novel objects in cluttered environments. *Robotics research* (pp. 337–348). Berlin: Springer.
Shin, D., Quek, Z., Phan, S., Cutkosky, M., & Khatib, O. (2011). Instantaneous stiffness effects on impact forces in human-friendly robots. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 2998–3003). IEEE.
Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2011). *Robotics: modelling, planning and control*. Berlin: Springer.
Smith, R., et al. (2011). Open dynamics engine. http://www.ode.org.
Sousa, C. A. D. (2014) SymPyBotics v1.0. doi:10.5281/zenodo.11365. http://zenodo.org/record/11365.
Srinivas, S., Ferguson, D., Helfrich, C., Berenson, D., Collet, A., Diankov, R., et al. (2009). Herb: a home exploring robotic butler. *Autonomous Robots*, *28*, 5–20.
Stewart, D., & Trinkle, J. (2000). An implicit time-stepping scheme for rigid body dynamics with coulomb friction. In *IEEE international conference on robotics and automation, 2000. Proceedings ICRA'00* (vol. 1, pp. 162–169). IEEE.
Stilman, M. (2010). Global manipulation planning in robot joint space with task constraints. *IEEE Transactions on Robotics*, *26*(3), 576–584.
Stilman, M., Schamburek, J., Kuffner, J., & Asfour, T. (2007). Manipulation planning among movable obstacles. In *Proceedings of IEEE international conference on robotics and automation* (pp. 3327–3332). Citeseer.
Wieber, P. B. (2006). Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In *6th IEEE-RAS international conference on humanoid robots, 2006* (pp. 137–142). IEEE.

**Marc D. Killpack** is an Assistant Professor in the Department of Mechanical Engineering at Brigham Young University. He completed his doctorate in Robotics from the Woodruff School of Engineering at Georgia Tech (2013) as well as his MS in Mechanical Engineering (2008). He also completed a Master Pro Degree from ENSAM in Metz, France, and a BS in Mechanical Engineering from BYU. He founded the Robotics and Dynamics Lab at BYU (http://radlab.byu.edu) in 2014 where they focus on research to improve the ability of robots to function in cluttered and dynamic environments. He has recently received the NASA Early Career Faculty Award which is allowing continuing research on control of soft robots for manipulation in cluttered environments.

**Ariel Kapusta** is a Robotics Ph.D. Student at the Georgia Institute of Technology. He earned a B.S. degree in Mechanical Engineering from Rice University in 2009. His research interests include haptic perception, control systems for robotic manipulation, human-robot interaction, novel robotic mechanical systems, and machine learning.

**Charles C. Kemp** is an Associate Professor at the Georgia Institute of Technology in the Department of Biomedical Engineering. He also has 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.E., and B.S. from MIT. In 2007, he founded the Healthcare Robotics Lab at Georgia Tech (http://healthcare-robotics.com), which focuses on mobile manipulation and human-robot interaction. He is an active member of the Institute for Robotics & Intelligent Machines (IRIM) and Georgia Tech's multi-disciplinary Robotics Ph.D. program. He has received the 3M Non-tenured Faculty Award, the Georgia Tech Research Corporation Robotics Award, and the NSF CAREER award.