Fast Reaching in Clutter While Regulating Forces Using Model Predictive Control

Marc D. Killpack and Charles C. Kemp

***Abstract*— Moving a robot arm quickly in cluttered and unmodeled workspaces can be difficult because of the inherent risk of high impact forces. Additionally, compliance by itself is not enough to limit contact forces due to multi-contact phenomena (jamming, etc.). The work in this paper extends our previous research on manipulation in cluttered environments by explicitly modeling robot arm dynamics and using model predictive control (MPC) with whole-arm tactile sensing to improve the speed and force control. We first derive discrete-time dynamic equations of motion that we use for MPC. Then we formulate a multi-time step model predictive controller that uses this dynamic model. These changes allow us to control contact forces while increasing overall end effector speed. We also describe a constraint that regulates joint velocities in order to mitigate unexpected impact forces while reaching to a goal. We present results using tests from a simulated three link planar arm that is representative of the kinematics and mass of an average male's torso, shoulder and elbow joints reaching in high and low clutter scenarios. These results show that our controller allows the arm to reach a goal up to twice as fast as our previous work, while still controlling the contact forces to be near a user-defined threshold.**

I. Introduction

The current state of robot manipulation capabilities lags far behind both human speed and capability. Meanwhile, an aging world population and events such as the nuclear disaster at Fukushima, Japan, have led to an increased call for robotic systems for assistive applications, disaster response, and search and rescue. However, many current approaches to robot control are most applicable in highly structured environments such as factories. Although this design has resulted in effective obstacle avoidance and precise motion control for industrial robots, these methods do not generalize well to unstructured applications where unmodeled impact and contact can occur.

In this paper we describe our formulation of a multi-step model predictive controller (MPC) that uses a dynamic model of the arm. This model incorporates simple impedance control at the joints and whole-arm tactile sensing. We use this model to explicitly control the joint states and contact forces for a three degree of freedom simulated planar robot arm modeled after the mass and kinematics of a 50^{th} percentile North American male (see Figure 1). We build and improve on our previous work on manipulating in clutter (see [1]) by decreasing the time to reach a goal using our new dynamic model formulation. In addition, we introduce a novel impact force constraint that is predictive in nature and allows an explicit trade-off between end effector velocity and allowable impact forces.

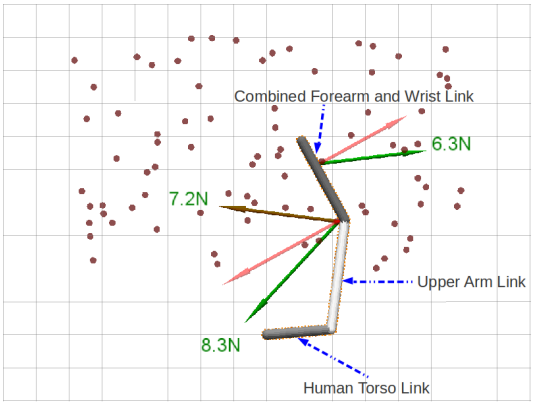

Fig. 1: Screenshot of our software simulation testbed. There are 80 red obstacles that are rigid and fixed. Red arrows show the sensed normal force while green arrows show the total resultant contact force.

II. Related Work

A. Multi-contact Manipulation

Contrary to most common geometric approaches to manipulating in clutter that require accurate 3D models of the world and expect contact only at the end effector [2]–[4], our approach is based on haptic feedback across the whole robot arm. Other work which more closely addresses the problem we are investigating presents a framework for controlling a robot with multiple contacts along the links [5]. That work along with work in [6], [7] models contact as being rigid and assumes that the robot has at least six degrees of freedom (DoF) between each contact and the torso of the robot. However, unexpected contact or manipulation in cluttered environments is not addressed. Other research has shown that it is possible to make contact at multiple locations to perform a task, but these also require complete geometric models of the world, assume rigid contact, or do not use sensor feedback during the planned actions [8]–[10]. Most related to our approach is recent work which uses an optimal control formulation and explicit contact modeling to perform multi-contact tasks [11]–[13]. However, this work generates open-loop trajectories and has currently not been applied to online feedback control for robot manipulation.

B. Collision Detection and Reaction

The majority of research for unwanted or unmodeled collisions in robotics has focused on reacting to impacts once collision has occurred [14]–[16]. This includes work which uses novel sensing technology to quantify forces during impact [17] and model the instantaneous stiffness effects

during collision [18]. Other work develops mathematical models to show the effect of compliance on impact forces [19]. Extensive work has sought to quantify the potential for injury when robots impact humans [20] and some work has been done recently to limit joint velocities accordingly [21]. However, in our work we use a novel constraint to regulate joint velocities so as to limit unexpected impact forces while still reaching the goal more quickly than if we limited all joint velocities uniformly.

C. Model Predictive Control in Robotics

One of the initial application areas for model predictive control (MPC) was chemical process control [22]. MPC is also generally referred to as receding horizon control such as in recent work on the control of aerial vehicles [23], [24]. MPC has also been used in robot locomotion research (e.g, [25]–[27]). In terms of robot manipulation, MPC has recently been used in applications such as bouncing a ball [28], generating manipulator trajectories to compensate for inertial forces on a boat [29], controlling a 6 DoF cable-driven parallel manipulator [30], and reaching in free space [31].

III. Controller Formulation

In this section, we present the basic mathematical models that we use to formulate our model predictive controller which uses haptic (tactile and torque) feedback to maneuver in clutter. For all mathematical formulas, we use bold face letters to represent matrices or vectors. We start with a joint space Lagrangian formulation for a serial torque controlled robot manipulator (see [32], [33]):

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T \boldsymbol{f}_i^{ext} + \boldsymbol{\tau}_{control} \quad (1)$$

The variables $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are joint angles, velocities and accelerations respectively and all terms in Equation 1 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 velocity dependent friction. The terms on the right hand side of Equation 1 represent the control torques ($\tau_{control}$) and the external torques ($\sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T \boldsymbol{f}_i^{ext}$) due to external forces applied by the environment. 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 $\boldsymbol{J}_{c_i}$ is the geometric contact Jacobian at that contact location. The contact Jacobian is calculated in the same manner as a traditional geometric Jacobian at a given location except that the columns related to all joints distal to the contact are zeroed because they have no direct effect on the contact. The contact forces and locations are measured by using whole-arm tactile sensing skin that covers our simulated robot arm.

All controllers in this proposed work are built on top of "simple joint impedance control" [34], [35] which we

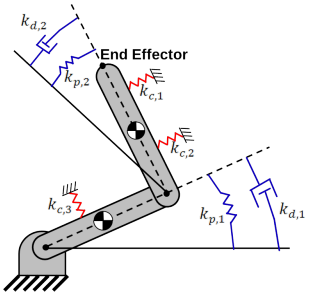

Fig. 2: Graphical representation of the dynamics that are included in our robot model. The links have mass and rotational inertia. The blue elements represent the simple impedance control at the joints and the red springs represent contact with the world.

model explicitly in our controller formulation. We use this method due to its inherent compliance and stability while making contact in unknown environments. Equation 2 shows the definition of how the control torques at the joints are generally calculated for this kind of joint impedance control.

$$\boldsymbol{\tau}_{control}(\boldsymbol{K}_p, \boldsymbol{K}_d, \boldsymbol{q}_{des}) = \boldsymbol{K}_p(\boldsymbol{q}_{des} - \boldsymbol{q}) - \boldsymbol{K}_d\dot{\boldsymbol{q}} + \hat{\boldsymbol{G}}(\boldsymbol{q}) \quad (2)$$

We define the desired joint angle $\boldsymbol{q}_{des}$ as the angle to which the joint would settle in the absence of external forces with perfect gravity compensation. The inputs to determine the applied control torques are $\boldsymbol{K}_p, \boldsymbol{K}_d, \boldsymbol{q}_{des}$. Where $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are m by m dimensional diagonal matrices and m is the number of joints on the robot. The term $\hat{\boldsymbol{G}}$ denotes an estimate of the torques required for gravity compensation and we assume that it compensates for $\boldsymbol{G}(\boldsymbol{q})$ in Eqn. 1. Our low-level impedance control runs at 1 kHz while our model predictive controller runs at 100 Hz.

For this work, we neglect joint friction ($\boldsymbol{F}(\dot{\boldsymbol{q}}) = 0$). We assume that each contact can be represented as a single point contact. Additionally, we assume that the change in force over time for that contact can be represented as the reaction force from interacting with a linear spring. This model of the robot arm is represented graphically in Figure 2. We approximate the variable $\boldsymbol{f}_i^{ext}$ using the following relation:

$$\boldsymbol{f}_i^{ext} \approx \boldsymbol{f}_i^{measured} + \boldsymbol{K}_{c_i}\boldsymbol{J}_{c_i}(\boldsymbol{q} - \boldsymbol{q}_0) \quad (3)$$

Here $\boldsymbol{f}_i^{measured}$ is the current measurement of the normal contact force at the i^{th} location, $\boldsymbol{K}_{c_i}$ is the Cartesian stiffness of the contact at i, and $\boldsymbol{q}_0$ is the starting configuration of the arm right after we obtain $\boldsymbol{f}_i^{measured}$.

We combine equations 1, 2, and 3, and rearrange to obtain a state space representation which gives the following:

$$\begin{bmatrix} \ddot{\boldsymbol{q}} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \boldsymbol{A}\begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{q} \end{bmatrix} + \boldsymbol{B}\begin{bmatrix} \boldsymbol{q}_{des} \\ \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T \boldsymbol{f}_i^{measured} \\ \boldsymbol{q}_0 \end{bmatrix} \quad (4)$$

where

$$\boldsymbol{A} = \begin{bmatrix} -\boldsymbol{M}(\boldsymbol{q})^{-1}(\boldsymbol{K}_d + \boldsymbol{C}(\dot{\boldsymbol{q}}, \boldsymbol{q})) & -\boldsymbol{M}(\boldsymbol{q})^{-1}(\boldsymbol{K}_p + \sum_{i=1}^{N}(\boldsymbol{J}_{c_i}^T \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i})) \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \quad (5)$$

$$\boldsymbol{B} = \boldsymbol{M}(\boldsymbol{q})^{-1} \begin{bmatrix} \boldsymbol{K}_p & \boldsymbol{I} & \sum_{i=1}^{N}(\boldsymbol{J}_{c_i}^T \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i}) \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \quad (6)$$

In this case, the matrices $\boldsymbol{I}$ are appropriately sized identity matrices. We derive the symbolic form of the mass matrix, Coriolis, and gravity terms using a symbolic python library (see [36] for an older version of this library). The above equations allow us to discretize our system using the matrix exponential (found in [37]) and formulate the following discrete-time state space equations:

$$\begin{bmatrix} \dot{\boldsymbol{q}}[t+1] \\ \boldsymbol{q}[t+1] \end{bmatrix} = \boldsymbol{A}_d[t] \begin{bmatrix} \dot{\boldsymbol{q}}[t] \\ \boldsymbol{q}[t] \end{bmatrix} + \boldsymbol{B}_d[t] \begin{bmatrix} \boldsymbol{q}_{des}[t] \\ \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T \boldsymbol{f}_i^{measured}[t_0] \\ \boldsymbol{q}[t_0] \end{bmatrix} \quad (7)$$

In these equations, $[t]$ and $[t+1]$ represent the current and next time steps. The state space matrices $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ are not constant because they are dependent on terms like the mass matrix which may change at every time step. We define a relation for the desired joint angles such that:

$$\boldsymbol{q}_{des}[t] = \boldsymbol{q}_{des}[t-1] + \Delta \boldsymbol{q}_{des}[t] \quad (8)$$

Accordingly, $\boldsymbol{q}_{des}$ becomes a state variable and $\Delta \boldsymbol{q}_{des}$ becomes the available control input.

These equations are exact if the Taylor series is infinite and the inputs do not change over the predicted time step. However, we use a Padé approximation [38] to represent the matrix exponential. In addition, our contact force model is imperfect and we neglect the non-normal force. However, we have found this formulation to give good performance for our initial tests. We use the matrix exponential to discretize because a first order Euler approximation was not as accurate and made control more difficult given our control rate of 100 Hz.

Using the dynamic constraints from Equations 7 and 8, and a cost function described below, we can form a model predictive controller. For our tests we used a time horizon of four discrete steps (meaning $H = 4$), with a control rate of 100 Hz (or a 0.01 second sample time). The procedure is then to solve the associated quadratic programming problem for $\Delta \boldsymbol{q}_{des}$ and apply the resulting control input for the first time step. After solving once and applying the control input, we reformulate the optimization given updated sensor and state values and solve again following the same process at every time step. Equations 9 through 18 show the complete optimization.

Our cost consists of a terminal cost (Eqn. 9) which attempts to move the end effector towards a desired goal position, a cost on forces over a desired force threshold (Eqn. 10), a cost on changing the contact force faster than a specified rate (Eqn. 11), and a cost on control effort (Equation 12). The desired force threshold ($f_{threshold}$) specifies that contact forces below the scalar value are acceptable, while any predicted force above the value incurs a penalty in our cost function.

Originally, the terms related to force were explicit constraints. However, due to feasibility issues with our optimization, we moved these terms into the cost function. The constraints in our current formulation consist of the discrete dynamic equations (Eqns 13 and 14), limits on joint actuation and angles (see Eqns. 15, 16, 17) and a joint velocity constraint detailed in Section III-A. The variable Δx_{des} is a waypoint that has a fixed step size in a straight line towards the goal unless the distance to the goal position is smaller than the step size. We have found that in practice, our new controller is able to control forces without any slewing of intermediate goal positions even if the goal position is physically unattainable during the short time horizon used for prediction. Instead the step size affects success rate and end effector speed but has little effect on contact forces or overall stable motion of the arm as discussed later in the paper.

A. Impulse-Momentum Impact Model

The term on the left side of Equation 18 represents a perfectly elastic collision which would result in the highest impact torque (as the link is completely decelerated and then accelerated to the same speed in the opposite direction). This joint space impulse model has been used before in work on walking robots [39]. The parameter Δt_{impact} is the time duration of the impact required to have the change in momentum on the left side and we tune it empirically according to the environment. The terms of this equation form an impulse-momentum constraint that limits the joint velocities in order to limit resultant impact forces. This initial formulation is conservative as it assumes perfectly elastic collisions and that each link of the arm is not decoupled. This may not be the case as the joint coupling is dependent on the arm configuration and impedance control parameters at the joints.

In order to determine τ_{max}, we relate the desired contact force threshold to a maximum torque by finding the approximate maximum moment arm from a joint to any distal position on the arm. We then calculate the resultant torque if the arm experiences the desired contact force threshold at that point. The calculated torque value changes if we change the desired contact force threshold as is the case in our tests. We specify the direction of the force as being normal to r_{max} (the distance from the point to the joint in question) and the joint axis. Figure 3 shows a visual representation of how τ_{max} is calculated for the first joint.

To solve this optimization at every time step, we use recent advances in optimization for embedded control to formulate our controller and generate efficient C code (see [40] for description of web-based CVXGEN).

$$\underset{\boldsymbol{q},\dot{\boldsymbol{q}}}{\textbf{minimize}}$$

$$\alpha \left\| \Delta \boldsymbol{x}_{des} - \boldsymbol{J}_{ee}(\boldsymbol{q}[t_0 + H + 1] - \boldsymbol{q}[t_0]) \right\|^2 \tag{9}$$

$$+\beta \sum_{t=t_0}^{t_0+H} \sum_{i=1}^{N} \max\left(\boldsymbol{n}_{c_i}^T \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i} (\boldsymbol{q}[t+1] - \boldsymbol{q}[t_0]) - (f_{threshold} - \left\| \boldsymbol{f}_i^{measured}[t_0] \right\|), \boldsymbol{0} \right) \tag{10}$$

$$+\zeta \sum_{t=t_0}^{t_0+H} \sum_{i=1}^{N} \max\left(abs(\boldsymbol{n}_{c_i}^T \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i} (\boldsymbol{q}[t+1] - \boldsymbol{q}[t])) - \Delta f_{rate,i}, \boldsymbol{0} \right) \tag{11}$$

$$+\mu \textstyle\sum_{t=t_0}^{t_0+H} \left\| \Delta \boldsymbol{q}_{des}[t] \right\|^2 \tag{12}$$

$$\textbf{subject to} : (for\ t = t_0 \ldots t_0 + H)$$

$$\begin{bmatrix} \dot{\boldsymbol{q}}[t+1] \\ \boldsymbol{q}[t+1] \end{bmatrix} = \boldsymbol{A}_d[t] \begin{bmatrix} \dot{\boldsymbol{q}}[t] \\ \boldsymbol{q}[t] \end{bmatrix} + \boldsymbol{B}_d[t] \begin{bmatrix} \boldsymbol{q}_{des}[t] \\ \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T \boldsymbol{f}_i^{measured}[t_0] \\ \boldsymbol{q}[t_0] \end{bmatrix} \tag{13}$$

$$\boldsymbol{q}_{des}[t+1] = \boldsymbol{q}_{des}[t] + \Delta \boldsymbol{q}_{des}[t] \tag{14}$$

$$\boldsymbol{q}[t+1] \leqq \boldsymbol{q}_{max} \tag{15}$$

$$\boldsymbol{q}[t+1] \geqq \boldsymbol{q}_{min} \tag{16}$$

$$abs(\Delta \boldsymbol{q}_{des}[t]) \leqq \Delta \boldsymbol{q}_{max,des} \tag{17}$$

$$abs(2\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}[t+1]) \leqq \boldsymbol{\tau}_{max} \Delta t_{impact} \tag{18}$$

NOMENCLATURE

$\alpha, \beta, \zeta, \mu$ Scalar weighting terms for the multi-objective cost function
$\boldsymbol{\tau}_{max}$ Maximum allowable torque due to impact forces
$\boldsymbol{A}_d, \boldsymbol{B}_d$ Discrete time linear approximations of the system state space matrices
$\boldsymbol{f}_i^{measured}$ Measured normal force for contact i
$\boldsymbol{J}_{c_i}$ Geometric Jacobian at contact i
$\boldsymbol{J}_{ee}$ Geometric Jacobian at the end effector
$\boldsymbol{K}_{c_i}$ Cartesian stiffness matrix for contact i
$\boldsymbol{n}_{c_i}$ Contact normal direction at contact i
$\boldsymbol{q}, \dot{\boldsymbol{q}}$ State variables of joint angle and velocity
$\boldsymbol{q}_{des}$ Commanded joint angles that are sent to the joint impedance controller
$\boldsymbol{q}_{max}$ Maximum joint angle limits
$\boldsymbol{q}_{min}$ Minimum joint angle limits
$\Delta \boldsymbol{q}_{des}$ Change in commanded joint angles, this is the output of our MPC
$\Delta \boldsymbol{q}_{max,des}$ Maximum allowable change in commanded joint angle
$\Delta \boldsymbol{x}_{des}$ Desired change in position at the end effector
$\Delta f_{rate,i}$ Max desired rate at which the contact force should change at contact i
Δt_{impact} Time duration of an expected impact
$f_{threshold}$ User-defined allowable contact force threshold
H Number of time steps in the prediction model
t_0 Current time for which state and other measurements are valid, this is a starting point for the predictive model

IV. METHODOLOGY AND TESTING

We used the open source physics simulation library Open Dynamics Engine (ODE) [41] to simulate reaching in clutter. ODE solves for unilateral contact constraints using a Linear Complementarity Problem (LCP) formulation. We used a planar arm with three rotational joints and joint space impedance control, as well as "skin" (or discrete tactile sensing elements) covering the entire surface of the arm at a resolution of one sensor per centimeter. This arm represents the form of a humanoid robot and the masses and kinematics are representative of an average North American male. We simulated tactile sensing elements (taxels) as only sensing the component of the applied force that is normal to the surface which is what we used for control. However, for all metrics that we reported in terms of contact forces, we used the full magnitude of the simulated contact force. Figure 1 shows a screenshot of the software simulation testbed and arm which is representative of a person with three degrees of freedom operating in the plane. The environment is composed of rigid cylindrical obstacles that for our tests were all rigidly fixed. Because the arm is planar, the cylinders can be thought of as disks or circles in a 2D plane. These obstacles have a

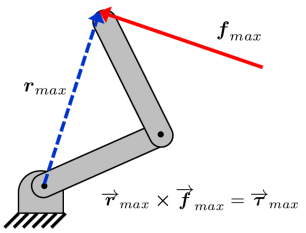

Fig. 3: Visualization of joint space impulse-momentum constraint for the first joint of a two link arm.

diameter of 2 cm. We used randomly generated environments with 20 and 80 fixed objects (see Fig. 1). This allowed us to test a large number of possible multi-contact conditions as the arm reached through the objects. For each environment, we reached to eight pre-specified goal locations using the same clutter environments as in our previous work [1].

Our initial design methodology involved setting the gain on the position cost term first (Eqn. 9) 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 size made tuning the controller difficult and sometimes unintuitive. In order to search the parameter space of our controller and acquire some intuition, we used simulated annealing on 15 randomly selected environments in ODE (5 with 20 fixed objects and 10 with 80 fixed objects). We ran these trials with two different maximum desired force thresholds of 5 N and 15 N, and accumulated the cost across all 30 trials. This optimization resulted in a Pareto front that gave us a better intuition for the trade-off between parameters while at the same time improving performance significantly. The cost we used was the following:

$$\begin{aligned} cost \quad = \quad & \sum_{i=k}^{30} [1000(f_{k,max} - f_{k,threshold}) \\ & +1000(f_{k,percent}) + 10(t_{k,duration}) \\ & +J_{k,result}] \end{aligned} \tag{19}$$

Where k represents an individual trial, $f_{k,max}$ is the maximum measured force in Newtons for a given trial, $f_{k,threshold}$ is the user-specified force threshold, $t_{k,duration}$ is the length of time in seconds of the trial, $f_{percent}$ is the percent of all contact forces over the threshold, and $J_{k,result}$ is a very large value (25,000) incurred for failure to reach the goal. The first two cost terms are only used if the threshold is exceeded for a given trial. The weightings are not used to normalize the cost terms in any sense, but were manually tuned such that the trade off between cost terms (especially 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 reasonable. Other methods such as multi-objective genetic algorithms can also be used to get similar results and in general the only criterion for our parameter tuning optimization was that it allow the definition of heuristic based costs instead of analytical costs due to the nature of our simulation trials.

Applying this on a real humanoid robot (where simulation is not an option) will be more difficult. However, we believe that our simulation approach can inform or restrict the search space when tuning for a real robot. As expected, we found that the parameters associated with our impulse-momentum impact model had the largest affect on the maximum contact forces experienced. However, the size of the waypoint also altered the success rate significantly such that above 10 cm the success rate was much lower than below 10 cm. We believe that this has to do with the the isocontours of the distance-to-goal cost such that smaller magnitude waypoints allow the arm to move around obstacles in a non-greedy fashion. The final parameters that we used for all tests are found in Table I. We identified these parameters by running this optimization many times which gave better coverage of the search space using multiple restarts. After we obtained a Pareto front, we identified candidate parameter sets that had approximately equal total cost but varied according to their performance with respect to each term in the cost function. We then ran a larger subset of trials on these parameters sets and picked the best one in terms of total cost. We used $\Delta t_{impact} = 0.02$ for the higher force threshold of 25 N and 0.04 for the 5 N threshold.

TABLE I: Controller parameters identified by our simulated annealing optimization.

Parameters	Value
α	275 (1/m^2)
β	229 (1/N)
ζ	0.01 (1/N)
μ	34.8 (1/rad^2)
Δt_{impact}	0.04 or 0.02 (s)
$f_{rate,i}$	10 (N)
$waypoint$	0.04 (m)

We also performed tests using the Matlab Robotics Toolbox [42] with Matlab's native numerical integration algorithms. Our Matlab implementation uses an explicit spring-damper contact model. The contact stiffness for these trials was 10,000 N/m whereas the stiffness for the rigid ODE objects was approximately 48,000 N/m. Our simulated robot arm has discrete tactile sensing elements that are spaced one centimeter apart along each link to which we discretize any simulated contact.

In all Matlab trials, we used a desired contact force threshold of 5 N. The task was specified as reaching to a goal location while controlling contact forces and we used the same stopping criteria for success or failure as reported in [1].

A. Success Rates, Time to Complete, and Contact Forces

In order to compare the performance of our new dynamic MPC to our previous work, we ran four sets of 1,200 trials each for the new controller and for our quasi-static controller from [1] in ODE. The four settings were specified by varying the density of clutter (20 objects and 80 objects) and the value of the force threshold (5 N and 25 N). For the 5 N threshold we used the same data for the quasi-static controller reported in [1]. However, for the 25 N threshold,

TABLE II: Summary statistics for comparing success rate, velocity and contact force control between our dynamic and quasi-static controllers at different densities of clutter and for different force thresholds.

		Low Clutter (20 objects)		High Clutter(80 objects)	
		dynamic model	quasi-static model	dynamic model	quasi-static model
High Force Threshold (25 N)	avg. time to complete (sec)	10.3	22.1	11.6	21.4
	success rate	76.7%	72.3%	30.0%	23.3%
	99 percentile contact force value (N)	30.5	27.6	29.1	27.8
		dynamic model	quasi-static model	dynamic model	quasi-static model
Low Force Threshold (5 N)	avg. time to complete (sec)	20.1	21.9	18.3	20.6
	success rate	70.8%	77.3%	28.3%	28.3%
	99 percentile contact force value (N)	5.6	7.9	8.0	7.9

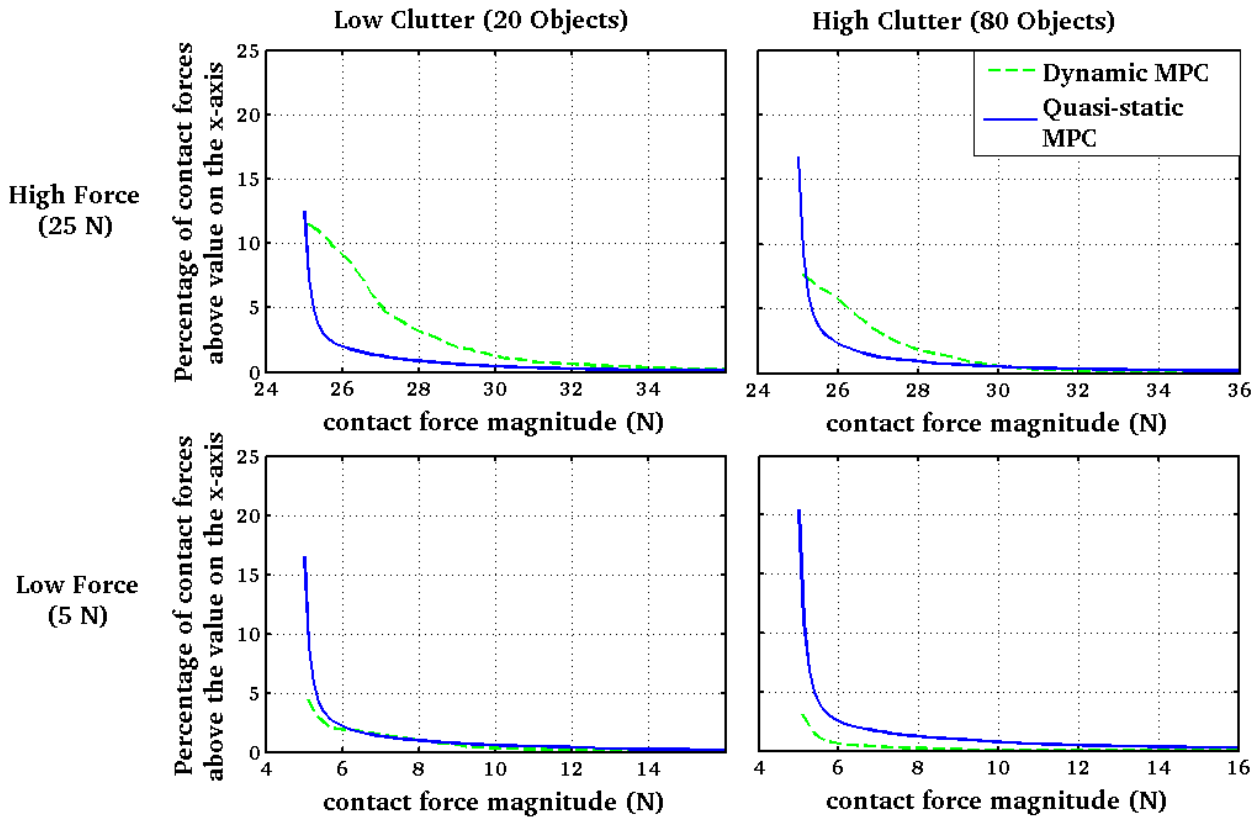

Fig. 4: These graphs shows how the percent of overall contact forces above the force value in the x-axis decrease as the force value increases.

we generated new results for the quasi-static controller using the same controller implementation and parameters from [1].

We used three main criteria for comparing our new controller that uses multiple time steps and a dynamic model to our quasi-static MPC. The first was overall success rate of reaching to a goal position through the simulated clutter. The second was the average time to complete the task for the intersection of successful trials for the two controllers. The third measure was comparing the ability of the two controllers to keep contact forces below the specified force threshold. Table II summarizes these comparisons.

In terms of success rates, the dynamic controller performed as well or better than the quasi-static controller for three out of the four scenarios. We also obtained the success rates for trajectories executed using a Bi-directional RRT [43] as we did in our previous work [1]. The planner has all knowledge of the environment and can be thought of as an upper bound on feasible success rates. For 20 objects the planner had a 96.7% success rate and for 80 objects the planner had a 48.3% success rate. In [1] we showed that multiple greedy reaches with our local quasi-static controller approached the success rates that the planner only achieved with full knowledge of the environment.

The table also shows that in all cases the dynamic controller completes the task faster than the quasi-static controller. The percent improvement in speed ranges from 8% to over 200% with all values being statistically significant (α = 0.05) using a paired t-test. Interestingly, at the force value specified by the force threshold (of either 5 N or 25 N) the dynamic controller always resulted in a lower percentage of the sensed contact forces being above the threshold than the quasi-static controller. However, the 99^{th} percentile contact

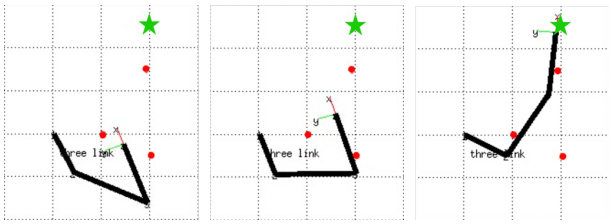

Fig. 5: Three intermediate arm configurations showing the goal location (green star) and object configuration (red circles) that we used for tests in Section IV-B.

force magnitude (the value below which 99% of all contact forces remains) was dependent on the clutter density and force threshold. In all cases, the dynamic MPC had a lower force value above which it had no more contact forces than our original quasi-static controller meaning that it better limited the maximum contact forces. Figure 4 shows the percentage of contact forces above the force value shown on the x-axis. Both controllers are able to successfully limit contact force according to the force threshold specification.

B. Effect of Impulse-Momentum on Impact Force Control

One way to reduce impact forces is to limit all joint velocities. However, if we plan for the worst case impact scenario, this would cause the arm to move unnecessarily slowly in many joint configurations. For this reason, we formulated the impulse-momentum constraint in Equation 18. To test the effect of this constraint, we varied the Δt_{impact} and evaluated performance in Matlab simulations. We let the value for Δt_{impact} equal 0.08 seconds and 5.0 seconds. With a value of 5.0, the impulse-momentum constraint is effectively inactive throughout the arm trajectory. Our environment had three fixed objects with which our simulated arm made contact as it reached to a goal position. Figure 5 shows the goal, obstacles and arm trajectory. The objects have a stiffness of 10,000 N/m and a damping value of 10 Ns/m.

The maximum measured force at every time step for the different values of Δt_{impact} are plotted in Figure 6. The max force in the worst case (with inactive constraint) is 14.1 N whereas in the case where $\Delta t_{impact} = 0.08$, the maximum force is only 5.26 N. That gives a 268% improvement in force control. This makes sense as the constraint limits the joint velocities according to unpredicted impact forces. We can see from Figure 6 that the controller with $t_{impact} = 0.08$ makes the first contact at a later time (approximately 0.61 seconds later). However, what is not clear from this figure is the effect of this constraint on the end effector velocity.

Although this constraint will slow the speed of the end effector in making progress towards the goal, it allows us to make a structured trade-off. Specifically, Figure 7 shows the end effector velocity for the two parameters of Δt_{impact} that we tested. The point at which the velocities for these trials settles to zero is where they have reached the goal. The end effector velocity for the larger parameter value is over three times higher at certain points in the trajectory, but it only settles to the goal approximately 3.1 seconds sooner.

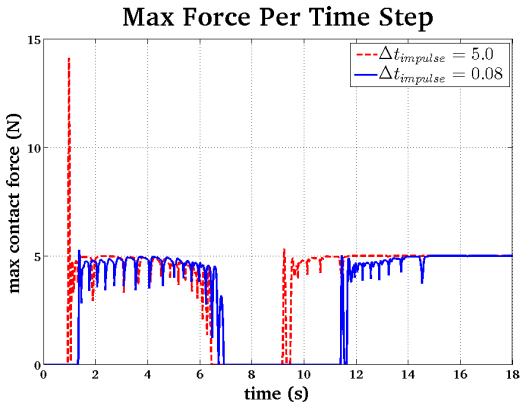

Fig. 6: This plot shows the effect of varying the impact time parameter on the maximum force measured at each time step. For $\Delta t_{impulse} = 5$ the constraint is essentially inactive.

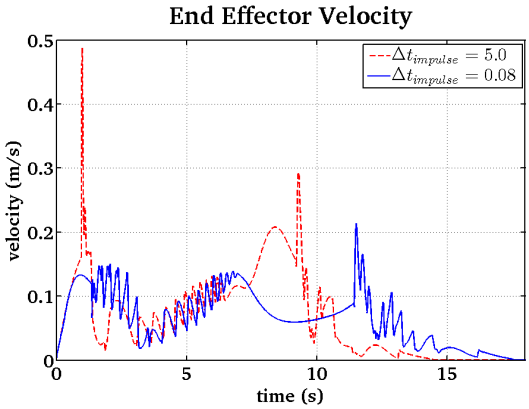

Fig. 7: Effect of varying the impact time parameter on the end effector velocity and time to complete the trajectory.

This means that for the smaller values of Δt_{impact} we have a 268% improvement in impact force control in return for only a 21.1% increase in time to complete.

C. Stability and Robustness

We have performed initial tests by varying the mass of the simulated robot $\pm 30\%$ as well as varying jitter in the control loop by up to three times the control rate. This testing, along with the 4,800 trials that we ran in ODE gives substantial empirical evidence that our controller is stable. In general, although this controller is still a greedy and suboptimal controller (due to the short time horizon), the model prediction enables the simulated humanoid to reach to the goal faster than our previous work with comparable force control.

V. Conclusion

Our work shows the possibility of reaching at faster rates into cluttered environments while controlling velocities, forces, and mitigating effects of unexpected impact in a planar simulated environment. We have shown that we can include an impulse-momentum model of impact that allows us to trade end effector velocity for impact forces. Additionally, unlike many other controllers using a dynamic model of a robot arm, our controller appears to be robust to model

variation which shows great promise for implementation on a real robot. We also ran tests that show our controller can run consistently around 50 Hz for robot arms with more degrees of freedom than our planar arm using an off-the-shelf Intel Core i7-3632QM CPU. Our approach and work has allowed us to increase the speed at which our simulated robot can reach in clutter while still regulating individual contact forces in multi-contact situations.

ACKNOWLEDGMENTS

We gratefully acknowledge support from DARPA Maximum Mobility and Manipulation (M3) Contract W911NF-11-1-603. We thank Jacob Mattingley and Stephen Boyd for their support with CVXGEN and Cristóvão Sousa for his support with the symbolic robotics library Sympybotics. We also thank Jeffrey Hawke and Tapomayukh Bhattacharjee for technical discussions on this work.

REFERENCES

[1] A. Jain, M. D. Killpack, A. Edsinger, and C. Kemp, "Reaching in clutter with whole-arm tactile sensing," *The International Journal of Robotics Research*, 2013.

[2] M. Dogar and S. Srinivasa, "A framework for push-grasping in clutter," *Robotics: Science and Systems*, 2011.

[3] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *IEEE Int. Conf. on Robotics and Automation*, 2007.

[4] M. Stilman, K. Nishiwaki, S. Kagami, and J. Kuffner, "Planning and executing navigation among movable obstacles," *Advanced Robotics*, vol. 21, no. 14, pp. 1617–1634, 2007.

[5] J. Park and O. Khatib, "Robot multiple contact control," *Robotica*, vol. 26, no. 5, 2008.

[6] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, 2005.

[7] L. Sentis, J. Park, and O. Khatib, "Compliant Control of Multicontact and Center-of-Mass Behaviors in Humanoid Robots," *IEEE Transactions on Robotics*, vol. 26, no. 3, 2010.

[8] S. Legagne, A. Kheddar, and E. Yoshida, "Generation of Optimal Dynamic Multi-Contact Motions : Application to Humanoid Robots," *IEEE Transactions on Robotics – under review*, 2011.

[9] K. Hauser, T. Bretl, and J. Latombe, "Non-gaited humanoid locomotion planning," in *Humanoids*, 2005.

[10] A. Escande and A. Kheddar, "Contact planning for acyclic motion with tasks constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009.

[11] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics." IROS, 2012.

[12] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 43, 2012.

[13] I. Mordatch, Z. Popovic, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, 2012, pp. 137–144.

[14] A. De Luca and R. Mattone, "An adapt-and-detect actuator FDI scheme for robot manipulators," in *ICRA*, 2004.

[15] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the DLR-III lightweight manipulator arm," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. Ieee, 2006, pp. 1623–1630.

[16] S. Haddadin, A. Albu-Schaffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *IROS*, 2008.

[17] S. Phan, Z. F. Quek, P. Shah, D. Shin, Z. Ahmed, O. Khatib, and M. Cutkosky, "Capacitive skin sensors for robot impact monitoring," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2992–2997.

[18] D. Shin, Z. F. Quek, S. Phan, M. Cutkosky, and O. Khatib, "Instantaneous stiffness effects on impact forces in human-friendly robots," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 2998–3003.

[19] S. Haddadin, K. Krieger, N. Mansfeld, and A. Albu-Schaffer, "On impact decoupling properties of elastic robots and time optimal velocity maximization on joint level," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5089–5096.

[20] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Safe physical human-robot interaction: Measurements, analysis and new insights," *Robotics Research*, pp. 395–407, 2011.

[21] S. Haddadin, S. Haddadin, A. Khoury, T. Rokahr, S. Parusel, R. Burgkart, A. Bicchi, and A. Albu-Schaffer, "A truly safely moving robot has to know what injury it may cause," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5406–5413.

[22] C. Garcia, D. Prett, and M. Morari, "Model predictive control: Theory and practice–a survey," *Automatica*, 1989.

[23] J. Bellingham, A. Richards, and J. How, "Receding horizon control of autonomous aerial vehicles," in *American Control Conference*, 2002.

[24] P. Abbeel, A. Coates, and A. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, 2010.

[25] P. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *IEEE-RAS International Conference on Humanoid Robots*, 2006.

[26] I. Manchester, U. Mettin, F. Iida, and R. Tedrake, "Stable dynamic walking over uneven terrain," *The International Journal of Robotics Research*, 2011.

[27] T. Erez, Y. Tassa, and E. Todorov, "Infinite-horizon model predictive control for periodic tasks with contacts," in *Robotics: Science and Systems (RSS)*, 2011.

[28] P. Kulchenko and E. Todorov, "First-exit model predictive control of fast discontinuous dynamics: Application to ball bouncing," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[29] P. From, J. Gravdahl, T. Lillehagen, and P. Abbeel, "Motion planning and control of robotic manipulators on seaborne platforms," *Control engineering practice*, 2011.

[30] V. Duchaine, S. Bouchard, and C. Gosselin, "Computationally efficient predictive robot control," *Mechatronics, IEEE/ASME Transactions on*, vol. 12, no. 5, pp. 570–578, 2007.

[31] S. Ivaldi, M. Fumagalli, F. Nori, M. Baglietto, G. Metta, and G. Sandini, "Approximate optimal control for reaching and trajectory planning in a humanoid robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[32] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer, 2011.

[33] J. J. Craig, "Introduction to robotics: mechanics and control," 2004.

[34] N. Hogan and S. Buerger, *Impedance and Interaction Control*. Robotics and Automation Handbook, 2005, ch. 19.

[35] N. Hogan, "Impedance control: An approach to manipulation," in *American Control Conference, 1984*. IEEE, 1984, pp. 304–313.

[36] C. a. D. Sousa and R. Cortesão, "Sagerobotics: open source framework for symbolic computation of robot models," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 262–267. [Online]. Available: http://doi.acm.org/10.1145/2245276.2245329

[37] W. L. Brogan, "Modern control theory, 3rd," 1991.

[38] C. Moler and C. Van Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM review*, vol. 45, no. 1, pp. 3–49, 2003.

[39] J. W. Grizzle, G. Abba, and F. Plestan, "Asymptotically stable walking for biped robots: Analysis via systems with impulse effects," *Automatic Control, IEEE Transactions on*, vol. 46, no. 1, pp. 51–64, 2001.

[40] J. Mattingley and S. Boyd, "Cvxgen: a code generator for embedded convex optimization," *Optimization and Engineering*, pp. 1–27, 2012.

[41] R. Smith *et al.*, "Open dynamics engine," 2011. [Online]. Available: http://www.ode.org

[42] P. I. Corke, "A robotics toolbox for matlab," *Robotics & Automation Magazine, IEEE*, vol. 3, no. 1, pp. 24–32, 1996.

[43] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008.