A Robotic System for Reaching in Dense Clutter that Integrates Model Predictive Control, Learning, Haptic Mapping, and Planning

Tapomayukh Bhattacharjee*, Phillip M. Grice*, Ariel Kapusta*, Marc D. Killpack*,
Daehyung Park*, and Charles C. Kemp

***Abstract*— We present a system that enables a robot to reach locations in dense clutter using only haptic sensing. Our system integrates model predictive control [1], learned initial conditions [2], tactile recognition of object types [3], haptic mapping, and geometric planning to efficiently reach locations using whole-arm tactile sensing [4]. We motivate our work, present a system architecture, summarize each component of the system, and present results from our evaluation of the system reaching to target locations in dense artificial foliage.**

I. Introduction

Mason et al. define clutter as "everything that might limit access to the object" [5], and Merriam-Webster's dictionary states that to clutter a place is "to fill or cover with scattered or disordered things that impede movement or reduce effectiveness" [6]. Research on robotic manipulation in clutter has looked at a number of problems, including searching for a visually identifiable object hidden behind sparse clutter [7], [8], visually inferring which objects to remove from sparse clutter in order to manipulate a visible object [9], and moving a cluttered pile of unknown objects into a bin [10].

In this paper, we focus on the problem of enabling a robot to reach a target in dense clutter. We use the term "dense clutter" to refer to clutter that results in the following challenges:

- Physical Challenge: All solutions require contact with parts of the environment other than the target.
- Perceptual Challenge: Line of sight to the target is completely occluded and inferring how the environment will respond to applied forces requires contact.
- Challenge Due to Disorder: No detailed model of the environment is available prior to encountering the scene.

Humans and other animals readily reach targets in dense clutter, such as foliage, using haptic sensing. We expect that robots capable of haptically reaching targets in dense clutter would perform well in a variety of applications, including assistive robotics [11].

Many approaches to robotic manipulation are poorly matched to the challenges of reaching in dense clutter. For example, 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 [12]–[19]. More generally, robotic manipulation research has frequently emphasized avoiding contact except at the end effector [12], [13], [20]–[22] or other single point contact locations [23], [24], which would unnecessarily limit a robot's actions when low-force contact is benign.

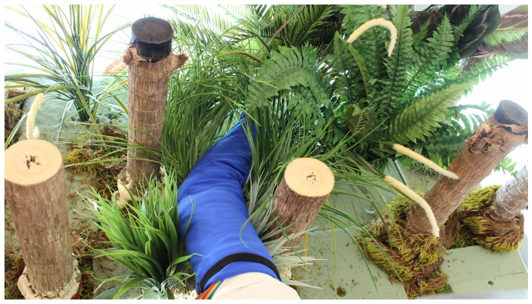
Fig. 1: A robot reaches in dense clutter using our system.

We present a system that enables a robot to reach target locations in dense clutter via joint-angle, joint-torque, and tactile sensing (Fig. 1). Our system first uses computationally efficient, memory-free greedy reaching followed, if necessary, by more resource-intensive geometric planning using a haptic map updated as the robot makes contact with the environment. A motivating intuition for our work is the human experience of reaching to a location with little attention and, upon failing, deliberately reaching with care.

When reaching to 7 distinct target locations in dense artificial foliage (Figs. 1, 7), our system successfully reached its target in 16 out of 21 attempts (76.2%). Notably, the system reached each of the 7 target locations in at least 1 of its 3 attempts.

II. Overview of System Execution

As shown in Algorithm 1, the system first performs a greedy reach using a task-space version of *DynamicMPC* that reaches to a 3D target location [1]. *DynamicMPC* uses model predictive control (MPC) with a collision model and a model of the arm's dynamics to quickly reach to a target with low contact forces. This first reach starts from an arm configuration that is likely to result in a successful reach to the target, as estimated by *LIC1*, where LIC stands for learning initial conditions [2]. If this first attempt fails, the robot performs a second greedy reach from an arm configuration selected by *LIC2* based on the target location, the initial arm configuration for the first reach, and where the first reach became stuck. Throughout this process, the system classifies contact on the arm as leaves or trunk based on tactile sensing and adds the locations of classified contact to its haptic map [3], [4]. If the second greedy reach fails, the system begins

*Listed alphabetically

T. Bhattacharjee, P.M. Grice, A. Kapusta, D. Park, and C.C. Kemp are with the Healthcare Robotics Lab, Georgia Institute of Technology.

M. D. Killpack is now with the Department of Mechanical Engineering, Brigham Young University.

All work performed at the Healthcare Robotics Lab at Georgia Tech.

Algorithm 1 Integrated System Procedure.

```
    function DynamicMPC$_{joint}$( $ArmPath$ || $JointAngles$ )
        if $JointAngles$ then
            ▷ Interpolate from current to goal arm joint angles.
            ▷ Maximum Δq= $1^o$/step along trajectory.
 5:     ▷ Use DynamicMPC to follow given/calculated arm path.
    return $Bool$ success

    function DynamicMPC$_{task}$( $EndEffectorPosition$ )
        ▷ Use DynamicMPC to reach an end-effector position.
    return $Bool$ success

10: function System_Reach( $TargetPosition$ tp )
        ClassifyContact.start()
        $HapticMap$ h ← MapContacts.start()
        $JointAngles$ q$_{LIC1}$ ←LIC1( tp )
        if ¬DynamicMPC$_{JOINT}$( q$_{LIC1}$ ) then return failure
15:     if DynamicMPC$_{TASK}$( tp ) then return success
        $Stuck\_EE\_Position$ s ← $Current\_EE\_Position$
        $LIC1\_Start\_Pose$ sp$_{LIC1}$ ← ForwardKin(q$_{LIC1}$)
        $JointAngles$ q$_{LIC2}$ ←LIC2( tp, sp$_{LIC1}$, s )
        $EEPositionPath$ Path$_{OUT}$ ← Reverse(Path$_{IN}$)
20:     for $GoalPosition$ gp in Path$_{OUT}$ do
            if ¬DynamicMPC$_{TASK}$( gp ) then return failure
        if ¬DynamicMPC$_{JOINT}$( q$_{LIC2}$ ) then return failure
        if DynamicMPC$_{TASK}$( tp ) then return success
        $Bool$ retreated ← False
25:     while not at tp do
            if $ArmPath$ ap ←Plan( tp, h ) then
                if DynamicMPC$_{JOINT}$( ap ) then return success
                if DynamicMPC$_{TASK}$( tp ) then return success
            else if ¬ retreated then
30:             $RetreatPosition$ rp ← ForwardKin( q$_{LIC2}$ )
                for $n = 0$; $n < 3$; n++ do
                    DynamicMPC$_{TASK}$( rp )
                    DynamicMPC$_{JOINT}$( q$_{LIC2}$ )
                    if at q$_{LIC2}$ then retreated ← True
35:                     goto 25
                return failure
            else return failure
        return success
```

planning. It starts a loop that uses an RRT planner and the haptic map to try to find a joint-space trajectory to the target that avoids trunk contact. If the planner succeeds, the robot attempts to follow the returned trajectory using a joint-space version of *DynamicMPC*. If this fails, the robot attempts another task-space greedy reach before planning again. If the planner does not return a trajectory, the robot attempts to greedily pull its arm out of the clutter before planning again. The system continues this loop until the robot reaches the target (success), the robot fails to extract its arm from the clutter (failure), or the planner does not return a trajectory while the arm is outside the clutter (failure).

III. The System

As illustrated by the block diagram in Fig. 2, our system integrates a number of components, which we now describe.

A. *DynamicMPC*

To perform greedy reaching movements, we use a multi-step model predictive controller that explicitly models the robot arm's dynamics and robot-environment contact forces.

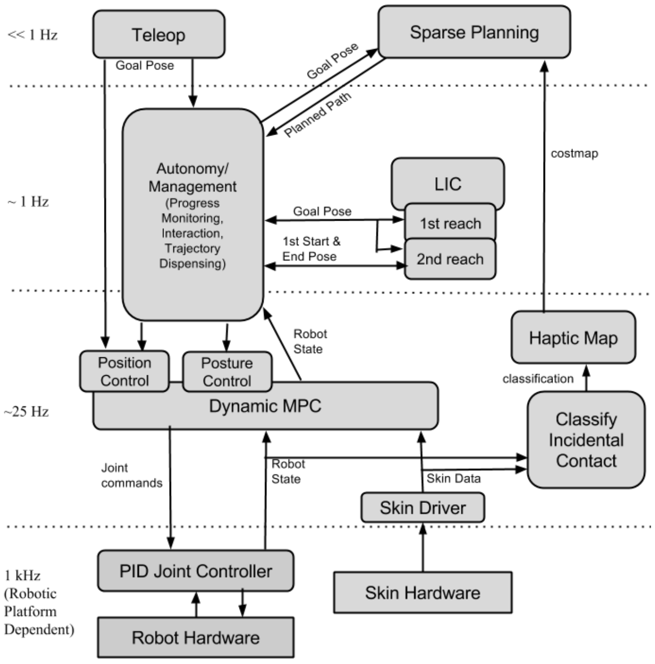

Fig. 2: Block diagram of integrated system architecture.

We use a task-space version of this controller that moves the robot's end effector to a 3D goal location, as described in [1] and [25]. We also developed a new joint-space version of this controller that moves the arm towards a goal joint configuration in order to follow planned joint-space trajectories. Both versions of the controller move to goals while keeping predicted contact forces and potential impact forces low.

The $\sim$1 kHz low-level joint controllers of the Meka M1 arm perform gravity compensation rather than MPC, which runs at $\sim$25 Hz. We added an integral control term to the cost function of both the task-space and joint-space versions of *DynamicMPC* to handle gravity compensation errors on the real robot. The joint-space version of the controller shown in Fig. 3 uses $\boldsymbol{d}_{grav}$, where

$$\boldsymbol{d}_{grav} = \boldsymbol{f}(e_{total}, e_{current}) \tag{1}$$

and

$$e_{total} = k_i \sum_{t=0}^{t_0} (\boldsymbol{q}_{goal} - \boldsymbol{q}[t]) \tag{2}$$

$$e_{current} = \boldsymbol{q}_{goal} - \boldsymbol{q}[t_0]. \tag{3}$$

$\boldsymbol{f}$ performs straightforward anti-windup with saturation limits, and only results in a non-zero value for $\boldsymbol{d}_{grav}$ when the end effector is within 8 cm of the desired goal location to avoid overshoot and high forces when the robot is stuck far from the goal. The integral gain, k_i, is also a small value.

The prediction model has four time steps with active control and four additional time steps during which the control input is set to zero. The cost function seen in Fig. 3 shows the convex optimization performed by the new joint-space version of the controller at each time step using CVXGEN [26]. For this version, we also removed the limit on the rate of change of contact forces as found in [1] and [25] to improve computational performance.

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

$$\alpha \left\| \Delta \boldsymbol{q}_{goal} - (\boldsymbol{q}[t_0 + H_u + H_y + 1] - \boldsymbol{q}[t_0]) - \boldsymbol{d}_{grav} \right\|^2 \tag{4}$$

$$+\beta \sum_{t=t_0}^{t_0+H_u} \sum_{i=1}^{N} \max \left(\boldsymbol{n}_{c_i}^T \boldsymbol{K}_{c_i} \boldsymbol{J}_{c_i} (\boldsymbol{q}[t+1] - \boldsymbol{q}[t_0]) - (f_{threshold} - \left\| \boldsymbol{f}_i^{measured}[t_0] \right\|), \boldsymbol{0} \right) \tag{5}$$

$$+\kappa \sum_{t=t_0}^{t_0+H_u} \sum_{i=1}^{N} \max \left(abs(2\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}[t+1]) - \boldsymbol{\tau}_{max} \Delta t_{impact}, \boldsymbol{0} \right) \tag{6}$$

$$+\mu \textstyle\sum_{t=t_0}^{t_0+H_u} \left\| \Delta \boldsymbol{q}_{des}[t] \right\|^2 \tag{7}$$

subject to : $(for\ t = t_0 \ldots t_0 + H_u + H_y)$

$$\begin{bmatrix} \dot{\boldsymbol{q}}[t+1] \\ \boldsymbol{q}[t+1] \end{bmatrix} = \boldsymbol{A}_d[t] \begin{bmatrix} \dot{\boldsymbol{q}}[t] \\ \boldsymbol{q}[t] \end{bmatrix} + \boldsymbol{B}_d[t] \begin{bmatrix} \boldsymbol{q}_{des}[t] \\ \sum_{i=1}^{N} \boldsymbol{J}_{c_i}^T \boldsymbol{f}_i^{measured}[t_0] \\ \boldsymbol{q}[t_0] \end{bmatrix} \tag{8}$$

$$\boldsymbol{q}_{des}[t+1] = \boldsymbol{q}_{des}[t] + \Delta \boldsymbol{q}_{des}[t] \tag{9}$$

$$\boldsymbol{q}[t+1] \leqq \boldsymbol{q}_{max} \tag{10}$$

$$\boldsymbol{q}[t+1] \geqq \boldsymbol{q}_{min} \tag{11}$$

$$abs(\Delta \boldsymbol{q}_{des}[t]) \leqq \Delta \boldsymbol{q}_{max,des} \tag{12}$$

Fig. 3: The altered form of the controller used for joint configuration posture control

Nomenclature

$\alpha, \beta, \kappa, \mu$	Scalar weighting terms for the multi-objective cost function	t_0	Current time where state measurements are valid. Starting point of predictive model
H_u	Number of time steps in the prediction model where we have control authority	H_y	Number of time steps in the prediction model with the control input set to zero
$\Delta \boldsymbol{q}_{goal}$	Desired change in joint configuration	$\boldsymbol{d}_{grav}$	Error correcting integral term
$f_{threshold}$	User-defined allowable contact force threshold	$\boldsymbol{M}(\boldsymbol{q})$	Configuration dependent robot joint-space inertia matrix
$\boldsymbol{n}_{c_i}$	Contact normal direction at contact i	$\boldsymbol{K}_{c_i}$	Cartesian stiffness matrix for contact i
$\boldsymbol{J}_{c_i}$	Geometric Jacobian at contact i	$\boldsymbol{q}_{min}$	Minimum joint angle limits
$\boldsymbol{q}_{max}$	Maximum joint angle limits	$\boldsymbol{\tau}_{max}$	Maximum allowable torque due to impact forces
Δt_{impact}	Time duration of an expected impact	$\boldsymbol{q}, \dot{\boldsymbol{q}}$	State variables of joint angle and velocity
$\boldsymbol{f}_i^{measured}$	Measured normal force for contact i	$\boldsymbol{q}_{des}$	Commanded joint angles that are sent to the joint impedance controller
$\Delta \boldsymbol{q}_{des}$	Change in commanded joint angles, this is the output of our MPC	$\boldsymbol{A}_d, \boldsymbol{B}_d$	Discrete time linear approximations of the system state space matrices
$\Delta \boldsymbol{q}_{max,des}$	Maximum allowable change in commanded joint angle		

The task-space controller reactively takes advantage of the 4 redundant degrees of freedom (DoF) associated with the task of achieving a 3-DoF end-effector goal position with a 7-DoF robot arm. This enables the robot to snake its arm around obstacles and move rapidly while in contact. In contrast, the joint-space controller uses a goal configuration that includes specific target joint angles for all of the robot's degrees of freedom. In practice, this allows the controller to achieve full 7-DoF configurations and track planned arm trajectories, but greatly reduces the ability of the controller to move around unexpected contact and increases the likelihood of the robot becoming stuck.

B. Learned Initial Conditions

The initial condition of a robot reaching into an environment can significantly influence its chance of success [27]. We use a data-driven approach called LIC from [2] to identify good initial configurations for reaching in clutter. LIC searches for a good initial condition x_0^* using a current situation descriptor and an experience library. It chooses x_0^* as the initial condition that it estimates is most likely to result in a successful reach.

In order to learn good initial conditions, we generated training data using a physics simulation of DARCI reaching into an environment with rigid, fixed floating spheres. The training environment contained 60 fixed-floating spheres, each with a 0.05 m radius, placed in a 0.5 m $\times$ 0.9 m $\times$ 0.6 m volume in front of the simulated DARCI robot (Fig. 4). The training was similar to that in [2]. We used 22,684 reaching trials to generate the training data.

We use *LIC1* to denote choosing an initial arm configuration for the first reach into a new cluttered environment. From the first reach, if the robot is not successful, it is able to obtain observations that can help it choose the initial arm configuration for the second reach. We use *LIC2* to denote the second reach's method for initial condition selection, leveraging observations of the environment. *LIC2*'s observation list includes the initial condition and the final

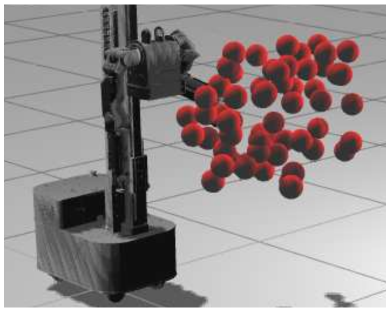

Fig. 4: Training environment in a physics simulator, Gazebo (http://gazebosim.org). Training for *LIC* is performed in simulation prior to the real demonstration. We use 60 fixed-floating spheres with $0.05m$ radius in front of DARCI to simulate a densely cluttered environment.

position of the end effector in the first attempt.

C. Haptic Classification

During manipulation in cluttered environments, incidental contact with objects can be frequent. By incidental contact, we mean unintentional contact that occurs while performing a goal-directed manipulation task, as opposed to contact from active and deliberate haptic probing. Our system uses a data-driven method from [3] to rapidly categorize incidental contact into categories relevant to reaching.

Our method uses hidden Markov models (HMMs) to model the time-series contact force data from the fabric-based tactile sensor [4] and uses the models to classify the objects in the environment into the categories of trunk and leaves. Researchers have used HMMs, in particular, for various online categorization tasks such as handwriting recognition [28], human actions [29], and sign language recognition [30]. Chu et al. have used HMMs for offline haptic categorization tasks [31] using data from specific exploratory behaviors. See [32] for a more thorough review of the large body of work related to haptic classification of objects using data-driven techniques.

For our system, we trained two HMM models (for trunk objects and leaf objects) using training data we collected using the robot Cody [3], on environments wholly composed of small tree trunks and artificial leaves as shown in Fig. 5. We used a previous controller from [27] for training in these cluttered environments. Notably, even though Cody used a different controller and different tactile sensors, we found that the same HMM models worked well in practice for DARCI. DARCI and the environment are shown in Fig. 1. Our rapid categorization method classifies, online and in real-time, the contact force data for every taxel on the tactile sleeve.

We create a haptic map by mapping the leaf and trunk contacts encountered, as described in Sec. III-D.1 for the planner. The visualization only shows the trunk contacts, in brown (Fig. 6).

D. Planning with Contact

In this section, we describe a global search-based planner with a haptic-cost map constructed by the haptic classifier described in Sec. III-C.

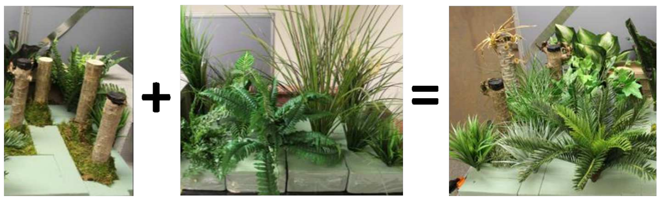

Fig. 5: (Left) Trunk-only environment for training the HMM model for Trunk category; (Middle) Leaf-only environment for training the HMM model for Leaf category; (Right) Combined environment for testing.

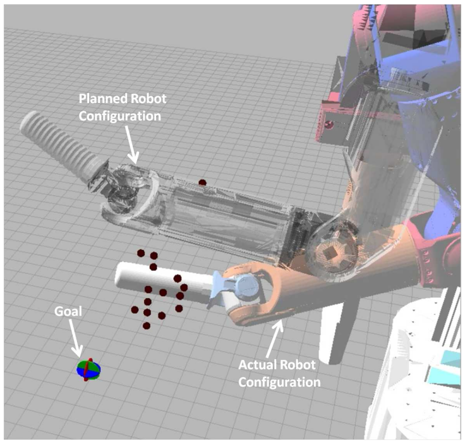

Fig. 6: Planned robot configuration with a visualization of trunk contacts in the associated haptic map.

1) The Haptic Map: We first construct a 3D cost map (haptic map). We represent the workspace of the robot as a 3D voxel grid with 0.01 $m \times$ 0.01 $m \times$ 0.01 m voxel size in Cartesian space. Each voxel includes a collision cost associated with the location. We define the collision-cost value as a scalar value between 0 to 100. Higher values indicate greater difficulty for traversal of the location by the robot's arm. The haptic classifier from Sec. III-C provides the 3D location and category of each detected contact while the robot moves. The system uses this information to continuously update its haptic map. It assigns collision costs of 50 and 100 for contacts classified as leaves or trunk, respectively. Open space has a collision cost of 0. Newly detected leaves and trunk contacts overwrite the current voxel values. For this implementation, voxels are never set back to 0. Implementations that allow the arm's volume to reduce voxel costs or that decay voxel costs over time might be valuable for dealing with dynamic environments and noisy sensing.

The total volume of the haptic map is a rectangular box, 0.6 $m \times$ 0.7 $m \times$ 0.6 m in front of the robot. The system initially populates this volume with zeros, using the optimistic initial guess that the entire unobserved environment is easy to traverse. The map records the contact information using the Point Cloud Library's (PCL) Voxel Grid [33].

2) The Haptic Planner: For our haptic planner, we use the global, sampling-based planner RRT-Connect [34], as implemented in OMPL [35]. RRT-Connect attempts to find an initial trunk-collision free trajectory using a binary map generated by ignoring any voxel in the haptic map with a cost $<$100. Using a cost-based planner instead, might have benefits. If RRT-Connect returns a trajectory, the haptic planner uses it to produce a simplified and interpolated trajectory. It then computes a cost for this candidate trajectory by summing up the costs of all contacts that occur with the vertices of the arm's collision mesh as it moves through the haptic map. If this cost is $>$1000, equivalent to 10 rigid (trunk) or 20 soft (leaf) contacts, then the candidate trajectory is rejected and the planner tries again with a different final configuration. This threshold allows the robot to try to follow a trajectory even if it the haptic map indicates that it will result in some contact.

The haptic planner selects a final configuration for the arm from a list of valid arm configurations, and then plans a trajectory from the arm's current configuration to this final configuration. Valid final arm configurations are joint angles that place the end-effector at the 3D target region with no trunk-collision in the current haptic map. To create the list of configurations, the system samples 20 random poses around the target location using uniform random quaternions described in [36]. Then, using full 6-DoF IK from OpenRave [37], the system selects a configuration with the minimum angular distance from current joint angles. Fig. 6 shows an example of a joint-space goal produced by the haptic planner for use by the joint-space version of *DynamicMPC*.

E. Implementation

1) The Tactile Sensor: For tactile sensing, we use our fabric-based tactile-sensing sleeve from [4]. The sleeve covers the forearm and end effector of the robot's left arm with 24 tactile pixels (taxels). The system converts the raw taxel measurements to approximate normal forces using a non-linear calibration function.

2) The Robot: We used the humanoid robot DARCI, a Meka M1 Mobile Manipulator, which includes a mobile base, a torso on a vertical linear actuator, and two 7-DoF arms. The mobile base and torso height remained fixed throughout our evaluation. The left arm had a 3D-printed cylindrical ABS plastic end effector (visible on our model in Fig. 6). The arms of the robot use series elastic actuators (SEAs) at the joints and have a real-time impedance controller that simulates low-stiffness visco-elastic springs with additive gravity-compensating torques at the robot's joints. Work by other researchers, such as [38], [39], has demonstrated that joints with low stiffness can lower forces due to unexpected contact and that the passive mechanical compliance of SEAs can be advantageous in the presence of shock loads. Within rigid clutter, low-stiffness joints can also mitigate jamming and wedging [40], [41].

IV. Evaluation and Results

We evaluated the system in the trunk-and-leaf environment from Sec. III-C by commanding the robot to reach to seven target locations throughout the environment (Fig. 7). We defined target locations by placing the robot's end effector at the target to ensure that it could be reached. The system attempted to reach each of the 7 target locations 3 times, for a total of 21 attempts. After each attempt, we reconfigured

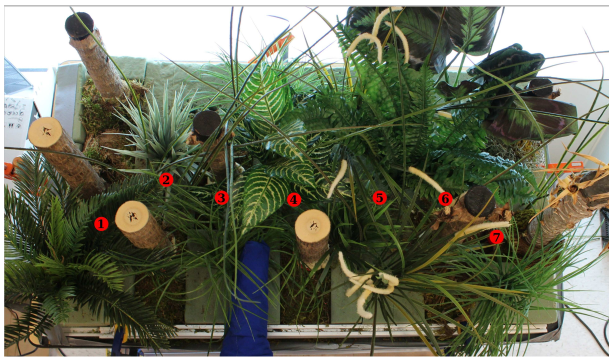

Fig. 7: Trunk-and-leaf test environment. Seven target locations identified by red dots (#1-#7 in order left to right).

displaced foliage in order to keep the environment substantially the same for each attempt. During testing, the robot reached each target successfully at least once.

In total, the robot successfully reached the target location in 16/21 (76.19%) of the attempts (5/21 failures). It reached 6 of the 7 distinct locations in less than 20 seconds on at least one attempt. The fastest successful attempt for target #5 required 70 seconds.

For the successful attempts, the robot took an average total time of $39.74 \pm 46.00s$ $(mean \pm std)$. It succeeded in 9/21 (42.86%) attempts on its first reach using the *DynamicMPC* controller from *LIC1*, which took $10.97 \pm 3.87s$ $(mean \pm std)$. In 1/21 (4.76%) attempts it failed when trying to pull back after its first reach failed, and in 2/21 (9.52%) attempts it failed when trying to reach the *LIC2* initial configuration. In 1/21 (4.76%) attempts it succeeded on the second reach starting from *LIC2*, which took $21.30s$. In 5/21 (23.81%) attempts it succeeded after using the first planned path based on the haptic map, which took $67.59 \pm 26.77s$ $(mean \pm std)$. In 1/21 (4.76%) attempts it succeeded after using a greedy reach starting from the failure point of a third planned trajectory, which took $177.93s$. In 2/21 (9.52%) attempts it failed when the planner did not return a plan after the *LIC2* reach, or after pulling back to the *LIC2* setup configuration.

Interestingly, the robot only reached target #5 twice, which required the longest and 3rd longest reaching times out of all successful attempts. The robot only reached target #6 once. The robot had difficulty reaching targets #5 and #6 even though they were near the robot and mostly obstructed by ostensibly movable foliage, rather than rigid trunks.

V. Discussion

When the robot only used *LIC* and *DynamicMPC* for a successful reach, it succeeded in 12.00 ± 4.90 seconds $(mean \pm std)$. When the robot also used planning, it succeeded in 85.98 ± 46.91 seconds $(mean \pm std)$. As illustrated by Fig. 8, when our system used planning, it took longer for it to be successful. This is in part due to the high efficiency of learned initial conditions and greedy reaching versus geometric planning and task-space control, but it also relates to the complexity of the particular reaching task. The success of *LIC1* with *DynamicMPC* in quickly reaching a variety of target locations in dense clutter emphasizes the

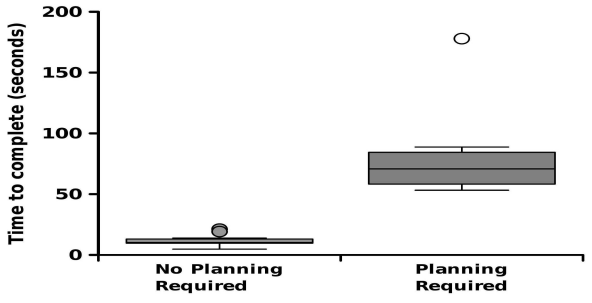

Fig. 8: Completion time in cases where planning is or is not required. Whiskers at 1.5·Interquartile Range (IQR), filled outliers $> 1.5 \cdot IQR$, open outliers $> 3 \cdot IQR$.

capability of these modules for operating with clutter without using a map of the environment. The tactile sensor provides data of limited size and scope that is immediately relevant to reaching a target and maintaining low contact forces. Also, because data is collected while reaching, rather than performing sensing and mapping in advance, our system avoids the delays of sensing and planning before acting, common to the 'Sense-Plan-Act' paradigm.

Our results also suggest that planning based on haptic maps can usefully complement greedy behaviors. For 6 of the 16 successful attempts, the robot used geometric planning. In 5 of the 8 attempts that used planning, the planning system succeeded using the first plan. In these cases, the end effector was typically near the target location, but stuck against some obstacle, and small alterations from the greedy approach freed the arm to reach the target. In one case, the end-effector became stuck against foliage intertwined between two plants. Notably, the maps collected and used by the planner were sparse compared to maps generated by modern 3D range sensors and included information about areas not visible to traditional line-of-sight sensors. This sparsity could enable faster planning with the trade-off that additional re-plans may be required as the robot makes contact and maps previously undiscovered obstacles.

Sequentially trying different methods enables the system to reach targets quickly when mechanically clear paths are available, while still finding less direct paths to targets which are harder to reach. Fig. 9 shows the cumulative success percentage as the system progresses through the defined sequence of actions. Our integrated system takes advantage of the complementary capabilities of the system components. We were inspired, in part, by the notion that people first attempt some tasks with immediacy and little preparation, trying to achieve rapid success. In doing so, people can gain information about the task that would be difficult to infer from passive observation. If these initial attempts fail, a person can then try a slower and more deliberative approach that may involve further exploration of the situation.

Our system is fallible, as evidenced by the robot's 5 failed attempts out of its 21 total attempts. In one case, the system became stuck when attempting to extract its arm after the first greedy reach. The current extraction behavior uses the task-space controller to move the robot's end effector along the reverse of the path it took into the clutter. We found this

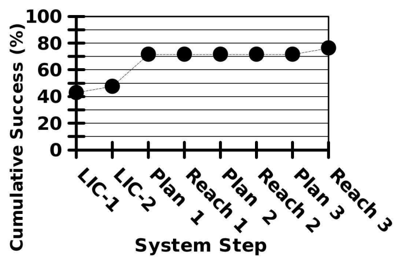

Fig. 9: Cumulative success percentage as the system progresses through the process in order. *LIC1* and *2* give almost 50% success. A single planned path brings it to >70%.

extraction method outperformed an extraction behavior we implemented that attempted use the joint-space controller to move the robot's entire arm along the reverse of the joint-space configuration it took into the clutter. This joint-space extraction method often became stuck from tracking error and its inability to navigate around contacts using the arm's redundant DoF (Sec. III-A).

The robot failed twice due to being unable to reach the *LIC2*-selected arm configuration. The robot attempts to use the joint-space version of *DynamicMPC* to follow a linear trajectory in joint-space from the arm's configuration after extraction to the *LIC2*-selected arm configuration. For one failure, while reaching to target #6, the robot's arm became stuck against the robot's torso. In the other failure, the robot's arm became stuck due to its end effector making contact with the environment. The current system does not attempt to become unstuck in these situations, which could potentially improve the system's overall performance.

Two other failures resulted from the planner not returning a plan before the 3 minute timeout. In these cases, the maps were relatively dense. We did not optimize the planning algorithm for speed, and it is possible that other methods, such as trajectory-optimization based planning, could provide valid paths more quickly and consistently. Likewise, our current mapping system does not clear out occupied voxels, even if they've subsequently been traversed with little effort.

VI. Conclusion

We have presented an integrated robotic system capable of haptically reaching locations in dense clutter using model predictive control (MPC), learning, haptic mapping, and planning. MPC enables the robot to rapidly reach into the unknown while keeping contact forces low. Learning enables the robot to reach from good initial arm configurations and categorize incidental contact based on whole-arm tactile sensing. Haptic mapping enables the robot to remember parts of the environment relevant to the task of reaching (e.g., impassable trunks) and ignore irrelevant clutter (e.g., movable leaves). Finally, planning enables the robot to take advantage of its haptic map and solve reaching problems with which greedy reaching has difficulty.

In our evaluation, the system successfully reached all 7 target locations in at least 1 of its 3 attempts. For some attempts, it succeeded using fast and efficient greedy reaching from learned initial arm configurations. For other attempts, it only succeeded after using geometric planning with a sparse map of locations at which it had detected impassable obstacles using tactile sensing. These approaches complement one another. The more efficient greedy reaching system can often find a solution quickly. However, if it does not, the less efficient planning system has an opportunity to solve the problem and benefits from the haptic map generated while the robot greedily reached to the target.

Acknowledgments*: This work was primarily supported by DARPA M3 Contract W911NF-11-1-603 with additional support from NSF Award IIS-1150157, an NSF GRFP, and NIDRR RERC Grant H133E130037.*

REFERENCES

[1] M. D. Killpack and C. C. Kemp, “Fast reaching in clutter while regulating forces using model predictive control,” in *Humanoid Robots, IEEE-RAS International Conference on*, Atlanta, GA, USA, Oct. 2013.

[2] D. Park, A. Kapusta, Y. K. Kim, J. M. Rehg, and C. C. Kemp, “Learning to reach into the unknown: Selecting initial conditions when reaching in clutter,” in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*, 2014.

[3] T. Bhattacharjee, A. Kapusta, J. M. Rehg, and C. C. Kemp, “Rapid categorization of object properties from incidental contact with a tactile sensing robot arm,” in *Humanoid Robots, IEEE-RAS International Conference on*, Atlanta, GA, USA, Oct. 2013.

[4] T. Bhattacharjee, A. Jain, S. Vaish, M. D. Killpack, and C. C. Kemp, “Tactile sensing over articulated joints with stretchable sensors,” in *World Haptics Conference (WHC), 2013*. IEEE, 2013, pp. 103–108.

[5] M. T. Mason, S. S. Srinivasa, and A. S. Vzquez, “Generality and simple hands.” in *ISRR*, 2009, pp. 345–361.

[6] Merriam-Webster.com. (2014) ”clutter”. [Online]. Available: http://www.merriam-webster.com/dictionary/clutter

[7] M. Gupta, T. Ruhr, M. Beetz, and G. S. Sukhatme, “Interactive environment exploration in clutter,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5265–5272.

[8] M. R. Dogar, M. C. Koval, A. Tallavajhula, and S. S. Srinivasa, “Object search by manipulation,” *Autonomous Robots*, vol. 36, no. 1-2, pp. 153–167, 2014.

[9] S. Panda, A. Hafez, and C. Jawahar, “Learning support order for manipulation in clutter,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 809–815.

[10] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz, “Perceiving, learning, and exploiting object affordances for autonomous pile manipulation,” 2013.

[11] P. Grice, M. Killpack, A. Jain, S. Vaish, J. Hawke, and C. Kemp, “Whole-arm tactile sensing for beneficial and acceptable contact during robotic assistance,” in *ICORR, 2013 IEEE*, Seattle, WA, USA, Jun. 2013.

[12] L. E. Kavraki and S. M. laValle, *Chapter 5: Motion Planning, Handbook of Robotics, Siciliano, Bruno; Khatib, Oussama (Eds.)*. Springer, 2008.

[13] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour, “Manipulation planning among movable obstacles,” in *Proc. IEEE Int. Conf. on Robotics and Automation*. Citeseer, 2007, pp. 3327–3332.

[14] A. Leeper, K. Hsiao, M. Ciocarlie, I. Sucan, and K. Salisbury, “Arm teleoperation in clutter using virtual constraints from real sensor data,” in *RSS Workshop on Robots in Clutter: Preparing Robots for the Real World*, 2013.

[15] ——, “Methods for collision-free arm teleoperation in clutter using constraints from 3d sensor data,” in *IEEE Intl. Conf. on Humanoid Robots*, Atlanta, GA, 10/2013 2013.

[16] S. Srinivasa, C. Ferguson, D. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe, “Herb: A Home Exploring Robotic Butler,” *Autonomous Robots*, 2009.

[17] A. Saxena, J. Driemeyer, and A. Ng, “Robotic Grasping of Novel Objects using Vision,” *The International Journal of Robotics Research*, vol. 27, no. 2, p. 157, 2008.

[18] A. Saxena, L. Wong, M. Quigley, and A. Y. Ng, “A vision-based system for grasping novel objects in cluttered environments,” in *Robotics Research*. Springer, 2011, pp. 337–348.

[19] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, “Navigation in three-dimensional cluttered environments for mobile manipulation,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 423–429.

[20] J.-C. Latombe, *Robot motion planning*. Springer Verlag, 1990.

[21] Z. Guo and T. Hsia, “Joint trajectory generation for redundant robots in an environment with obstacles,” *Journal of robotic systems*, vol. 10, no. 2, pp. 199–215, 1993.

[22] M. Stilman, “Global manipulation planning in robot joint space with task constraints,” *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 576–584, 2010.

[23] 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.

[24] A. De Luca and L. Ferrajoli, “Exploiting robot redundancy in collision detection and reaction,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3299–3305.

[25] M. D. Killpack, “Model predictive control with haptic feedback for robot manipulation in cluttered scenarios,” Ph.D. dissertation, 2013.

[26] J. Mattingley and S. Boyd, “Cvxgen: a code generator for embedded convex optimization,” *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11081-011-9176-9

[27] A. Jain, M. D. Killpack, A. Edsinger, and C. C. Kemp, “Reaching in clutter with whole-arm tactile sensing,” *International Journal of Robotics Research (IJRR)*, vol. 32, no. 4, pp. 458–482, April 2013.

[28] U. Garain and B. Chaudhuri, “Recognition of online handwritten mathematical expressions,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 6, pp. 2366–2376, 2004.

[29] J. Yamato, J. Ohya, and K. Ishii, “Recognizing human action in time-sequential images using hidden markov model,” in *in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1992, pp. 379–385.

[30] T. Starner, J. Weaver, and A. Pentland, “Real-time american sign language recognition using desk and wearable computer based video,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371–1375, 1998.

[31] V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. Perez-Tejada, M. Arrigo, N. Fitter, J. C. Nappo, T. Darrell, and K. J. Kuchenbecker, “Using robotic exploratory procedures to learn the meaning of haptic adjectives,” in *Proceedings of International Conference on Robotics and Automation*, 2013.

[32] T. Bhattacharjee, J. M. Rehg, and C. C. Kemp, “Haptic classification and recognition of objects using a tactile sensing forearm,” in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, October 2012, pp. 4090 – 4097.

[33] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[34] J. Kuffner and S. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in *Robotics and Automation, 2000. Proceedings. ICRA ’00. IEEE International Conference on*, vol. 2, 2000, pp. 995–1001 vol.2.

[35] I. A. Şucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[36] K. Shoemake, “Uniform random rotations,” in *Graphics Gems III*, D. Kirk, Ed. Academic Press, 1992, pp. 124–132.

[37] R. Diankov and J. Kuffner, “Openrave: A planning architecture for autonomous robotics,” Tech. Rep., 2008.

[38] G. Pratt and M. Williamson, “Series elastic actuators,” in *IROS*, 1995.

[39] S. Buerger and N. Hogan, “Complementary stability and loop shaping for improved human–robot interaction,” *Robotics, IEEE Transactions on*, vol. 23, no. 2, pp. 232–244, 2007.

[40] P. Dupont and S. Yamajako, “Jamming and wedging in constrained rigid-body dynamics,” in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. IEEE, 1994, pp. 2349–2354.

[41] M. Mason, *Mechanics of robotic manipulation*. MIT Press, 2001.