Proceedings of Australasian Conference on Robotics and Automation, 2-4 Dec 2014, The University of Melbourne, Melbourne, Australia

Fast Path Planning for Precision Weeding

James Ju Heon Lee[1], Kris Frey[1,2], Robert Fitch[1] and Salah Sukkarieh[1]
[1]Australian Centre for Field Robotics
The University of Sydney, NSW, Australia
jlee3701@uni.sydney.edu.au, {rfitch,salah}@acfr.usyd.edu.au
[2]Massachusetts Institute of Technology (MIT)
Boston, MA, USA
kfrey@mit.edu

Abstract

Agricultural robots have the potential to reduce herbicide use in agriculture and horticulture through autonomous precision weeding. One of the main challenges is how to efficiently plan paths for a robot arm such that many individual weeds can be processed quickly. This paper considers an abstract weeding task among obstacles and proposes an efficient online path planning algorithm for an industrial manipulator mounted to a mobile robot chassis. The algorithm is based on a multi-query approach, inspired by industrial bin-picking, where a database of high-quality paths is computed offline and paths are then selected and adapted online. We present a preliminary implementation using a 6-DOF arm and report results from simulation experiments designed to evaluate system performance with varying database and obstacle sizes. We also validate the approach using a Universal Robots UR5 manipulator and ROS interface.

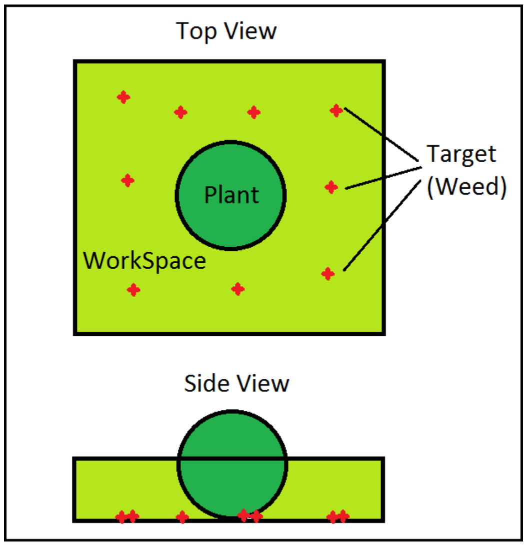

Figure 1: Graphical depiction of an abstract precision weeding problem. Point-weeds lie on a ground plane. A convex obstacle represents crop plants to be avoided. Top view and side view are shown.

1 Introduction

Agricultural robots operate in various farm environments and perform useful tasks in collecting information and automating mechanical operations. We are interested in using robot manipulators to perform tasks such as weeding and harvesting in outdoor environments for agriculture and horticulture. For example, a robot arm could perform mechanical weed control or else could precisely apply herbicides to reduce the need for broadcast spraying. Robots could also be used for selective harvesting which otherwise would be cost-prohibitive to perform manually. In these types of tasks, one of the basic underlying research problems is how to efficiently plan manipulator trajectories such that system performance is eventually feasible for industrial applications. In this paper we consider fast planning methods for an abstract weeding task in a horticulture application.

In Australia, agricultural robotics is motivated by several factors. One factor is the decreasing availability of human labour: the number of people working in agriculture in Australia has declined by 40% since 1981 [Australian Bureau of Statistics, 2012]. At the same, there is strong interest in increasing production in an environmentally sustainable manner [DAFF, 2013]. In previous work, we have investigated the use of teams of small autonomous robots to perform tasks such as spot-spraying [Ball *et al.*, 2013]. This type of system reduces soil compaction by replacing large tractors with small robots, and reduces the volume of herbicide applied in comparison to broadcast spraying. Robot manipulators have the potential to reduce chemical use even further in more general circumstances by precisely targeting individual weed types detected by a perception system, or to eliminate herbicides for problem weeds through mechanical weed control.

The planning challenge in robotic weed control is how to exploit the structure of the problem to produce fast planners. From a motion planning perspective, manipulator planning for weed control can be viewed as a relatively simple case because the environment is moderately structured, as shown in Fig. 1. Weeds lie on the ground which is uneven but still relatively flat. Workspace obstacles arise from the geometry of the mobile robot chassis that carries the manipulator, which is known, plus the geometry of the crop which could be modelled using convex volumes. The planning problem in its simplest form in this context is to position the end effector above weeds in sequence. Industrial manipulators typically are packaged with efficient controllers that allow direct positioning of the end effector, but since these controllers do not generally consider external obstacles they cannot be applied directly. Powerful sampling-based planners that can solve the general motion planning problem can be applied, but this added theoretical power comes at the expense of planning time.

It is important to consider fast planning methods in order to support the feasibility of industrial-scale applications. Fast planning is also important from an algorithmic perspective in order to support higher-level optimisation of weed sequences that would require some notion of 'traversal cost' between weed pairs. Recently, similar problems have been studied in the context of industrial bin-picking [Ellekilde and Petersen, 2013] and hexapod stabilisation [Hörger *et al.*, 2014] where a set of paths is pre-computed and adapted online. This approach is similar in spirit to well-known multi-query roadmap methods [Choset *et al.*, 2005].

In this paper, we propose a multi-query approach in the context of precision weeding. We construct a database of paths offline using a sampling-based planner, and adapt these paths online using convex optimisation to match the weed locations and crop obstacle geometry at hand. We implemented our algorithms and present simulation experiments that examine system performance for varying database and obstacle sizes. Further, we performed experiments using an industrial robot manipulator and report results from two trials with ten weed locations each.

Results from the robot experiments indicate that our method is capable of efficient planning with fast run time (under 0.6 s) and a reasonable success rate (80%). These are preliminary results that encourage further refinement of the method. This paper is one of the first to explore the problem of manipulator planning for precision weeding, and is the first to propose a multi-query approach for this task.

2 Related Work

Our work is inspired by exciting recent results in fast motion planning for industrial bin-picking [Ellekilde and Petersen, 2013]. Here, a path database is created where path endpoints lie in defined pick and place volumes in configuration space. Our work differs in that we focus on the precision weeding task, experimentally evaluate various database and obstacle sizes, and use a more general convex optimisation procedure that does not involve tuning parameters to determine which path segments to optimise.

Related work in weed control robots primarily focuses on exotic end-effectors that kill weeds through electrocution [Blasco *et al.*, 2002], rotating blades [van Evert *et al.*, 2011], and direct chemical application combined with a cutter [Jeon and Tian, 2009]. This work ignores obstacles and relies on simple steering methods for end-effector positioning. Obstacle avoidance has been studied in simulation using a 2DOF arm by choosing via points and fitting a cubic polynomial to find a trajectory [Sengupta *et al.*, 2011]. A survey of work in robotics for weed control as of 2008 is presented in [Slaughter *et al.*, 2008].

Other related work includes manipulator planning for harvesting tasks. Interesting early work explored heuristic search (A*) for cucumber harvesting [van Henten *et al.*, 2002], and fast planning where robot links are assumed to be decoupled [Van Willigenburg *et al.*, 2004]. Recently, general sampling-based planners have been studied for apple harvesting with planning times shown to be less than 5 s [Nguyen *et al.*, 2013]. Trajectory planning ignoring obstacles has been explored for kiwi fruit harvesting [Scarfe *et al.*, 2009] and general fruit picking [Baur *et al.*, 2014]. Our work differs in that it focuses on how to exploit the power of general sampling-based planners in a time-efficient manner.

Motion planning in general has a rich history in robotics [Choset *et al.*, 2005]. Within the mobile manipulation community, there has been considerable recent interest in gradient optimisation techniques that adapt an initial guess [Ratliff *et al.*, 2009]. Our work currently uses simple gradient optimisation for adapting paths, but can benefit from more advanced gradient optimisation methods in future work.

3 Problem Setup

Our objective is to quickly plan trajectories for a robot arm with a virtual end effector that must 'touch' weed locations. This abstract end-effector could represent a dripper mechanism, tine, or rotary tool used to effect chemical or mechanical weed control.

We assume that the robot arm is suspended (upside down) beneath a mobile chassis. The workspace of the robot is bounded by the internal dimensions of the mobile chassis and the ground. The ground surface is assumed to be planar. We consider the case where there is a single plant (such as capsicum or beetroot) that acts as an obstacle. The plant is represented by a sphere of known radius resting on the ground plane centred beneath the robot base. Weeds are located randomly on the ground plane within the robot's workspace and are represented as point-weeds with known locations provided *a priori* by a perception system. A trajectory is represented by a sequence of waypoints in the configuration space of the arm, along with time durations between waypoints. A controller is available that allows the arm to follow a given trajectory.

The basic problem statement is as follows: given a weed location and initial arm configuration, efficiently find a path that places the robot's end effector immediately above the weed location. This basic problem can be trivially extended to the case where the robot must touch a sequence of weeds in turn, given such a sequence. Considering sets of weeds in any given workspace motivates a multi-query approach to the problem. We do not consider the optimisation problem of choosing an efficient traversal sequence over an initially *unordered* set of weeds, but we note that the basic problem could be considered as a sub-problem of this more complex case; a solution to the basic problem could provide inter-weed costs in a Travelling Salesman Problem (TSP) formulation, for example.

4 Path Planning Algorithm

We propose a multi-query approach where a database of diverse paths is computed and stored offline, and an online trajectory optimiser selects and adapts paths from this database to solve the problem instance at hand. This algorithm architecture is shown in Fig. 2.

The offline component generates a set of paths intended to span the space of possible weed locations, or *targets*. We choose a set of targets, compute the corresponding robot configurations using an IK solver, and compute paths between all pairs of targets using a sampling-based algorithm (the ROS implementation of RRTConnect is used in our experiments). All collision-free paths are stored. Configurations that are in collision with the robot chassis, the spherical plant obstacle, or the ground plane are discarded. Likewise, a target pair is discarded if the sampling-based algorithm fails to find a collision-free path. An index is constructed over the resulting set of paths according to their start and end configurations (joint angles). The set of targets is chosen according to a uniform grid placed on the ground plane. The grid resolution therefore determines the maximum size of the path database and is treated as an input parameter to the offline algorithm.

The online component takes as input the initial robot configuration and a new target position (not generally present in the database) on the ground plane, and computes a collision-free path to position the end effector immediately above the target. This path is computed by selecting a pre-computed path from the database and adapting it such that the start and end configurations correspond with the input. To do this, we first compute the desired end configuration using an IK solver. We then use the database index to select a number of candidate paths from the database. A set of paths is selected according to the minimum average Euclidean distance between the target start/end configurations and those of the candidate database path. We then iterate through the candidates and attempt to adapt (warp) each path using a gradient descent algorithm. We return the first successful attempt, or else return failure if no candidates can be adapted.

The gradient descent method attempts to warp the candidate path to minimise its time duration while fixing the start and end configurations to the input. The trajectory duration depends on each of the robot's joints, and hence the gradient is defined as $\frac{dt}{d\theta_i}$, where t is time-length of the whole trajectory, θ is joint angle, and i is joint number. The algorithm first moves the start and end configurations towards the targets so long as the path remains collision-free, and then iteratively moves each other waypoint along the path according to the gradient. Pseudocode is listed for this process in Algorithm 1 and for the gradient computation in Algorithm 2.

5 Experiments and Results

In this section, we evaluate our approach both in simulation and using a real robot manipulator. First, we present the implementation of our method. Then, we examine simulated performance with varying database size and spherical plant radius. Finally, we present experimental results with deterministic and random weed locations.

5.1 Implementation

We implemented our algorithms for a robotic system consisting of a Universal Robots UR5 6-DOF manipulator [Universal Robots, 2014] mounted beneath a mobile robot chassis. The UR5 is mounted upside down with its base 1.0 m from the ground plane. The workspace is bounded by the internal dimensions of the mobile robot chassis which are 1.0 m in the vertical dimension and 0.8 m in the lateral dimensions. The UR5 system includes a control computer that accepts trajectory commands for execution. Our algorithms were implemented using a standard laptop computer (1.8GHz CPU, 8 GB RAM) connected to the UR5 controller via a standard communication network.

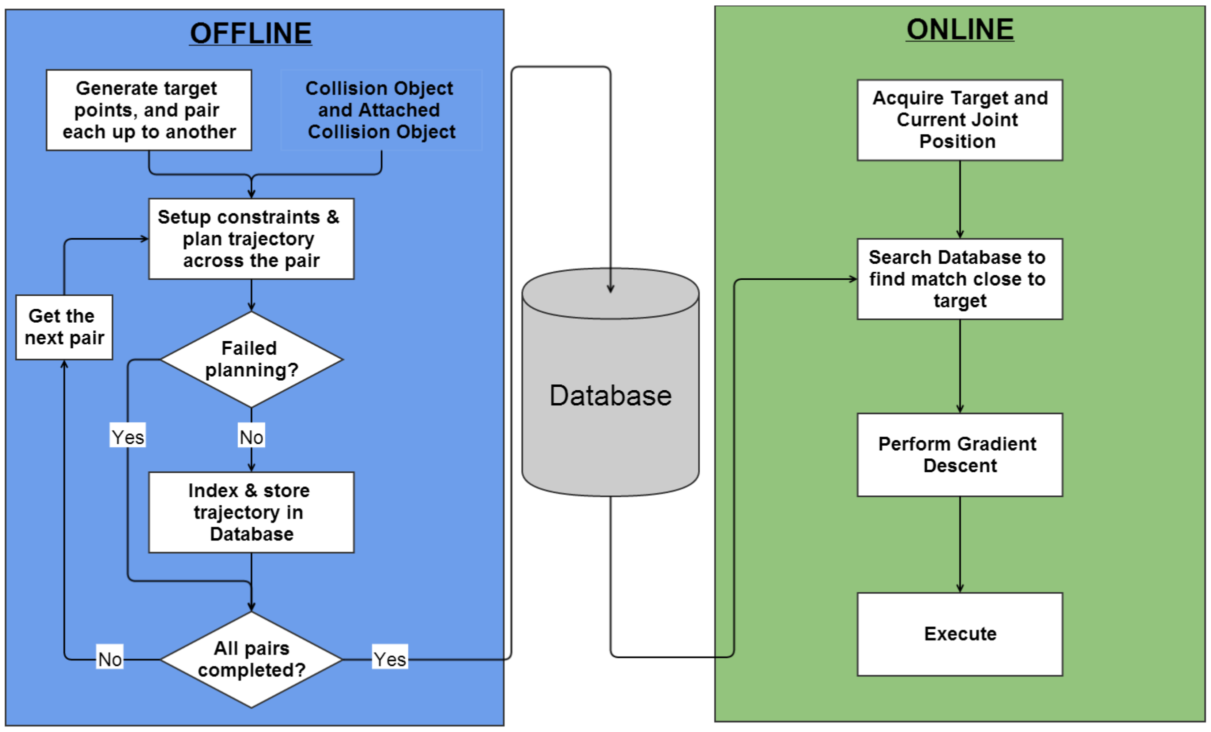
OFFLINE
Generate target points, and pair each up to another
Collision Object and Attached Collision Object
Setup constraints & plan trajectory across the pair
Get the next pair
Failed planning?
Yes
No
Index & store trajectory in Database
All pairs completed?
No
Yes
Database
ONLINE
Acquire Target and Current Joint Position
Search Database to find match close to target
Perform Gradient Descent
Execute

Figure 2: Algorithm overview.

Algorithm 1 Warp path to match target endpoints

```
Input: Initial path $pathDB$, target endpoints $pStart$ and $pEnd$
while path endpoints do not match targets do
    ▷ Move endpoints
    $stepSize = 1$
    repeat
        $pathTemp \leftarrow pathDB$
        move start towards $pStart$ by factor of $stepSize$
        move end towards $pEnd$ by factor of $stepSize$
        $stepSize = stepSize/2$
    until $pathTemp$ is collision-free or failure

    ▷ Smooth other waypoints
    $t_{old}$ = duration of $pathTemp$
    repeat
        Compute gradients for $pathTemp$ (Alg. 2)
        for all waypoints do
            $stepSize = 1$
            repeat
                move joints by $stepSize(-gradient)$
                $stepSize = stepSize/2$
            until $pathTemp$ is collision free
        end for
        $t_{new}$=duration of $pathTemp$
    until $t_{old} - t_{new} < \epsilon$
end while
```

Algorithm 2 Compute gradients

```
Input: Sequence of waypoints
$t_{old}$ = initial trajectory duration
for all waypoints do
    for all joints do
        Perturb joint angle by a small value $\delta$
        $t_{new}$ = resulting trajectory duration
        $gradient = (t_{new} - t_{old})/\delta$
        Revert joint angle to initial value
    end for
end for
return array of gradients
```

The software components of the system are built using the Moveit! package within ROS [I.A. Sucan and S. Chitta, 2014] and the associated UR5 driver. Simulation experiments use the Gazebo simulation environment [Koenig and Howard, 2004].

We implemented the offline and online components of our algorithms as two separate software modules. The database module builds the set of paths (using ROS RRTConnect) and stores them in flat-file format. The planner module reads the database file on startup and builds the index for use during planning.

We built databases for target grids ranging in resolution from 3×3 to 7×7. Time to construct the databases ranged from 30 minutes to 8 hours depending on size. All

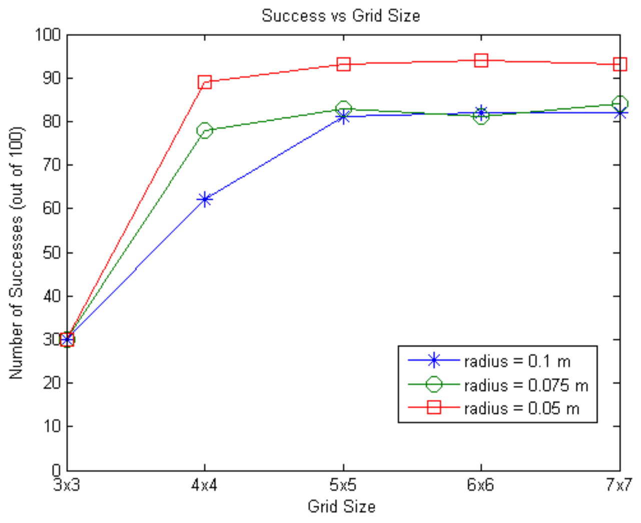

Figure 3: Number of successful trials for simulation experiments with varying database and obstacle sizes.

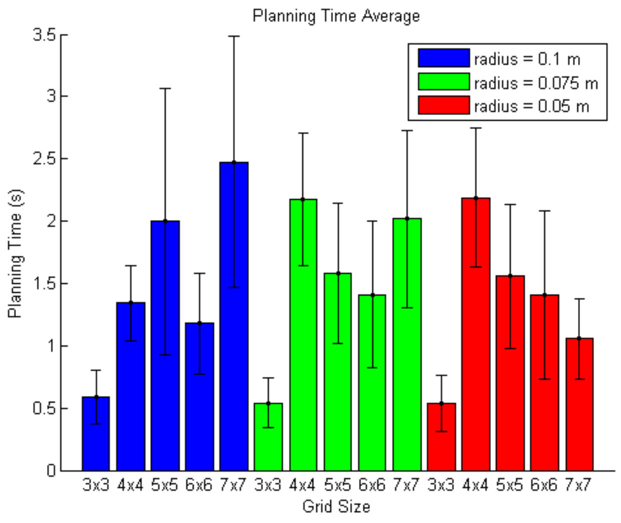

Figure 4: Planning times (search plus adaptation) in simulation for successful trials only. Error bars indicate standard error of the mean (SEM).

databases were constructed with a spherical plant (obstacle) of radius 0.1 m. In our experiments, we varied the plant radius from 0.05 m to 0.1 m. Weed target positions in the experiments were either chosen randomly or pre-computed.

The planner module plans and executes trajectories for a sequence of weed targets. The planner considers the three closest database paths. The first path to be successfully adapted is immediately executed. We instrumented the implementation to record planning time (including database search) and trajectory duration. We limited all joint velocities to 50% of maximum for safety during execution.

5.2 Experiments in Simulation

We performed simulation experiments in order to evaluate the performance of our method with varying database size and obstacle radius. The main purpose of these experiments is to determine an appropriate database size for practical applications. The experiments investigate the trade-off between smaller databases and increased planning time.

We also varied the size of the spherical plant obstacle. In a practical application, crop volume will vary and our intent is to validate the performance of the online component of the system with obstacle sizes smaller than that used during offline database creation.

We performed 100 trials for each experimental condition. A trial consists of planning and executing a path to a randomly chosen feasible weed target position within the planar workspace. The target position specifies the desired position of the UR5's tool centre point and is chosen to be 0.15 m above the ground plane in order to account for the presence of a virtual end effector. The experimental conditions consist of all pairs of database sizes $\{3\times3,\ 4\times4,\ 5\times5,\ 6\times6,\ 7\times7\}$ and spherical plant radii $\{0.05\text{ m},\ 0.075\text{ m},\ 0.1\text{ m}\}$. There are a total of 15 experimental conditions.

Figure 3 shows the number of successes for each of the experimental conditions. A failure occurs when none of the three possible plans selected from the database could be successfully adapted. The general trend is that successes increase with database size. This is expected since larger databases would have plans that are more likely to be closer to the target positions. Also, smaller plant radii lead to more successes due to lower chance of collision. We note that the effect of larger databases appears to plateau at size 5×5.

Figure 4 shows mean planning times for successful trials only. The very low values for the smallest database conditions are related to the corresponding low success rate. Few plans succeeded, but did so quickly. Otherwise we observe an interaction effect between database size and obstacle size. Planning time increases with database size for large obstacles but decreases for small obstacles. This is most likely to do with poor plan selection. With large obstacles and a large database, selected plans are close to limits and more computation is required for adaptation. With smaller obstacles the plans are more easily adapted without collision.

Due to the large variation in planning times in general, we re-analysed the data to evaluate the portion of plans that succeeded quickly (under 0.5 s). Figure 5 shows the number of successes and Fig. 6 shows the mean planning times for this case. Here the planning times are near 0.4

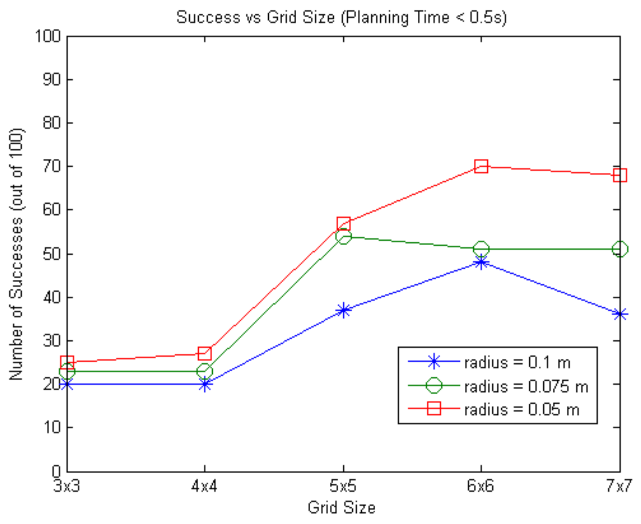

Figure 5: Number of successful trials in simulation for trials where planning time is less than 0.5 s.

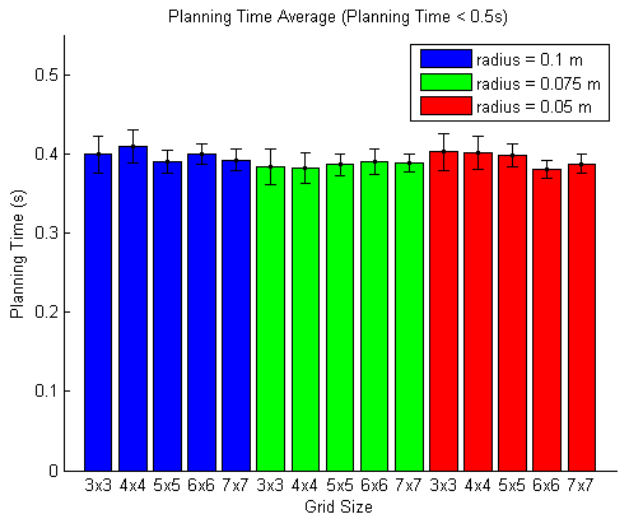

Figure 6: Planning times in simulation for successful trials with time less than 0.5 s. Error bars indicate SEM.

s, but the success rate has also decreased. Fast planning times could possibly arise because this subset of trials involve very short paths, but the execution times shown in Fig. 7 indicate a mix of long and short paths. These results are encouraging because they show that system performance is consistently fast for a reasonable subset of problem instances. Given our earlier observations on the effect of poor path selection, the focus of future work should be to improve the path selection aspect of the system.

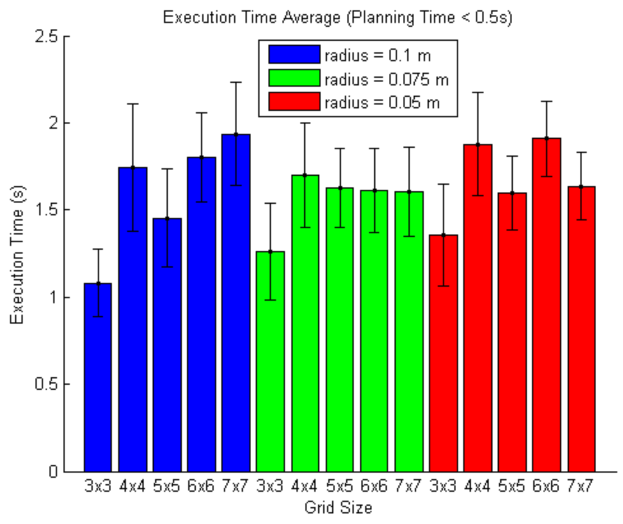

Figure 7: Plan execution time in simulation for trials where planning time is less than 0.5 s. Error bars indicate SEM.

5.3 Experiments with UR5 Manipulator

We performed two types of experiments with the hardware system. One uses a pre-computed set of weed target positions and the other uses random targets as in the simulation experiments. The purpose of these experiments is to validate our method using a real robot and to examine execution time feasibility for practical applications.

The first experiment uses 10 pre-computed weed positions chosen uniformly around a circle of radius 0.27 m placed on the ground plane and centred beneath the robot base. This experiment examines performance in the case where targets are close to each other on the ground plane, which is representative of a relatively dense weed pattern. Video snapshots of the robot executing these trajectories are shown in Fig. 8.

Quantitative results are shown in Table 1. Planning times are all between 0.3 s and 0.4 s. Execution times for trials 1, 3, and 8 are higher than the others because, although the end effector positions are close together in the workspace, the paths are longer in configuration space due to joint angle limits.

The second experiment uses 10 randomly chosen weed positions within the reachable region of the workspace. This experiment examines the case where weed positions may be sparse.

Results are shown in Table 2. Here we observe two failures with large planning time values. Otherwise the results are similar to those in Table 1.

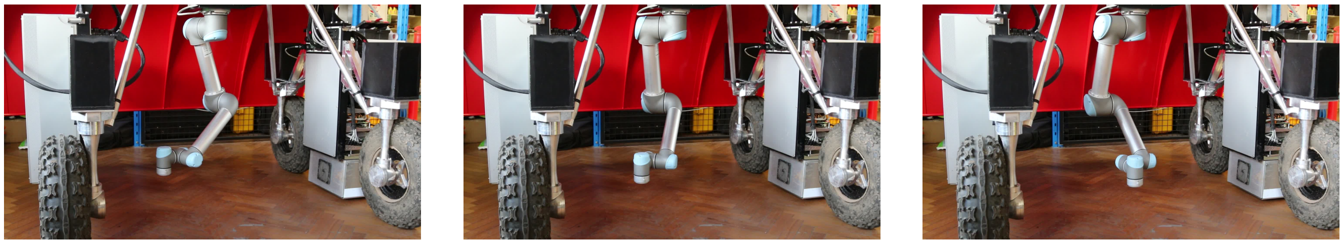
(a) Trial 6 (b) Trial 7 (c) Trial 8

Figure 8: Video snapshots from the experiment with pre-computed weed positions. The path from (a) to (b) is very short in configuration space and mainly involves rotating the joint at the robot's base. The path from (b) to (c) is much longer in configuration space due to joint limits.

Table 1: Pre-computed weed locations.

Trial number	Success	Planning time (s)	Execution time (s)
1	Yes	0.307	3.205
2	Yes	0.338	1.621
3	Yes	0.375	3.610
4	Yes	0.352	1.621
5	Yes	0.332	1.621
6	Yes	0.371	1.621
7	Yes	0.330	1.621
8	Yes	0.340	3.560
9	Yes	0.341	1.621
10	Yes	0.327	1.621

Table 2: Random weed locations.

Trial number	Success	Planning time (s)	Execution time (s)
1	Yes	0.281	3.049
2	No	3.542	–
3	Yes	0.276	0.964
4	Yes	0.300	1.531
5	No	9.461	–
6	Yes	0.352	1.546
7	Yes	0.574	3.373
8	Yes	0.380	1.653
9	Yes	0.272	2.400
10	Yes	0.277	3.004

6 Lessons Learned

As with any convex optimisation method, our path adaptation algorithm finds a local minimum. Choosing good initial conditions is therefore critical. It is clear from our experimental results that we indeed did achieve good initial conditions in a subset of trials. For many of the trials, however, this was not the case.

We believe that failures can be mainly attributed to issues with database construction and database searching. In constructing the database, we used only one IK solution (configuration) for each weed location. The IK solution computed during the online phase for an identical weed location could be different. This is due to the IK solver implementation. A similar problem could occur for weed locations that are close together; the IK solutions computed offline and online could be very far apart in configuration space. An obvious solution is to expand the database to include multiple IK solutions (where available) for each weed location pair.

We observed that in the dense databases, there was greater chance of failure since all three chosen paths were similar. If the first failed, the others also failed. This can be addressed by looking into how to achieve greater path diversity, or in evaluating more than three paths. To do this, we would have to consider 'fail fast' methods to avoid spending too much time in attempting to adapt paths. If a 100% success rate cannot be achieved in practice, it is also possible to fall back to a sampling-based planner in the worst case.

For a portion of the trials, the planning time was less than 0.6 s. We did not focus on optimising the implementation and used a standard laptop. Planning times could be improved by tuning the implementation, and also by using more capable computing hardware. For example, adapting multiple paths could easily be implemented in parallel.

7 Conclusion and Future Work

In this paper we have presented a multi-query method for efficiently planning paths for a robot manipulator in a precision weeding task. A database of pre-computed paths is constructed offline and paths are chosen and adapted online. We implemented this algorithm and provided experimental results in simulation and using an industrial manipulator. Planning times were less than 4.0 s for successful trials on average, and less than 0.6 s for a subset of trials.

Our results are encouraging and motivate several areas of future work. The algorithms should be improved, particularly with respect to database construction and searching, to improve success rate. More sophisticated path adaptation algorithms should be explored, such as [Ratliff *et al.*, 2009]. There is also good scope to improve planning times through a parallel implementation. In the longer term, we would like to consider actual end effectors as opposed to the abstract end effector assumed in this work, to perform experiments where weed locations are provided by a perception system, and to consider an extended version of the problem where the robot chassis is non-stationary. It would also be interesting to consider stochastic planning with safety constraints [Yoo *et al.*, 2013] where a non-stationary chassis is subject to control uncertainty [Peynot *et al.*, 2014] in this context.

Acknowledgements

This work was supported by the Australian Centre for Field Robotics and Horticulture Australia Limited (HAL) fund VG12104 'An Intelligent Farm Robot for the Vegetable Industry'. We would also like to thank Graeme Best for his help with software implementation.

References

[Australian Bureau of Statistics, 2012] Australian Bureau of Statistics. *Australian Social Trends.* Catalogue number 4102.0. 2012.

[Ball *et al.*, 2013] D. Ball, P. Ross, A. English, T. Patten, B. Upcroft, R. Fitch, S. Sukkarieh, G. Wyeth, and P. Corke. Robotics for sustainable broad-acre agriculture. In *Proc. of FSR*, 2013.

[Baur *et al.*, 2014] J. Baur, C. Schütz, J. Pfaff, T. Buschmann, and H. Ulbrich. Path planning for a fruit picking manipulator. In *Proc. of International Conference of Agricultural Engineering*, 2014.

[Blasco *et al.*, 2002] J. Blasco, N. Aleixos, J.M. Roger, G. Rabatel, and E. Molt. Robotic weed control using machine vision. *Biosyst. Eng.*, 83(2):149 – 157, 2002.

[Choset *et al.*, 2005] H. Choset, K.M. Lynch, S. Hutchinson, G.A. Kantor, W. Burgard, L.E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, Cambridge, MA, June 2005.

[DAFF, 2013] DAFF. *National Food Plan, Our food future.* Department of Agriculture, Fisheries and Forestry, Canberra, 2013.

[Ellekilde and Petersen, 2013] L.-P. Ellekilde and H.G. Petersen. Motion planning efficient trajectories for industrial bin-picking. *Int. J. Rob. Res.*, 32(9-10):991–1004, 2013.

[Hörger *et al.*, 2014] M. Hörger, N. Kottege, T. Bandyopadhyay, A. Elfes, and P. Moghadam. Real-time stabilisation for hexapod robots. In *Proc. of ISER*, 2014.

[I.A. Sucan and S. Chitta, 2014] I.A. Sucan and S. Chitta. Moveit! `http://moveit.ros.org`, 2014.

[Jeon and Tian, 2009] H.Y. Jeon and L.F. Tian. Direct application end effector for a precise weed control robot. *Biosyst. Eng.*, 104(4):458 – 464, 2009.

[Koenig and Howard, 2004] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proc. of IEEE/RSJ IROS*, pages 2149–2154, 2004.

[Nguyen *et al.*, 2013] T.T. Nguyen, E. Kayacan, J. De Baerdemaeker, and W. Saeys. Task and motion planning for apple-harvesting robot. In *Proc. of IFAC AGRICONTROL*, pages 247–252, 2013.

[Peynot *et al.*, 2014] T. Peynot, S.-T. Lui, R. McAllister, R. Fitch, and S. Sukkarieh. Learned stochastic mobility prediction for planning with control uncertainty on unstructured terrain. *J. Field Robot.*, 31(6):969–995, 2014.

[Ratliff *et al.*, 2009] N. Ratliff, M. Zucker, J.A. Bagnell, and S. Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *Proc. of IEEE ICRA*, pages 489–494, 2009.

[Scarfe *et al.*, 2009] A.J. Scarfe, R.C. Flemmer, H.H. Bakker, and C.L. Flemmer. Development of an autonomous kiwifruit picking robot. In *Proc. of IEEE ICARA*, pages 380–384, 2009.

[Sengupta *et al.*, 2011] A. Sengupta, T. Chakraborti, A. Konar, and A. Nagar. Energy efficient trajectory planning by a robot arm using invasive weed optimization technique. In *Proc. of Nature and Biologically Inspired Computing (NaBIC)*, pages 311–316, 2011.

[Slaughter *et al.*, 2008] D.C. Slaughter, D.K. Giles, and D. Downey. Autonomous robotic weed control systems: A review. *Comput. Electron. Agr.*, 61(1):63 – 78, 2008.

[Universal Robots, 2014] Universal Robots. `http://www.universal-robots.com/`. WebSite, 2014.

[van Evert *et al.*, 2011] F.K. van Evert, J. Samsom, G. Polder, M. Vijn, H.-J. van Dooren, A. Lamaker, G.W.A.M. van der Heijden, C. Kempenaar, T. van der Zalm, and L.A.P. Lotz. A robot to detect and control broad-leaved dock (rumex obtusifolius l.) in grassland. *J. Field Robot.*, 28(2):264–277, 2011.

[van Henten *et al.*, 2002] E.J. van Henten, J. Hemming, B.A.J. Van Tuijl, J.G. Kornet, J. Meuleman, J. Bontsema, and E.A. Van Os. An autonomous robot for harvesting cucumbers in greenhouses. *Auton. Robot.*, 13(3):241–258, 2002.

[Van Willigenburg *et al.*, 2004] L.G. Van Willigenburg, C.W.J. Hol, and E.J. Van Henten. On-line near minimum-time path planning and control of an industrial robot for picking fruits. *Comput. Electron. Agr.*, 44(3):223–237, 2004.

[Yoo *et al.*, 2013] C. Yoo, R. Fitch, and S. Sukkarieh. Provably-correct stochastic motion planning with safety constraints. In *Proc. of IEEE ICRA*, pages 981–986, 2013.