2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Task Space Motion Planning Decomposition

Nathan Larkin[1,2], Andrew Short[1], Zengxi Pan[1,2] and Stephen van Duin[1,2]

***Abstract*— In autonomous robotics there are many situations that require solving a motion planning problem to complete a task. A Task Space (T-Space), composed of parameters that define the task being performed, can be a more effective planning space for these problems, however, planning within a T-Space is often computationally challenging. In this paper, we present a novel method to analyse the relationship between T-Space parameters and the pose of manipulator bodies to create a dependency matrix. We then use this information to decompose the motion planning problem into sequential lower complexity sub-problems. We call this approach Task Space Motion Planning Decomposition (TSMPD). This paper introduces TSMPD and quantifies the improvement to planning efficiency on a challenging maze navigation problem and weld path planning problem.**

I. Introduction

There are many situations in autonomous robotics that require planning a motion to perform a specific task, rather than just moving between two points. Examples include the motion to open a door, or to perform a complex weld. These types of problem require a different approach to that typically used for high Degree of Freedom (DOF) robotic systems, where motion planning is typically performed in Configuration Space (C-Space) [1]. An alternative is to perform motion planning in Task Space (T-Space), which is defined by the task being performed. For many problems T-Space offers a more effective space for planning than C-Space: for example, consider a serial link manipulator moving an object; it is more natural to plan the task in terms of the object's position rather than the joint angles of the manipulator [2]. However, planning within T-Space is often computationally challenging [1], particularly for highly redundant manipulators [3].

In some cases the workspace pose of manipulator bodies is aligned with T-Space dimensions in such a way that they can be considered to be decoupled. This means that moving along some T-Space dimensions may not change the pose of all manipulator bodies. This phenomenon can be exploited to simplify a T-Space problem to improve solving efficiency, or to allow employing a greedy search algorithm to improve the quality of the solution.

We present a novel method to simplify a T-Space motion planning problem by identifying decoupled T-Space dimensions and manipulator bodies. Our method analyses the effect of each T-Space dimension on the pose of each manipulator body to eliminate redundant collision checks.

[1]School of Mechanical, Materials, Mechatronic and Biomedical Engineering, University of Wollongong, NSW 2522 Australia
[2]Defence Materials Technology Centre, Australia

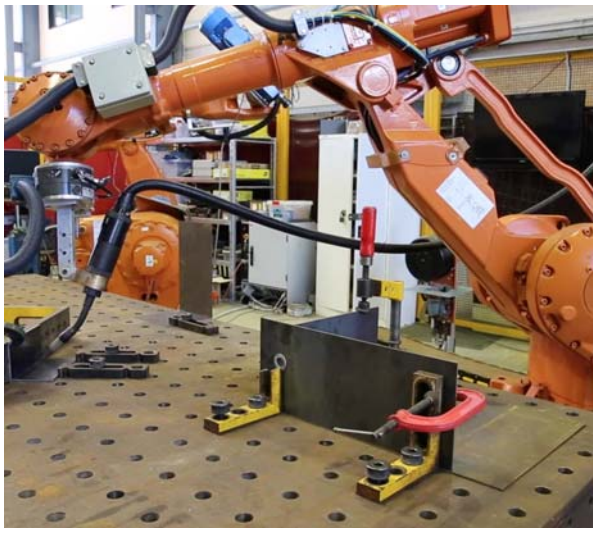

Fig. 1. Robot welding plates using paths planned with TSMPD

We call this method Task-Space Motion Planning Decomposition (TSMPD). This approach was previously alluded to in [4] and [5] applied to planning for welding. We will detail the TSMPD approach and two ways to implement the results to optimise planning. The first is to separate the T-Space planning problem into smaller sub-problems to be solved sequentially. The second is a method that performs collision checks in a hierarchy defined by the TSMPD algorithm.

The paper begins by covering the background and related work followed by the problem definition of the T-Space motion planning problem, and subsequent decomposition problem. Sec. IV-A will illustrate a simpler example before we formally explain the TSMPD algorithm in Alg. IV-B. Finally in Sec. V, we will quantify the computational efficiency improvement due to the TSMPD algorithm when compared to other approaches to solving T-Space motion planning with two examples. In this paper we will limit the problem to those with only a single serial manipulator, however this approach can be extended to multiple manipulators. We also assume a static, known environment and kinematic structure.

II. Related Work

T-Space is composed of parameters that define the robot task being performed. For a camera inspection task, the T-Space parameters may be the camera's focal point and rotation angles; for welding it may be the torch position and rotation angles. A T-Space configuration is a single point within T-Space, analogous to a configuration in C-Space. In some cases, there is a one-to-one mapping between T-Space

978-1-5386-3081-5/18/$31.00 ©2018 IEEE

and C-Space [6] but robot redundancies may mean this is not always the case [2]. For many applications, T-Space defines the set of end-effector poses, allowing for an intuitive problem formulation when interacting with workspace objects [2]. Examples of this include opening a door [7], solving a maze while maintaining end-effector orientation [6], or dragging an object across a table [8].

T-Space problems are often defined with partial constraints on parameters. Yao and Gupta [6] specify a series of equations that must be satisfied for each end-effector pose along a path. Stilman [7] used a motion constraint vector attached to a task frame which indicated positional or rotational degrees of freedom that must remain fixed during a movement. These constraints form a manifold, which is difficult to represent and poses challenges for mapping to other spaces. These constraint manifolds can also be lower dimensional than the C-Space they map to, introducing a null space [9].

Researchers have taken several approaches to solving the motion planning problem in T-Space. Greedy algorithms have been used with limited success to plan motions on the constraint manifold. The best-first approach can generate optimal motions, but their exhaustive nature makes them often unsuitable for high-dimensional problems. Yao and Gupta [10] used a depth-first search for trajectory tracking with a redundant robot, and found this approach was only feasible for simple environments and short paths.

Sampling Based Planners (SBPs) have proved to be an effective method for solving the motion planning problem in high-dimensional spaces by using random sampling to approximate configuration space [11]. The sampling-based approach means that an explicit environment representation is not required, and the global nature prevents planning from becoming trapped by local minima [11]. The two most common variants of SBP are the Probabilistic Roadmap Method (PRM) [12] and Rapidly-exploring Random Trees (RRTs) [13]. There are three methods for ensuing that samples satisfy constraints: rejection validation, projection, and direct sampling. Typically, SBPs use rejection validation, in which a random sample is chosen and then discarded if it does not meet the constraints. However, for all but the most trivial constraints the likelihood of a sample satisfying the constraints is too low for this to be a feasible approach [14].

Projection adjusts generated samples until they lie on the constraint manifold. Randomised Gradient Descent (RGD) was used by LaValle et al. to incrementally repair samples until they satisfied a closed-loop closure constraint [14]. RGD was adapted by Yao and Gupta [6] to project samples onto a general constraint manifold. Stilman applied the RGD, tangent-space sampling, and first-order approaches to projection [7]. The Constrained Bi-Direction RRT (CBiRRT) by Berenson et al. [9] has been demonstrated for a wide number of constraints and use cases. This method grows a tree towards a random sample in the workspace, stepping towards the sample and continually projecting onto the constraint manifold. Projection approaches are widely applicable to SBP, but depend on the efficiency of the projection technique and form of the constraint manifold [15].

An alternative to projection is to take samples directly from T-Space, ensuring that all samples lie on the constraint manifold. Alternate T-Space and C-Space Exploration (ATACE) [6] samples T-Space for paths that satisfy constraints, and then maps them to C-Space with a constrained local planner. Task-space RRT [2] extends RRT by growing the tree in T-Space using the Jacobian Pseudo-inverse, and also allows for direct sampling from T-Space. Tangent-space RRT [15] attempts to generate samples in the tangent spaces of the constraint manifold to minimise the need for projection.

To date researchers have applied a number of varying methods to solving the T-Space motion planning problem with success. Our approach does not look at the issue of solving a T-Space problem specifically. Instead, we describe a method to automatically decompose a complex T-Space problem into smaller, sequential sub-problems with reduced T-Space parameters and manipulator bodies. The approach of decomposing a high-DOF problem into smaller sub-problems has previously been validated for C-Space in [16]. After decomposition, many of the existing techniques described can be applied to each sub-problem with improved efficiency.

III. Problem Definition

A. Task Space Motion Planning Problem

The T-Space motion planning problem can be defined as a manipulator in a workspace $\mathcal{W} \in \mathrm{R}^3$ which has a configuration space (C-Space) containing all possible configurations. For a typical serial link manipulator a configuration q is a vector of joint positions. The forward kinematic function $FK(q)$ specifies the pose of each manipulator body for configuration q. The Inverse Kinematics (IK) function $IK(p, cfg)$ describes the C-Space configuration for end-effector pose p and configuration specification cfg to describe the solution where multiple exist. A T-Space T consists of the task-specific parameters. A individual T-Space configuration is denoted by $s \in T$. The task space kinematic equation $TK(s) \rightarrow (p, cfg)$ describes the pose of of the end-effector and robot configuration for T-Space configuration s. This information can be input into IK to determine the pose of all bodies for s.

Using this notation the T-Space motion planning problem is formulated as: Given a start and goal described by T-Space configurations, determine a reachable, continuous, collision free path that satisfies all task-specific constraints.

B. Motion Planning Decomposition

We extend this problem with a set of manipulator bodies B. Each body has a pose $p = (\mathcal{P}, \mathcal{O})$. The position $\mathcal{P}$ is a three-length vector, and the orientation $\mathcal{O}$ is a 3×3 rotation matrix. Each body can also have a symmetry matrix associated with it as per Sec. III-C.

Using this notation the TSMPD problem is formulated as: Given a set of bodies B with position and orientation set by the forward kinematic function $FK(q)$, where q is defined by the IK function $IK(p, cfg)$, and (p, cfg) are described by the T-Space kinematic equation $TK(s)$, identify a set of sequential sub-problems each with a subset of T-Space

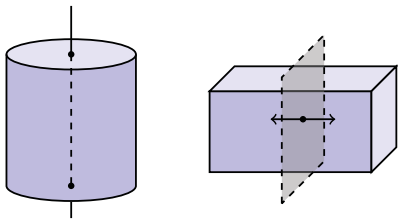

Fig. 2. Axis and plane body symmetry

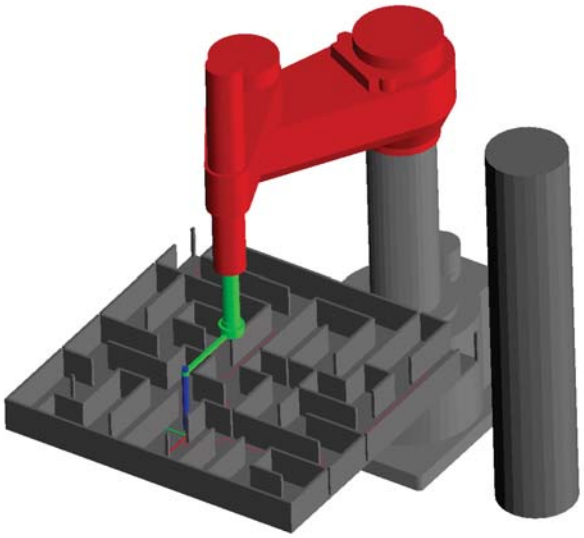

Fig. 3. SCARA robot tasked with tracing a line through a maze

parameters and body collision objects such that that when solved form a valid solution to the overall T-Space motion planning problem (defined in Sec. III-A).

C. Body Symmetry

Bodies may also have a notion of symmetry: that is that some operations may change the pose of the body without changing its physical shape. This forms a critical part of the TSMPD algorithm as redundant collision checks resulting from symmetrically equivalent poses can be removed. There are two types of body symmetry as shown in Fig. 2: rotation around an axis and mirroring around a plane. These symmetry operations can be represented by a 3×3 matrix S which is multiplied with the orientation of the body. For the same body, two rotations $\mathcal{O}_1$ and $\mathcal{O}_2$ are considered equivalent under symmetry if $\mathcal{O}_1 S = \mathcal{O}_2 S$.

IV. Algorithm

A. A Simple Illustration

We will start with a simple example to introduce our algorithm, before moving onto a formal description and challenging problems. Consider a 4-axis SCARA robot with the task of drawing a line through a maze with a pen mounted parallel to the rotation axis of the end effector as shown in Fig. 3. In this case the position of the end effector is constrained to the maze plane, pen and robot cannot hit any obstacles, pen is free to rotate around its z axis and the robot

Body	Parameter			
	x	y	R_z	cfg
Base	✗	✗	✗	✗
J_1	✓	✓	✓	✓
J_2	✓	✓	✓	✓
J_3	✓	✓	✓	✓
J_4	✓	✓	✓	✗
Pen mount	✓	✓	✓	✗
Pen	✓	✓	✗	✗

TABLE I
DEPENDENCY MATRIX FOR A 4-DOF PROBLEM

can take a right-hand or left-hand configuration. The motion planning problem is: find a continuous collision free path from start to goal along the maze plane where the start and goal are represented by local x, and y coordinates in the maze.

For this case the T-Space defines the location of the pen tip in local two-dimensional Cartesian coordinates located on a plane, the rotation of the pen, and the configuration of the robot solution. Formally the T-Space is can be defined by the following parameters:

x, y the x and y position in the maze, $x, y \in \mathrm{R}$.
R_z the rotation around the z axis of the end-effector frame.
cfg the robot configuration $cfg \in \{0, 1\}$ where 0 represents the left-handed kinematic solution and 1 represents the right-handed solution.

The T-Space kinematic equation $TK(s)$ describes the pose of the pen in the robot base frame as detailed by Eqns. 1–2 where T^{pen}_{maze} is the transformation matrix calculated in Eqn. 3. The configuration $q \in \mathrm{R}^4$ is then calculated using the appropriate IK equation. In addition, the Pen body has symmetry S around the z axis detailed in Eqn. 4.

$$TK(x, y, R_z, cfg) = (T^{pen}_{base}, cfg) \tag{1}$$

$$T^{pen}_{base} = T^{maze}_{base} \times T^{pen}_{maze} \tag{2}$$

$$T^{maze}_{pen} = \begin{bmatrix} cos(R_z) & sin(R_z) & 0 & x \\ sin(R_z) & -cos(R_z) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$S = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The first step of the TSMPD algorithm is to determine dependent body motion for each T-Space parameter. For example, if the T-Space parameter x is modified which body poses change? Table I shows this dependency matrix. It can be noted that the Pen body is not dependent on the parameter R_z. This is due to the axis symmetry that exists around the z direction of the Pen body.

Order	Parameter	Body
	-	Base
1	x	
2	y	Pen
3	R_z	Pen mount, J_4
4	cfg	J_1, J_2, J_3

TABLE II

ALLOWED PARAMETER ORDER FOR 4-DOF PROBLEM

The second stage is to define the search order for the T-Space parameters. This is done by sorting each T-Space parameter by the number of dependent bodies in descending order. In this example the ordered T-Space search list is: x, y, R_z, cfg. The x and y parameters have equal dependencies and can be represented in either order.

The third stage is to allocate bodies for collision check to the T-Space parameter list. The aim of this process is to allocate each body with the minimum number of set T-Space parameters. For this example, the robot base is not dependent on any parameters, therefore the collision check can be completed before any are set. After the first parameter in the order is set, x, there are no bodies that are not dependent on a non-allocated parameter therefore we move to the next parameter. After the parameter y is set, the Pen which is not dependent on any unallocated parameters can be set. In effect, this process is substituting the T-Space equations into the IK equation for the pen and determining that the pose of the pen is not a function of the parameters R_z or cfg. This process continues until all bodies have been allocated resulting in the order, parameter and body allocation shown in Table II.

This can be considered a decomposition of the original T-Space problem. Where we started with a 4-DOF problem, we now have three less complex sub-problems:

(a) Choose values for $\{x, y\}$, check collision with Pen, then
(b) choose value for $\{R_z\}$, check collision with Pen mount, J_4, then
(c) choose value for $\{cfg\}$, check collision with J_{1-3}.

To illustrate these results the bodies in Fig. 3 are colour coded to represent the different body allocation as per the TSMPD results. The Pen is coloured in blue, the Pen mount and J_4 are coloured green, and J_1, J_2, J_3 are coloured red.

Another way to consider this is that we have created an optimised search order of T-Space parameters and body collision checks such that the minimum number of collision checks invalidates the maximum number of T-Space configurations. If the Pen body is colliding for a certain $\{x, y\}$ value for example, there will be no $\{R_z\}$ or $\{cfg\}$ that will not result in collision.

B. Algorithm Description

In this section we formally describe the TSMPD algorithm. There are three parts to the TSMPD algorithm:

1) Determine the dependency matrix for each body against the T-Space parameters
2) Order the T-Space parameter search.
3) Allocate bodies for collision checking and determine each sub-problem.

1) Dependency Identification: The dependency identification algorithm determines the dependency between each body $b \in B$ and the T-Space parameters T. This forms a boolean dependency matrix as shown in Eqn. 5 where D_{ij} is true if the pose of B_i is affected by T-Space parameter T_j.

$$D_{ij} = \frac{dB_i}{dT_j} \neq 0 \; \forall \; i \in [0, n], j \in [0, m] \tag{5}$$

We utilise a sampling based approach to solve for these relationships where a single T-Space parameter t is varied within a random T-Space configuration u_a to generate pose sets P_1 and P_2. This algorithm is detailed in Alg. 1 and this is invoked for all $t \in T$ to form the dependency matrix D shown in Eqn. 5. The SAMPLE function uniformly samples T-Space configuration or individual parameter value. The VALID function then checks if the T-Space kinematics function $TK(s)$ can be successfully solved. ISPOSEEQUIVALENT then tests if two body poses are equivalent taking into account symmetry matrix S as defined in Sec. III-C.

Algorithm 1 Calculate the bodies which have a pose dependent on parameter t

```
samples ← 0
while samples < required samples do
    u_a ← SAMPLE(T), u_b ← u_a
    u_b(t) = SAMPLE(t) | u_a(t) ≠ u_b(t)
    if VALID(u_a) & VALID(u_b) then
        P_1 ← TK(u_a)
        P_2 ← TK(u_b)
        for all b ∈ B do
            if ¬ISPOSEEQUIVALENT(P_1b, P_2b) then
                D_bt ← true        ▷ Mark b dependent of t
            end if
        end for
        samples ← samples + 1
    end if
end while
```

2) Parameter Ordering: In the TSMPD algorithm, the search order for T-Space parameters is from the parameter with highest number of dependent bodies to that with the least dependent bodies. In D, each column corresponds to a T-Space parameter and can be summed to find the number of bodies which depend on that parameter. Thus a search order permutation U can be found by sorting T in descending order according to the values of Eqn. 6.

$$\text{sort} = \left\{ \text{sum}(D_i^T) \;\middle|\; i \in [0, m] \right\} \tag{6}$$

3) Assigning Bodies: The final part of the TSMPD algorithm is to iterate through the sorted T-Space order U and allocate bodies that require collision checking when each parameter is assign, shown in Alg. 2. Sec. IV-B.2 creates an

optimised order to search through T-Space to solve planning problems, decomposing the problem as much as possible. As each parameter is assigned, a subset of bodies must be collision checked to verify validity. If, after a parameter has been assigned, a body is found to be in collision, then the remaining parameters do not affect the pose of this body and will not resolve this collision. As such, this candidate solution can be rejected without any further collision checks.

Algorithm 2 Assign bodies to collision check when values in U are assigned

```
for all t ∈ U do
    t ← assigned
    for all b ∈ B do
        count ← 0
        for all t ∈ T do
            if t = assigned then
                count ← count + D_bt
            end if
        end for
        if count = 0 then
            Assign b to t
        end if
    end for
end for
```

C. Algorithm Implementation

We couple the decomposition with existing planning approaches. In this section we outline two implementations: a collision cache and a sequential sub-problem search.

One method is to stage collision checks to invalidate a T-Space configuration with a smaller subset of geometry. We can then take this a step further and memoise collision results for each decomposed parameter subset and associated bodies in a collision cache. Sampling methodologies will need to be adjusted using an approach such as discretisation, or sub-sampling existing results to ensure partial collision results can be re-used.

Another method is to find a solution for each sub-problem sequentially. For the example in IV-A this would mean solving a path for the Pen body with parameters $\{x, y\}$ through the maze, then find a solution for R_z and the Pen mount, J_4 using the previously solved path, and finally determining cfg for J_1, J_2, J_3.

The completeness and optimality of the planning approach using TSMPD depends on the underlying implementation details. For example, if TSMPD is only used to cache collision results internally in a planner, our approach will inherit the completeness properties of the original planner. However, if a sequential search is used without full backtracking then completeness will be lost. As such the underlying planner must be carefully chosen.

V. Results

In this section we examine the performance of two T-Space problems that have been decomposed using our TSMPD algorithm. The first problem is the example of a SCARA robot tracing a path through a maze used in Sec. IV-A, the second problem is to solve the welding path for a common six axis industrial robot with a linear external axis.

Parameter	Discretisation Step	Unit
x	10	mm
y	10	mm
R_z	10	deg
cfg	-	-

TABLE III
DISCRETISATION OF PARAMETERS USED FOR SCARA MAZE PROBLEM

A. SCARA Robot in Maze

In this problem we use the example described in Sec. IV-A for a SCARA robot to draw a line through a maze as shown in Fig. 3. To create a more challenging problem, the path is partially obstructed by a post to force a tool rotation in the solution. We apply two approaches to solve the T-Space problem, a PRM approach that samples directly in T-Space, and an A* algorithm with a cost function applied to minimise the total path length. In each case we implement a cached collision hierarchy as generated by our TSMPD algorithm and compare the solving efficiency in terms of time and number of collision checks required.

As previously shown, our TSMPD algorithm results in a parameter and body allocation shown in Table II. We use this information to structure the collision check operation as:

(a) check collision with Pen body at $\{x, y\}$, memoise, then
(b) check for collision with pen mount, J_4 at $\{x, y, R_z\}$, memoise, then
(c) check for collision with J_1, J_2, and J_3 at $\{x, y, R_z, cfg\}$.

To make best use of the collision cache, the T-Space is discretised as detailed in Table III. This approach couples well with graph based search algorithms such as A* [17].

Each test was run 10 times with the minimum, maximum and average results shown in Table IV. These results clearly show the improved efficiency of the TSMPD algorithm. When solving with the A* approach, solve times were on average five times faster by implementing a collision cache optimised from the TSMPD results. When using the T-Space PRM approach solve times were on average three times faster. For both methods we were able to reuse at least some of the stored collision information in 98% of queries.

As expected the path length using the PRM approach was longer than that using the A* approach with a path length minimisation cost function. What should be noted here is that without the cache, the solve time for the A* and PRM approaches were similar, however with the cache, the A* solution was found in half the time of the PRM solution, This demonstrates that greedy search methods can be utilised to maximise the quality of the result when coupled with a decomposed T-Space problem.

Metric	T-Space PRM		A*	
	without cache	with cache	without cache	with cache
Planning Time (s)	16.1 · **73.1** · 160.8	8.3 · **22.6** · 43.4	60.6 · **61.3** · 62.7	11.9 · **12.1** · 12.4
Path Length (mm)	2600 · **3083** · 3684	2600 · **3083** · 3684	1941	1941
Collision Checks ($\times 10^3$)	20.1 · **274** · 631	4.85 · **5.53** · 6.23	221	3.73
Cache Lookups ($\times 10^3$)	-	58.1 · **308** · 707	-	217

TABLE IV

PLANNING RESULTS FOR THE SCARA MAZE PROBLEM, SHOWING MINIMUM · **AVERAGE** · MAXIMUM VALUES

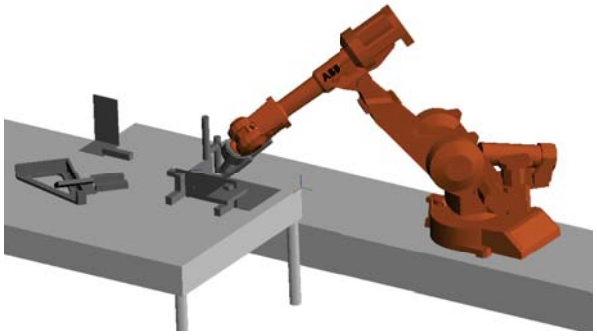

Fig. 4. Welding robot tasked to perform a set of welds

B. Welding Robot

Planning an arc welding path is a common motion planning problem best described using a T-Space. There is path a to b that the torch tip needs to follow with some allowable adjustments that can be made to reach into tight areas and avoid collision. In this example we have a 6-DOF manipulator with an external axis to perform a number of welds as shown in Figs. 1 and 4. We define the T-Space as:

R_x, R_y rotation around the x and y axes of the end effector frame. Has a subtle effect on the resulting weld quality.

R_z the rotation around the z axis of the end-effector frame. The z axis is coincident with the welding wire and does not have a quality cost.

$ctwd$ translation in the end-effector frame along the z axis.

cfg the robot configuration $cfg \in \{0, 7\}$ where each integer number represents a different configuration that results in the equivalent end-effector pose.

$rail$ the position of the robot base along the external axis.

t the position along the path from a to b in $[0, 1]$

The T-Space kinematic equation $TK(s)$ is defined by Eqns. 7–8. T_0^{target} represents the transformation matrix at position t along path $a \to b$. This is determined using a transform interpolation function (Eqn. 9). Finally, T_{target}^{tool} applies the local T-Space configuration parameters $R_x, R_y, R_z, ctwd$ to the tool position as shown in Eqn. 10.

$$TK(R_x, R_y, R_z, ctwd, cfg, rail, t) = T_0^{tool} \quad (7)$$

$$T_0^{tool} = T_0^{target} \times T_{target}^{tool} \quad (8)$$

$$T_0^{tool} = a + t \times (b - a) \quad (9)$$

$$T_{target}^{tool} = \begin{bmatrix} & & & 0 \\ & Rot_x(R_x)Rot_y(R_y)Rot_z(R_z) & & 0 \\ & & & ctwd \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

To solve this problem, we first utilise the TSMPD algorithm resulting in the dependency matrix shown in Table V and the parameter collision body order in Table VI. This is then used to optimise the overall T-Space motion planning problem. In this example we have implemented the TSMPD algorithm result as a set of sequential sub-problems that when solved will form a solution to the overall problem.

(a) For the Nozzle, find a collision free path $\forall t \in [0, 1]$ for parameter subset $R_x, R_y, ctwd$, then

(b) for the Torch and J_6, find a collision free path for R_z using the previously found Nozzle path, then

(c) for the Base, find a collision free path for parameter $rail$ using the previously found Torch path, then

(d) for J_{1-5}, find a collision free path for parameter cfg using the previously found Base path.

A caveat of this process is that a secondary sub-problem may not have a solution based on the outcome of a previous sub-problem. In such a case we invalidate the solution from the previous problem, backtrack and re-solve.

For this T-Space problem we use a discretised T-Space graph planner similar to the Descartes Cartesian motion planner [18]. In our case we utilise a depth first search algorithm with a cost function related to the expected weld quality. We compare our optimised approach of solving the decomposed sub-problems with directly solving the original T-Space problem. Results are shown in Table VII.

All welds showed improvement using the optimised approach to a varying degree. Problems that were solved quickly using the original problem showed a smaller improvement than those that were more complex. This is to be expected using a depth first search methodology where little adjustment is required from the optimal result. For the most complex weld (a), a 3.5 times improvement was recorded.

VI. CONCLUSION

In this paper we describe an algorithm to identify decoupled collision bodies from parameters in a T-Space motion planning problem. We are then able to create a set of lower complexity sub-problems that can be used to minimise

Body	Parameter						
	t	R_x	R_y	R_z	$ctwd$	cfg	$rail$
Base	✗	✗	✗	✗	✗	✗	✓
$J_1 - J_5$	✓	✓	✓	✓	✓	✓	✓
J_6	✓	✓	✓	✓	✓	✗	✗
Torch	✓	✓	✓	✓	✓	✗	✗
Nozzle	✓	✓	✓	✗	✓	✗	✗

TABLE V

DEPENDENCY MATRIX FOR THE ARC WELDING PROBLEM

Order	Parameters	Body
1	t, R_x, R_y	
2	$ctwd$	Nozzle
3	R_z	Torch, J_6
4	$rail$	Base
5	cfg	J_1, J_2, J_3, J_4, J_5

TABLE VI

ALLOWED PARAMETER ORDER FOR ARC WELDING PROBLEM

collision checks. We tested our approach on two T-Space motion planning problems: drawing a line through a maze using a SCARA robot, and performing a set of welds with a 6-DOF robotic arm on a linear rail.

By using the decomposed T-Space sub-problems resulting from our algorithm we achieved a five fold reduction in planning time for the SCARA maze problem using an A* approach and a three fold reduction using a T-Space PRM approach. In the welding problem, using the decomposed approach resulted in improvement for all welds tested. For the most difficult weld a 3.5 times improvement was found using the decomposition over the original T-Space problem.

TSMPD can optimise a wide range of T-Space problems using information implicit in the task definition. It exploits the relationship between the manipulator kinematics and the task being performed to eliminate unnecessary collision checks, leading to a dramatic reduction in planning times. An added advantage is that greedy search strategies, such as A*, can be successfully applied to complex problems.

Future work includes applying this approach to more challenging and cluttered problems. We also hope to analyse the completeness and optimality of TSMPD coupled with existing common planning approaches.

Weld	Original (s)	Decomposed (s)	Improvement
a	78.9	22.5	3.5
b	1.6	1.3	1.2
c	1.5	1.1	1.4
d	1.0	0.9	1.1
e	0.4	0.2	1.7
f	13.8	7.9	1.7

TABLE VII

AVERAGE PLANNING TIME FOR THE ARC-WELDING PROBLEM

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of the Defence Materials Technology Centre (DMTC), which was established and is supported by the Australian Government's Defence Future Capability Technology Centre (DFCTC) initiative.

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[2] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2061–2067.

[3] G. Oriolo, M. Ottavi, and M. Vendittelli, "Probabilistic motion planning for redundant robots along given end-effector paths," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2. IEEE, 2002, pp. 1657–1662.

[4] N. Larkin, A. Short, Z. Pan, and S. van Duin, "Automatic program generation for welding robots from CAD," in *Advanced Intelligent Mechatronics (AIM), 2016 IEEE International Conference on*. IEEE, 2016, pp. 560–565.

[5] N. Larkin, A. Short, Z. Pan, and S. van Duin, "Automated programming for robotic welding," in *Transactions on Intelligent Welding Manufacturing*, S. Chen, Y. Zhang, and Z. Feng, Eds. Springer, Singapore, 2018, pp. 48–59.

[6] Z. Yao and K. Gupta, "Path planning with general end-effector constraints," *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 316–327, 2007.

[7] M. Stilman, "Task constrained motion planning in robot joint space," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 3074–3081.

[8] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 625–632.

[9] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, 2011.

[10] Z. Yao and K. Gupta, "Self-motion graph in path planning for redundant robots along specified end-effector paths," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2004–2009.

[11] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[12] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[13] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[14] S. M. LaValle, J. H. Yakey, and L. E. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 3. IEEE, 1999, pp. 1671–1676.

[15] C. Suh, T. T. Um, B. Kim, H. Noh, M. Kim, and F. C. Park, "Tangent space rrt: A randomized planning algorithm on constraint manifolds," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4968–4973.

[16] V. Pilania and K. Gupta, "A hierarchical and adaptive mobile manipulator planner with base pose uncertainty," *Autonomous Robots*, vol. 39, no. 1, pp. 65–85, Jun 2015. [Online]. Available: https://doi.org/10.1007/s10514-015-9427-2

[17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[18] S. Edwards, "The descartes planning library for semi-constrained cartesian trajectories." in *ROSCon 2015*, September 2015. [Online]. Available: https://vimeo.com/142622435