UNIVERSITY OF TECHNOLOGY SYDNEY

Uncertainty Modelling and Motion Planning of an Inchworm Robot Navigating in Complex Structural Environments

by

David Pagano

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering and IT
Intelligent Mechatronic Systems Group

July 2018

Declaration of Authorship

I, David Pagano , declare that this thesis titled, 'Uncertainty Modelling and Motion Planning of an Inchworm Robot Navigating in Complex Structural Environments' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Production Note:
Signature removed prior to publication.

Date: 30/7/2018

UNIVERSITY OF TECHNOLOGY SYDNEY

Abstract

Faculty of Engineering and IT

Intelligent Mechatronic Systems Group

Doctor of Philosophy

by David Pagano

Many ferromagnetic structures require continuous inspection and maintenance routines to ensure longevity, structural integrity and aesthetics. For most structures, routines are performed by teams of personnel, with each individual performing specific tasks. These tasks may be highly hazardous; being performed at height, in confined spaces or in the presences of hazardous materials such as lead based paints and vehicle fumes. Adopting a robotic solution for inspections would significantly improve occupational health and safety for maintenance personnel, while increasing the quality and reducing the cost.

An inchworm robot has been developed for inspection of confined spaces in the Sydney Harbour Bridge. With a 7 degree of freedom multi-link serial body and magnetic pads for adhesion, the inchworm robot provides a dexterous means for climbing and inspecting particularly difficult-to-access sections of the bridge. However, due to the structure and the adhesion mechanism of the inchworm type robot, deformation of the robot body (i.e. structural uncertainty) and inaccurate landing position of the permanent magnet adhesion pads (i.e. hand position uncertainty) cause imperfect knowledge about the robot state. This prevents safe motion in a real world setting. The combination of these uncertainties present a unique challenge in robot motion planning and collision avoidance which is not considered in the literature.

This thesis first focuses on developing a model for representing the structural and hand position uncertainties. The model describes the uncertainty in the coordinate frame of reference for the joints.

A 3D probabilistic force field (3D-PF^2) algorithm is developed to incorporate the uncertainties and allow for smooth, collision-free path planning. A force field surrounds each link to prevent collisions with each force field sized to account for the dimensions of the link and the uncertainty at the joints related to the link. Force fields are used to generate repulsive forces which push the robot away from obstructions while an attractive force pulls the end-effector towards a goal location.

A Line of Sight Tree (LoST) algorithm is developed for longer time-horizon motion planning with the 3D-PF^2 algorithm used for local motion planning. The LoST algorithm provides waypoints as goal locations for the 3D-PF^2 algorithm. Waypoints are found in a manner loosely based on the way a person views a scene whereby their gaze tends towards important regions such as the edges of objects.

Extensive simulations and experiments have been conducted to test the performance of both the 3D-PF^2 and the LoST algorithms within a number of environments including the specific application scenario at the Sydney Harbour Bridge.

Acknowledgements

First I would like to thank my supervisor Professor Dikai Liu for all the support he has given me - intellectual, mental and otherwise - throughout the project to allow me to attain the quality of the final thesis.

Thanks to the CROC team including Professor Ken Waldren, Phillip Quin, Gavin Paul, Peter Ward, John Yang for their hours of work into the project from which my work is founded. A special thank you to Phillip for the chats over the years, and to Peter and John for all the help with the experiments.

Thanks to all my friends who have supported me emotionally and mentally throughout this entire process. The list is long and the reasons vast; each of you have helped me in some way, in some cases it may be unknowingly. I am grateful.

And finally I would like to thank my family; Sister, Mother and Father. These last few years haven't been the greatest but they have been there for me through it all and without them this thesis would not have come to pass.

Nonni and Zia...I miss you all...

Contents

List of Figures

List of Tables

Abbreviations

3D-F^2	3 Dimensional Force Field
3D-PF^2	3 Dimensional Probabilistic Force Field
COG	Centre of Gravity
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
FEA	Finite Element Analysis
IMU	Inertial Measurement Unit
HMI	Human-Machine Interface
GA	Genetic Algorithm
LOS	Line of Sight
LoST	Line of Sight Tree
LQG	Linear-quadratic Gaussian
MDP	Markov Decision Process
NSW	New South Wales (Australia)
PF	Potential Field
POMDP	Partially Observable Markov Decision Process
PRM	Probabilistic Roadmap
RANSAC	Random sample consensus
RGB-D	Red, green, blue and depth
RRT	Rapidly-exploring Random Tree
SLAM	Simultaneous Localisation and Mapping
UTS	University of Technology Sydney
VFF	Virtual Force Field

Nomenclature

	General Formatting Style
$f(\cdots)$	A scalar valued function
$\mathbf{f}(\cdots)$	A vector valued function
$[\cdots]^T$	Transpose
$\lvert \cdot \rvert$	Absolute value
$\lVert \cdot \rVert$	Vector length and normalised vector
	Local and Global Variables
ψ_G	The global coordinate frame
ψ_R	Coordinate frame of reference of a link
i, j, k	Variables signifying the index of and counts associated with joints and links
	Specific Symbol Usage
${}^{i-1}T_i$	A homogeneous transformation matrix (4 by 4)
a	A DH parameter describing the distance along the x-axis
d	A DH parameter describing the distance along the z-axis
α	A DH parameter describing the rotation about the z-axis
$\mathbf{R}$	A rotation matrix (3 by 3)
$\mathbf{R_n}$, $\mathbf{R_a}$, $\mathbf{R_o}$	The normal, approach and orientation vectors of a rotation matrix (3 by 1)
n_x, n_y, n_z	Components of normal vector
a_x, a_y, a_z	Components of approach vector
o_x, o_y, o_z	Components of orientation vector

$\mathbf{P}$	A position vector (3 by 1)
x, y, z	Components of the position vector along a x-, y- and z-axis
$\mathbf{P_{xy}}$	A position vector along the xy-plane (2 by 1)
n	Number of joints in the multi-link serial robot
$\mathbf{q}$	A joint position matrix (n by 1)
q	A component of the joint position matrix
${}^{0}\mathbf{P_g}$	Goal location position vector (3 by 1)
${}^{0}\mathbf{R_g}$	Goal location orientation vector (3 by 1)
l_c	Distance between the base and the end-effector
E_c	Allowed maximum structural translational deformation
γ_c	Allowed maximum structural rotational deformation
d_{xy}	Maximum distance along the xy-plane between a joint's and a preceding joint's coordinate frames of reference
β_c	Joint distance ratio
$G_c(\cdots)$	Function relating allowed maximum translational deformation to the joint distance ratio
$F_c(\cdots)$	Function relating allowed maximum rotational deformation to the joint distance ratio
$\mu_{E.i}$	Estimate of the mean structural translational deformation at a joint
$\mu_{\gamma.i}$	Estimate of the mean structural rotational deformation at a joint
$\sigma_{E.i}$	Structural translational variance
$\boldsymbol{\sigma}_{E.i}$	A matrix describing the structural translational variance in 3D (3 by 3)
$\mu_{x.h}$, $\mu_{y.h}$	Hand position mean along the x- and y-axis
$J_c(\cdots)$	Function relating the hand position variance to the joint position ratio
d_a, d_b, d_e	Preparatory approach, surface approach and end-effector distances to the surface goal location

K_g	The growth factor
σ_h	Hand position variance
$\boldsymbol{\sigma}_h$	A matrix describing the hand position variance in 3D (3 by 3)
$\boldsymbol{\sigma}$	A matrix describing the variance at the joint (3 by 3)
$\mathcal{T}$	A homogeneous transformation matrix with uncertainty (4 by 4)
$\mathcal{R}$	A rotation matrix with uncertainty (3 by 3)
$\mathcal{P}$	A position vector with uncertainty (3 by 1)
$\mathbf{I}$	Mass-inertia matrix of the robot links
$\boldsymbol{\beta}$	Damping coefficient matrix of the robot joints
$\mathbf{k}_{sp}$	Stiffness coefficient matrix of the robot joints
$\boldsymbol{\Gamma}$	A torque-force matrix defining in joint space (n by 1)
τ	Joint torque-force of a joint, a component in Γ
$\dot{\mathbf{q}}$	A joint velocity matrix (n by 1)
$\ddot{\mathbf{q}}$	A joint acceleration matrix (n by 1)
$\mathbf{H}$	A force transformation matrix
$\mathbf{h}$	Component of the force transformation matrix, H
$\mathbf{M}$	A rotation skew-symmetric matrix
$\mathbf{L}$	Configuration matrix of a joint
$l_{r.x}$, $l_{r.y}$, $l_{r.z}$	Rotational components of the configuration matrix of a joint
$l_{t.x}$, $l_{t.y}$, $l_{t.z}$	Translational components of the configuration matrix of a joint
$\boldsymbol{\mathcal{H}}$	A probabilistic force transformation matrix
$\boldsymbol{\mathcal{M}}$	A probabilistic rotation skew-symmetric matrix
$\hat{\mathbf{h}}$	Component of the probabilistic force transformation matrix, H
S	A skew-symmetric matrix (3 by 3)
$\mathbf{F}$	A 6DOF spatial force acting on the multi-link serial manipulator
$\mathbf{f}$	The positional component of the spatial force
$\boldsymbol{\omega}$	The rotational component of the spatial force

$\boldsymbol{\mathcal{F}}$	A 6DOF spatial force acting on the probabilistic multi-link serial manipulator
$\boldsymbol{f}$	The positional component of the probabilistic spatial force
$\boldsymbol{w}$	The rotational component of the probabilistic spatial force
P_{att}	The point the attractive force is applied
K_{att}	Coefficient of an attractive force amplitude
K_s	A constant for defining a transient state of the attractive force
K_{zero}	A small non-zero positive constant
K_P	Ellipsoid coverage constant
K_f	Coefficient of a repulsive force amplitude
K_{de}	The environmental repulsive force distance factor
K_{ds}	The self repulsive force distance factor
d_0	Distance between the closest points between two links
E_r	Force field error factor
$\hat{\mathbf{D}}$	The variance force field
$\hat{\mathbf{D}}_{min}$	The minimum force field
$\hat{\mathbf{D}}_{minV}$	The minimum variance force field
$\hat{\mathbf{D}}_{maxV}$	The maximum variance force field
$\mathbf{R_{ff}}$	A rotation matrix for a force field (3 by 3)
$\mathbf{r}$	A matrix describing the radius of a force field (3 by 1)
$\mathbf{c}$	A matrix describing the centre of a force field (3 by 1)
ξ_{min}, ξ_{max}	The minimum and maximum variance factors
$\mathbf{V}$	A matrix of a component's vertices
$\boldsymbol{\sigma}_{\mathbf{V}}$	A matrix of the variance of a component's vertices
$\mathbf{P}_{\boldsymbol{\sigma}_{\mathbf{V}}}$	A matrix of points representing the variance of a component's vertices
$\hat{\mathbf{V}}$	A component's vertices relative to a unit sphere
$\psi_{\hat{\mathbf{D}}}$	A matrix used to create a unit sphere
ξ_{dist}	Furthest distance to a vertex within the unit sphere
$\mathbf{r}_{min}$	The radius of the minimum force field

$\mathbf{r}_{minV}$	The radius of the minimum variance force field
$\mathbf{r}_{maxV}$	The radius of the maximum variance force field
$\mathbf{P}_{obs}$	The environment voxels
$\hat{\mathbf{P}}_{obs}$	The environmental voxels transformed relative to a unit sphere
$\hat{\mathbf{P}}_{dist}$	The distance from the origin of the unit sphere to the transformed environmental voxels
$\hat{r}_l$	The radius of the maximum variance unit sphere
d_{cls}	The distance penetrated into a force field
k_{sample}	Local minima detection sample period
k_q	Insignificant motion detection threshold
$k_{\dot{q}}$	Insignificant velocity detection threshold
$k_{\dot{q_{set}}}$	Repeated motion threshold
$\mathbf{N}_0$	The start node
$\mathbf{N}_g$	The goal node
$\mathbf{N}$, $\mathbf{N}_{CR}$	An intermediate node and a common region centre
u	Current nodes index
v	Parent node index
$\mathbf{P}_{mid}$	A discontinuous minpoint
$\mathbf{P}_{min}$, $\mathbf{P}_{max}$	The end point of the shorter and longer discontinuous pair
l_{min}, l_{max}	The shorter and longer lengths of a discontinuous pair
θ_{min}	The minimum angle
Z_{off}	Number of rays in the minimum angle
θ_{res}	The angle between rays
c	The radius of a robot's workspace
$\mathbf{L}$	A matrix of ray distances
δ	A ray's offset angle
$\boldsymbol{\theta}$	A matrix of angles between the each ray and the offset ray

m	Weighting criteria set
d_m	The distance for a weighting criteria set
K_m	Normalised distance for a weighting criteria set
α_m	The weighting factor for a weighting criteria set
w	The final weight for an intermediate node or common region centre

Glossary of Terms

Base	The foot pad of the robot that is currently fixed to the surface.
Common Region	Where visible regions of the end node and an intermediate node overlap. Termination criteria.
Common Region Centre	The midpoint of the common region.
End Node	Ending position of the inchworm robot end-effector.
Environment	A 3D structure in which a manipulator is positioned. Assumed to have some structural characteristics such as planar surfaces.
Ferromagnetic	Made of metals to which magnets are attracted.
Hand position uncertainty	The deviation in the end-effector position when attaching to the surface caused by the magnetic force.
Hybrid Planner	Path planner which comprises of two or more individual path planners working in tandem.
Inchworm Robot	A 7DOF serial robot which uses magnets to adhere to ferromagnetic surfaces.
Intermediate Nodes	Points within Cartesian space which serve as subgoals with the LoST algorithm.
Line of sight	The visibility of the surroundings from a point which is not impeded by obstacles.
Maximum index level	Maximum number of iterations down a single branch of the line of sight tree the line of sight algorithm searches.

Multi-link serial robot	A series of actuated joints and connected links describing a robot from a base to an end-effector.
Node	Possible robot position in Cartesian space.
Obstacle	An object which a manipulator can potentially collide with.
Planning	The act of generating a path (and motion) which the robot can then follow between two poses.
Pose	The joint configuration of a robot.
Ray Casting	Process which uses rays to determine intersection points with obstacles.
Start Node	The start position of the inchworm robot end-effector.
Start Pose	The start pose of the inchworm robot.
Structural uncertainty	The deformation caused by gravitational loads on the inchworm robot.
Surface Normal	A 3D vector perpendicular to a surface.
Visible Region	Area surrounding a node which is visible and determined through line of sight.
Voxel	Volumetric Pixel which represents a 3D cube-like volume in Cartesian space.
Waypoint	Either a common region centre or intermediate node used as a goal for the 3D-PF^2 algorithm to manoeuvre to.

Chapter 1

Introduction

Robots are becoming prevalent in many aspects of today's society. Where once people were required to perform dangerous or difficult tasks, robots are now used with both higher efficiency and lower ongoing costs. While it is sometimes difficult for a robot to mimic the required behaviour and mobility of a person in order to achieve these tasks, advancements in both hardware and software have allowed robots to be used in more practical applications. CROC, an inchworm robot, is one such robot [1]. With a 7 degrees of freedom (DOF) multi-link serial body providing means for dexterous motion and a permanent magnet adhesion system allowing adhesion to ferromagnetic surfaces, this robot is able to inspect steel structure environments which are particularly difficult for people to access. However there still exists challenges in the use of such a robot.

One such challenge is safe robot motion under uncertainty. In real world applications uncertainty can cause imperfect knowledge about the robot state which, among other concerns, prevents safe motion. The inchworm robot has been identified to suffer from two main sources of uncertainty: structural uncertainty and hand position uncertainty. The structural uncertainty is related to gravitational loads causing deformation in the robot body, while the hand position uncertainty arises due to the permanent magnet adhesion system causing inaccuracies in the landing position. Both prevent accurate knowledge about the inchworm robot state.

This thesis explores the effects these uncertainties have on the inchworm robot and develops models and algorithms used in an approach for planning with uncertainty for multi-link serial robots in 3D environments. A model representing the structural and hand position uncertainties is developed. This model is incorporated into a 3D Probabilistic Force Field (3D-PF^2) algorithm through force fields surrounding robot links. The 3D-PF^2 algorithm allows for smooth, collision-free path planning for multi-link serial robots. A Line of Sight Tree (LoST) algorithm is developed for longer time-horizon planning with the 3D-PF^2 algorithm used for local, short time-horizon motion planning. The LoST algorithm generates waypoints as goal locations for the 3D-P^2 algorithm to plan paths and perform collision avoidance.

This chapter introduces the research presented in this thesis. First the background of the Sydney Harbour Bridge application scenario is presented with the research issues detailed. The remainder of the section describes the thesis scope, the main contributions and provides the outline of the thesis.

1.1 Background

Many steel structures, such as the Sydney Harbour Bridge, require continuous maintenance and inspection to ensure longevity, structural integrity and aesthetics. Maintenance and inspection procedures are highly demanding, especially when the structure is used extensively by trains, cars and pedestrians, or located in destructive surroundings such as marine environments. Figure 1.1a shows an example steel structure that requires continuous inspection.

While the exterior surfaces of most structures can be reasonably accessed and inspected, albeit with bulky equipment or at extreme heights, the interior of some structures cannot. Confined space entry procedures and experienced personal are required to enter and inspect these areas [2]. Legislation requires that risk to personnel entering confined spaces must be reduced as much as is practicable. Hazardous chemicals can be present such as lead based paint or airborne contaminants which require purging techniques to remove. If these are present, accessing confined spaces can become a costly endeavour. Maintenance teams have multiple personnel with each individual required to perform different tasks such as inspection, maintenance, monitoring equipment and acting in the case of an emergency. Hazardous conditions due to heights, confined spaces or external environmental factors can compound the difficulty in performing maintenance and inspection procedures [2, 3].

Deploying a robot eliminates many issues associated with manual maintenance and inspection. By using a robot, human exposure to hazardous conditions is reduced, which also reduces the number of support personnel required. Additionally, the robot is able to inspect and navigate areas that a human would not be able to access without support equipment or personal protective equipment to protect them from dangerous elements such as excessive heat, fumes or contaminants. However, navigating in a small confined space presents a number of challenges for robots. Limited access prevents acquisition of a priori knowledge; confined spaces and small gaps limit the size of the robot; riveted surfaces and poor surface condition reduces the available surfaces for adhesion; and complex, internal obstructions limit the type of robot used. Especially difficult areas often occur within a tunnel structure; a partition plate with limited access through the top and bottom, and a partition plate with a manhole through the centre (Figures 1.1b and 1.1c respectively).

Many climbing robots exist which are capable of inspecting a wide variety of structures. CLIBO [4] uses claws to climb rough, vertical surfaces using motion techniques that mimic both rock climbers and cats. W-Climbot [5] optimises the surface placement of its suction

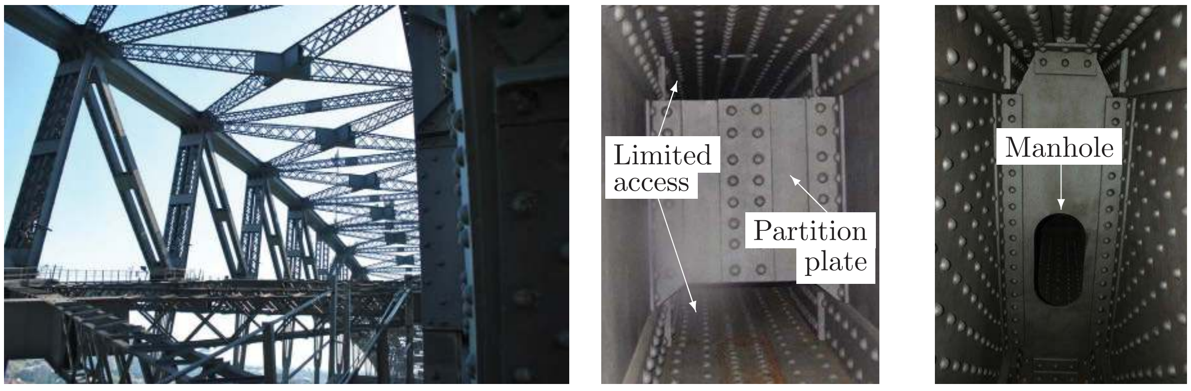


Figure 1.1: (a) A bridge scenario and related difficult areas; (b) a partition plate with limited space on the top and bottom for access and (c) a partition plate with a small manhole for access.

caps on climbing surfaces through pose selection techniques to improve the efficiency of the vacuum. Waalbot II [6] utilises micro-fibre foot pads to attach to a number of different types of surfaces. DuCTTv2 [7] is designed for exploration and maintenance of ducts. Robots have different adhesion methods, and use different control techniques, appropriate to their application and to overcome environmental constraints they may encounter [8]. These robots are not suitable for steel structures.

There are a number of robots capable of navigating steel structures. Wheeled [9], tracked [10] and pipe line [11] robots are available which would be able to traverse over a rough, discontinuous surface. However, they would not be able to traverse between acutely angled (greater than 90°) surfaces. Suction based inspection robots [12] may be suitable for exterior surface inspections, but would fail on interior inspections due to their innate bulkiness and the difficulty that arise when adhering to rough surfaces such as those covered with rivets or rust. A simple inchworm design [13] has been developed, however it is limited to a single plane of motion and requires relatively flat surfaces. Previous research [14–16] has been conducted on the effectiveness of climbing quadrupeds and arthropodal hexapods with three DOF per leg. Their superior climbing capabilities are due to their body segmentation, with each leg encapsulating varying workspace volumes and with significant compliance in their joints. Implementing this design into a robotic system for practical applications would be problematic. By introducing active joints between body segments or increasing the number of joints in legs the mechanical and control complexity and the overall weight of the system adversely increases.

Steel bridges and other complex steel structures lend themselves to being climbed by inchworm robots, as the many degrees of freedom of the body allow transitions between surfaces with high angular variation and the ability to navigation around smaller obstacles such as rivets [17]. A 5DOF robot, MATS [18], developed for the elderly and disabled to improve their quality of life demonstrated a level of manipulability which would be suitable for inspection. However, this system requires docking stations for motion which may not be possible to install on most structures limiting its effectiveness. A similar 5DOF climbing robot, W-Climbot [19], uses multi-sucker pads to adhere to surfaces. Special pads increase the reliability and safety of the suction and developed joints provide high torque to weight ratio. However, suction adhesion systems have high power requirements [20] and are not suited for rough, riveted surfaces. Some inchworm robots have been developed which are capable of climbing pole-like objects such as trusses [21] and branches [22] using grippers. This adhesion method would be ineffective in the mostly flat environment presented in the tunnel substructure.

An inchworm robot [1] (Figure 1.2) has been designed for the application scenario of inspection of a steel bridge (Figure 1.1) focusing on areas which are particularly difficult and hazardous for personnel to access. The inchworm robot provides a suitable platform for navigating obstacles and stepping between rivets due to the dexterity of the 7DOF multi-link serial body. Its permanent magnet adhesion system allows non-destructive adhesion to ferromagnetic surfaces to facilitate climbing and ensures surface adhesion will be maintained in the event of a power failure. Each permanent magnet is installed in its own individual housing which can be oriented to change the strength of the magnetic force acting towards the surface. The inchworm robot utilises a number of sensors: a RGB-D camera is used to capture point clouds of the environment; IMU sensors measure the orientation of the base and end-effector; and encoders provide feedback of the joint locations.

1.2 Motivation

Safe motion is a requirement for robotic systems with the selected algorithms heavily influenced by the robot's kinematics and properties, and the application environment.

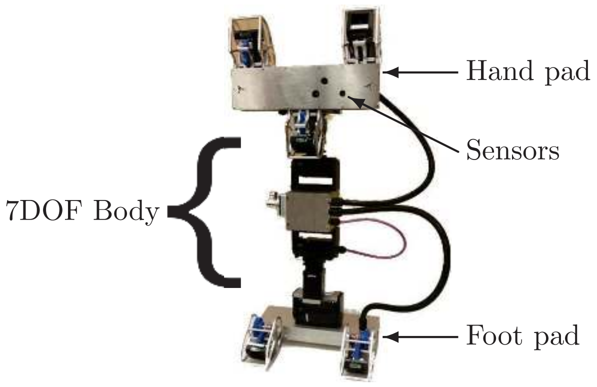


Figure 1.2: A 7DOF inchworm robot for steel bridge inspection.

The inchworm robot presents a number of challenges despite being identified as a suitable platform for traversing steel structures such as the Sydney Harbour Bridge.

The weight of the inchworm robot is a major concern. This is due to the weight of the permanent magnet adhesion system situated at either end of the inchworm causing the body to deform; this is considered as structural uncertainty (Figure 1.3). While increasing the rigidity of the structure may solve this problem it would increase the overall weight of the system. This would then require a stronger, larger and heavier adhesion system to facilitate surface adhesion. As the deformation cannot be eliminated it must be addressed in path planning. Various forms of deformation are considered in literature; however, the inchworm robot presents an interesting challenge. The deformation in the inchworm robot is highly dependent on its state and, as the base may be orientated in any direction, the number of possible states is near infinite. This prevents many existing methods of representing or planning with structural uncertainty from being used.

The strength of the permanent magnets in the adhesion system cause additional difficulties. As the magnets are deployed to facilitate surface attachment, the magnetic force acting towards the surface increases. Once the force becomes significantly high, the pad is forced to comply to the surface. Ideally the adhesion system attaches at the desired goal location; however, the magnetic force exerted by the permanent magnets may result in the pad deviating unexpectedly from the intended goal location with the body of the inchworm robot moving to comply with the motion. This deviation is considered the hand position

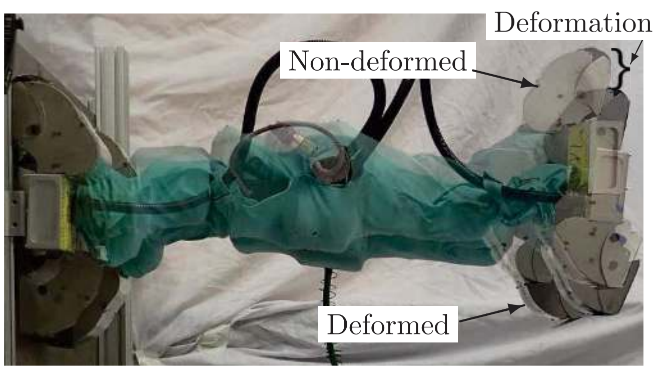
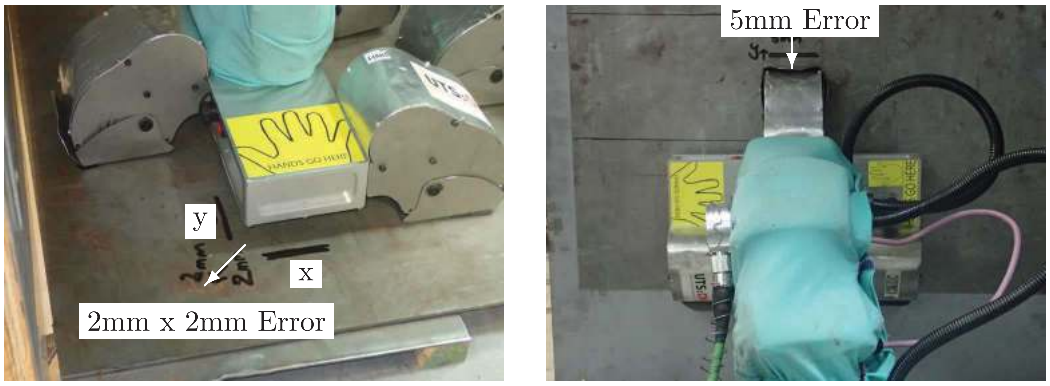


FIGURE 1.3: The inchworm robot deforming due to the weight of the permanent magnet adhesion system. The deformation of the inchworm robot is superimposed with an image of the robot in an upright position which is not affected by deformation. The variation between these two images shows the deformation. Covers are attached to the robot for protection.



(a) (b)

Figure 1.4: The hand position error due to the strength of the permanent magnets in the adhesion system (a) on the floor and (b) on the wall (pad detached from the surface for clarity). The arrows show the direction of the hand position error from the intended goal location.

uncertainty (Figure 1.4). This is a concern as collisions may occur either along the length of the robot body or at the pad. A path planner must be able to consider this uncertainty. Due to this unique adhesion method, this has not been considered in literature.

Collision avoidance for the inchworm robot is challenging; a challenge which is compounded by the adhesion system. The size of the adhesion system limits the size of gaps the inchworm robot can move through. This situation often occurs within the application scenario. When attempting to move through gaps, unless the end-effector is perfectly aligned, there is a high chance of collision. While this can be mitigated if the state is known accurately it has been established that this is not often the case.

The inchworm robot is required to navigate through 3D environments, specifically those identified within the application scenario (Figures 1.1b and 1.1c respectively). This is intrinsically a difficult task. Accurate pad placement is paramount for safety as difficult manoeuvres, such as moving through small gaps or stepping between rivets which are situated on many internal surfaces, are required. Surface attachments are required to transition between various surfaces with varying orientations. This is necessary for climbing and to avoid difficult to traverse areas, such as areas with a high concentration of rivets or heavily rusted areas which are unsafe to step on. Any implemented planner must then be able to plan a path for the end-effector to the goal location on the surface while simultaneously avoiding collisions.

1.3 Scope

This research aims to develop models and algorithms used in an approach for safe, collision-free path planning for a multi-link serial robot with structural and hand position uncertainty in 3D environments.

The inchworm robot suffers from two types of uncertainties - structural and hand position uncertainties. The effect of these two uncertainties are modelled. This model describes the uncertainty in the coordinate frame of reference of the joint's of a multi-link serial robot.

A 3D probabilistic force field (3D-PF^2) algorithm is developed which incorporates the modelled uncertainties and develops smooth, collision-free paths to goal locations for multi-link serial robots. Each link is surrounded by an ellipsoid which is sized to encompass the link and the uncertainty in the joints related to the link. Based on this ellipsoid, a force field is defined for motion planning and obstacle avoidance. Paths are developed through a combination of repulsive forces which push the robot away from obstructions which penetrate the force fields and an attractive force which pulls the end-effector to a given goal location.

The 3D-PF^2 algorithm is considered as a short time-horizon planner for generating paths to a goal location. The Line of Sight Tree (LoST) algorithm was developed as a longer time-horizon planner by finding waypoints around obstacles for the 3D-PF^2 algorithm to plan short time-horizon paths towards. The waypoints improve the performance of the 3D-PF^2 algorithm by helping to avoid local minima. The LoST algorithm loosely draws inspiration from how a person would perceive a scene whereby their gaze tends to fixate on important and information-rich regions, such as edges of objects.

The following assumptions have been made to bound the scope of this research. *a*) It is assumed that the environment in the immediately vicinity surrounding the robot is known; however, it is not necessarily assumed that the entire workspace is known. This is to ensure that some initial motion is possible. *b*) It is assumed that the orientations of surfaces have been identified and are readily available for surface attachment. If the surfaces have not been identified, the orientation can be readily calculable. *c*) Once the robot has attached itself to the surface, it is assumed that it is capable of supporting itself through the adhesion system.

1.4 Contribution

The contributions of the thesis are:

1. A model of structural and hand position uncertainty in multi-link serial robots.
2. A 3D-PF^2 algorithm - a short time-horizon path planning algorithm for developing safe, collision-free paths in 3D environments for a multi-link serial robot with uncertainty in its joint coordinate frame of reference.
3. A line of sight based path planning algorithm - the LoST algorithm - for longer time-horizon path planning for use with a multi-link serial robot path planner.

1.5 Publications

1. Pagano, D. and Liu, D. An Approach for Real-time Motion Planning of an Inchworm Robot in Complex Steel Bridge Environments. In *Robotica*, pages 1–30. Cambridge Univ Press, 2015
2. Ward, P., Paul, G., Quin, P., Pagano, D., Yang, C.-H., Liu, D., Waldron, K., Dissanayake, G., Brooks, P., Mann, P., Kaluarachchi, W., Manamperi, P., and Matkovic, L. Climbing Robot For Steel Bridge Inspection: Design Challenges. In *9th Austroads Bridge Conference*, pages 1–12, Sydney, New South Wales, 2014
3. Pagano, D., Liu, D., and Waldron, K. A Method for Optimal Design of an Inchworm Climbing Robot. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1293–1298, Guangzhou, China, December 2012

1.6 Thesis Outline

This thesis is structured such that the first two chapters provide an outline of the research and related work. Chapter 3 develops a model for multi-link serial robots with structural and hand position uncertainty. Chapter 4 incorporates the model into a 3D-PF^2 algorithm for short time-horizon path planning. Chapter 5 develops a LoST algorithm for longer time-horizon path planning for use with the 3D-PF^2 algorithm. Conclusions are presented in Chapter 6.

Chapter 2 Research describing methods for modelling the uncertainty related to the inchworm robot is presented. The research continues to explore both deterministic and probabilistic path planning algorithms which may be applied to a multi-link serial robot within the given application scenario.

Chapter 3 The effects of the structural and hand position uncertainties in the inchworm robot are analysed. From this analysis representative models of the uncertainty are created and applied to a multi-link serial robot.

Chapter 4 A 3D-PF^2 algorithm developed for short time-horizon path planning of a multi-link serial robot with uncertainty in the coordinate frame of reference of joints is described. A number of simulations and experiments are detailed to exhaustively test and evaluate the algorithm.

Chapter 5 A LoST algorithm for use with the 3D-PF^2 algorithm described in Chapter 4 is developed. The LoST algorithm is detailed both as a standalone path planning algorithm and as a longer time-horizon path planner for use with the 3D-PF^2 algorithm. Simulations are performed to evaluate the LoST algorithm detailing its effectiveness in known and unknown 2D and 3D environments with further simulations and experiments performed with the 3D-PF^2 algorithm.

Chapter 6 The research presented in this thesis is summarised. Analyses are made regarding the limitations of the presented algorithms. Conclusions are drawn in regards to the research and future work proposed.

Chapter 2

Review of Related Work

A challenge for the inchworm robot is motion planning with uncertainty. The design of the inchworm robot and its application environment raises a number of challenges. This chapter outlines the following research topics: the types of uncertainty present in robotic systems; modelling techniques and methods for representing the uncertainty in robotic systems; and path planning methods which consider modelled uncertainties.

2.1 Uncertainty Related to the Inchworm Robot

In many applications it can be assumed that the system state and models are fully known and accurate. However, if this information is incorrect or incomplete the outcome of any decision cannot be guaranteed. This is often the case in many real robotic applications as factors such as external forces can cause the state or models to deviate from their expected values. This is considered as uncertainty. In controlled environments, such as in a laboratory, measures can be put into place to mitigate any uncertainty although these measures may not be applicable in real application environments. Robotic systems generally have multiple sources of uncertainty. Uncertainty can be categorised as either control uncertainty, sensor uncertainty or imperfect map data [24]. The following details some of the types of uncertainty present in robots.

2.1.1 Control Uncertainty

Control uncertainty occurs when the response of the robot cannot be precisely predicted or there is a physical variation in the expected state of the robot.

Error in the response of a robot can take many different forms. Noise in actuators may impair accurate responses from control actions in robots such as in nonholonomic differential-drive robots [25, 26] or manipulators [26]. Vibration in the robot structure during motion can prevent accurate paths from being followed [27–29]. External forces acting on robots prevent accurate motion such as in steerable needles moving through tissue [30] or water currents acting on underwater robots [31, 32].

Uncertainty in the physical state of a robot may occur for a number of reasons. Deflection in the robot body may occur due to loads acting on the robot [28, 33, 34] or unknown loads on the end-effector [35]. The physical state may differ from the expected state due to manufacturing errors [36], construction errors [37, 38] or general wear affecting position accuracy [39]. Also the kinematic or dynamic properties of the robot itself may prevent accurate knowledge of its state, such as with the highly elastic links used in elastic robots [27, 40]; the elastic legs used for location of the STARBot robot [41]; or a pneumatically controlled soft robot [42].

2.1.2 Sensor Uncertainty

Sensors include any piece of equipment used to gather information about the physical state of the system. Sensor uncertainty can occur from inaccurate data which is often induced from internal sensor noise. Internal sensor noise affects many different types of sensors such as range sensors [25, 43, 44], cameras [26, 45, 46], encoders [47], inertial measurement units (IMU) [46, 48, 49] and beacons for localisation [26]. In some cases the sensor uncertainty may be due to a sensor not being suitable for the task [50] or simply uncalibrated [51].

2.1.3 Imperfect Map Data

Imperfect map data occurs when the stored environmental information is incorrect or incomplete. This may arise from unknown or partially unknown maps [51–56], dynamic obstacles [57–61], or uncertain map data which may be as a result of variation or inconsistency between sensor scans [25, 45, 62]. In some cases it is assumed that the initial map data is uncertain [63, 64].

2.2 Modelling Uncertainty

2.2.1 Deterministic Modelling

In some cases the model of a robot's state with uncertainty can be represented deterministically. The following section describes literature related to deterministic methods of modelling a robot's state.

2.2.1.1 Modelling of Manipulators

In many cases manipulators have errors in their kinematic model caused during construction or due to external factors during operation. The inchworm robot has a similar kinematic structure to serial manipulators.

Modelling techniques are used to account for deflection and variation in the kinematic parameters of manipulators. Early work by Wu [65] developed error models to represent the differential error in DH parameters between coordinate frames of a manipulator. Backer and Bolmsj [34] developed a deflection model for a 6DOF welding manipulator acting under high loads assuming rigid links and flexible joints. While successful, the models rely on force sensors embedded in the robot for calibration and to predict only the deflection in the end-effector. Schneider et al. [38] presented a stiffness model for deformation compensation in a machining manipulator. These techniques are often implemented when the error in the deflection is small, generally only a few millimetres. Additionally, their use primarily in highly controlled environments, such as CNC workshops, allow for greater control of unknown factors such as precise base locations. Li et al. [35] used knowledge of joint angles to estimate the payload on an end-effector and adapt the controller to account for variances. Phong et al. [66] used an estimation algorithm to predict the external force on an end-effector based on joint torque sensors. Hafezipour and Khodaygan [36] considered uncertainty in joints and links due to a number of factors including manufacturing and sensor errors. Through sensitivity analysis, those variables which most greatly affect the end-effector location are identified. This allows tolerances in the design to be tighter around more influential variables and less stringent around less influential ones.

Sensor based techniques are used to determine the position of a robot through sensor data relative to a reference frame in the environment. Visual servoing is one such technique which corrects position errors in the system state through cameras. Zergeroglu et al. [67, 68] used both fixed camera and camera-in-hand approaches to develop time-varying, end-effector position trajectories and error models; however, their approach is limited to manipulators limited to a 2D plane. Tao et al. [69] compared point clouds of work objects, determined through sensor data, to a registry of CAD models of work objects. The relative variation between the sensor frame and work object frame is determined and corrected for. This is not suitable for the inchworm robot as the work objects within the environment are either non-existent or unknown.

Optimisation techniques can provide locally or globally optimal solutions to mechanical uncertainty [70, 71]. Lightcap et al. [33] presented a 30 parameter optimisation algorithm, 4 DH parameters and 2 stiffness parameters per link, to determine the flexibility and geometric parameters for a 6DOF robot using measurements from a coordinate measuring machine. In this application, the system only suffers from small deflections (approximately 0.03mm) which may not be applicable to links with higher elasticity. Aghili [72] developed an optimisation problem to determine base and geometric uncertainties in a closed loop formed by two manipulators sharing a load. While the routine does minimise the error in the system, the developed algorithms are specific as they only consider the relative frames between each base and not a global coordinate frame. Zhang et al. [37] used an optimisation algorithm to determine a manipulator's stiffness matrix and geometric errors due to manufacturing using the assumption that the manipulator is rigid. Through observability analysis Joubair et al. [73] selected the best poses which will allow the error in the model to be identified through a least-square minimisation optimisation routine. For a 6DOF robot, 336 poses were used. For the inchworm robot with a high number of DOFs, this optimisation routine would become computationally intractable.

2.2.1.2 Modelling of Elastic Robots

Elastic robots are becoming more popular due to the reduced risk of injuring humans and the robot itself in the event of collisions [27]. They generally are lighter, have higher payload-to-weight ratios and require less energy compared to rigid robots. However accurate control can be difficult to achieve due to oscillations or gravitational loading [28, 74]. Elastic robots under loads are similar in geometry to the inchworm robot with structural deformation.

Methods for parametrically modelling elastic robots are prolific. Many solutions focused on control and minimising vibration through dynamics and dynamic equations in single DOF [75–80], 2DOF [29, 81] and 3DOF [27, 28] robots. Heidari et al. [40] considered the maximum payload for a wheeled robot mounted with a flexible manipulator. The manipulator's links are considered as elastic beams with related strain models implemented. Mayyas [41] presented small-scaled bioinspired legged robots - STARBot. Three and four legged varieties are presented with each modelled as a flexible single DOF fixed cantilever. While these techniques model the given robots, extending the approaches to higher dimensionality is challenging. Jiang et al. [82] modelled a flexible 7DOF manipulator as a series of lumped masses and massless springs. Using an end-effector mounted camera, visual information is used to control vibrations and can be applied in lieu of traditional techniques which rely on strain gauges and accelerometers. Unfortunately, this approach disregards the deflection due to gravity. Abiri et al. [83] introduced minirobots modules which utilise shape memory alloys for actuation and when placed in series act like an elastic robot. Through analysis of the energy in the system and the physical properties of the components, equations for the relative orientation between the origin and end-effector of individual models are developed. Due to their unique design, these models developed for the minirobots modules are not applicable to the inchworm robot.

Finite element analysis (FEA) is a technique for estimating a model by dissecting the model and analysing its parts. In elastic robots FEA has provided some utility. Reddy and Jacob [84] used FEA to model the dynamics of a weighted-end, flexible, single-link manipulator. The system is analysed to determine equations for the kinetic and potential energies, internal structural damping of the link and the joint friction. This approach, like most FEA techniques, defines equations for improving the overall accuracy of the models which, for the inchworm robot, would be difficult to determine due to the high DOF and large number of possible base locations. Farid and Cleghorn [85] used a curvature-based FEA to model the dynamics of planar flexible manipulators with an n number of links. This method established equations which are used to determine both an arbitrary point on a given link through geometry and the transverse deflection of a link by analysing the curvature of a beam element. Significant work would be required to extend the method for non-planar manipulators as it cannot be assumed that the curvature of the beam element is consistent in 3D and is unaffected by the robot's orientation.

2.2.1.3 Modelling of Continuum and Soft Robots

Continuum robots are hyper redundant manipulators. Due to their high number of joints continuum robots mimic snakes and elephant trunks allowing for a high level of flexibility. Their models tend to behave in a similar manner to the inchworm robot under the influence of structural deformation.

Renda and Laschi [86] developed a general steady state mathematical model for modelling a continuum robot as a Cosserat beam. A matrix of coordinate frames is maintained which details the strain between consecutive sections as a geometric transformation. Khoshnam et al. [87] modelled a steerable catheter as a beam with large deflection considering deflection along its length due to forces exerted at the tip. These models cannot be directly applied to the inchworm robot due to their method of actuation. Yang et al. [88], Du et al. [89] represented a prototype continuum robot as a manipulator with n number of joints allowing DH parameters to be used for modelling. While the prototype is geometrically similar to the inchworm robot with structural deformation, the design of the prototype which allows the kinematic model to be expressed with n number of joints cannot be applied to the inchworm robot. Dehghani and Moosavian [90] modelled the dynamics of a planar continuum robot using FEA. This approach segments the robot into a series of circular arcs and geometrically determines the kinematics for the robot by considering each segment's origin location and orientation. As this approach assumes continuous, controllable bending in the robot's link, it is not applicable to the inchworm robot.

Shapiro et al. [91] used a piezoelectric transducer polyvinylidene fluoride (PVDF) strip to track the shape of a planar continuum-like robot. Stresses are measured along the length which correlate to deflection angles. While indicative of the deflection, PVDF strips would be unsuited in their current form to determine the deflections expected in the inchworm robot. Li et al. [92] used tension wires to control a geometrically static model of a continuum robot. Despite the curvature of the system mimicking the effect of structural deformation on the inchworm robot, due to the continuum robot's tension wire design the developed model cannot be considered.

Soft robots have soft outer coverings which may reduce the impact of collisions. However, in some robots their internal structure is not completely solid either. Classical control methods cannot necessarily be applied to these robots due to their hardware design while their high dimensional state spaces and "softness" make learning their models infeasible [42]. Katzschmann et al. [93], Marchese et al. [94] assumed constant curvature to model the kinematics of soft robots; this assumption is not applicable to the inchworm robot as the curvature in a link may vary depending on its pose. Queisser et al. [42] presented a soft robot which utilises pneumatics to control three body segments. The robot is modelled considering the pressure in and length of each segment. As pneumatic actuation has a long delay, a machine learning approach is used to reduce these issues. Due to the specific nature of the robot, this method cannot be used to model the structural deformation in the inchworm robot.

2.2.1.4 Modelling of Walking Robots

Walking robots, including bipedal, quadrupedal and hexapod robots, control sets of legs for locomotion. Locomotion is performed using various gaits. Modelling the effects of the surface contact may be related to the inchworm robot with its pads in contact with the steel surface.

Walking robots generally maintain control of their centre of gravity (COG) to ensure stability. Stabilisation strategies exist which consider uncertainty in modelling. Chung et al. [95] and Li et al. [96] considered uncertainty in trajectory and force tracking during each gait step. Wang et al. [97] added white noise to the gait state equation. Wang et al. [98] used fuzzy logic to compensate for uncertainty for a gait derived by a machine learning algorithm. Degrave et al. [99] used particle swarm optimisation to learn walking gaits in different environments. Walking robots are highly concerned with maintaining stability.

Other strategies model the uncertainty as a function of the distance each foot travels. Zheng et al. [100] controlled the distance travelled by a foot along the global z-axis based on the type of terrain being traversed. Kim et al. [101] corrected the position error at the COG by adapting the stepping position with further research performed to deal with highly pitched surfaces [102]. Zhu et al. [103] implemented a fuzzy control algorithm to adapt parameters of a foot controller for different environments. Liu et al. [104] used state vector machines to learn foot landing positions. Position control for walking robots generally are concerned with maintaining stability based on control of the COG. COG control is not necessary for the inchworm robot to maintain stability; these techniques are not suitable.

2.2.1.5 Modelling of Mobile Wheeled Robots and Mobile Manipulators

Uncertainty in the base location of mobile wheeled robots is widely considered with the state modelled as the x and y coordinates on a 2D plane and orientation with respect to a global reference frame. Generally the uncertainty in the base location is due to the noise in the encoder feedback or the wheels themselves. Mobile manipulators are mobile wheeled robots mounted with manipulators. At a single base location, a mobile manipulator can be modelled similarly to the inchworm robot.

Deterministic methods are used to correct the base state of mobile wheeled robots; examples are fuzzy logic based sliding model control [105], adaptive neural networks [106], and feedback linearisation [107]. Trajectory tracking algorithms are used for mobile manipulators which not only consider the base uncertainty but also that of the manipulator with feedback controllers [108], fuzzy controllers [109], fuzzy neural networks [110, 111] and adaptive controllers [112–115] all used. However, as these techniques are centred around trajectory tracking using the nonholonomic characteristic of mobile robots they are not suited to the inchworm robot.

Other algorithms exist for determining the system state of mobile wheeled robots. Morales et al. [116] experimentally estimated the COG for a mobile manipulator to prevent the robot from tipping over while moving on inclines. A virtual force field (VFF) algorithm was introduced by Wen et al. [117] which used a neural network to compensate for uncertainties in the dynamics of a wheeled robot. While learning to compensate for a small number of variables is possible, this may be intractable due to the high dimensionality of the inchworm robot's state.

Deterministic models exist which utilise sensors to reduce the uncertainty in the system state. Lee and Lee [118] used radio-frequency identification (RFID) to localise a mobile manipulator's base using a mounted camera and RFID tags placed throughout the environment. Hamner et al. [119] used a combination of visual and force servoing to increase the performance of an assembly mobile manipulator. Hvilshoej et al. [120] used two varieties of calibration techniques - high precision and high speed - to reduce the uncertainty in a mobile manipulator in an industrial environment. The high precision technique makes use of a hand-eye camera to take scans of the target position from multiple preprogrammed poses, while the high speed technique makes use of image processing and laser triangulation. Cheng et al. [121] utilised a RGB-D camera to perform visual servoing to correct a mobile manipulator relative to a distinguished target in the environment. While these techniques do reduce the uncertainty in the state, augmenting the environment is not possible with our application scenario. Igawa et al. [122] presented a base position detection algorithm for a vineyard weeding mobile manipulator which uses a stereo camera system for visual sensing and a tactile sensing system for confirmation. This technique may be adaptable to the inchworm robot after a step has occurred, however no corrective action could be performed prior to surface attachment which would be undesirable.

2.2.1.6 Summary of Deterministic Modelling

Mathematically modelling the system through analysing the geometric, kinematics or energy conservation laws, can provide an accurate representation of robots. While this provides a precise representation of the state, it is often reliant on assumption, conditions or the application. The structure of the inchworm robot's body is inconsistent as each link may have different cross-sectional areas along the length. Additionally, reducing the model to a series of equations and parameters which explicitly describe the inchworm robot state would be challenging due to the high dimensionality of the state, especially with the near infinite number of base states.

Optimisation algorithms and finite element analysis normally require accurate models. The dimensionality of the state of the inchworm robot prevents accurately defining a model with structural and hand position uncertainties.

Machine learning can be used to learn unknown parameters of a state representation. This does not necessarily require an accurate model of a system, albeit with more information a machine learning algorithm will converge towards a more accurate solution. Systems with high dimensionality, such as the inchworm robot, will take a longer time to learn and the resultant state may not be applicable across all possible states.

Sensors have extensively proven to be useful to detect uncertainties in a system. As sensors detect the current state of the environment, they can be used in lieu of accurate system models. Some trajectory tracking and visual servoing algorithms rely on augmenting the environment with targets which are tracked using on-board sensors or tracking the end-effector based on external sensors.While some sensors may be used to directly measure the effects of the structural deformation, it is still a challenge to measure the uncertainty in all possible states of the inchworm robot. Furthermore, installing additional sensors on the inchworm robot may inhibit its dexterous workspace preventing poses which are required to traverse the environment.

For these reasons a deterministic modelling approach is unsuitable to model the uncertainty in the inchworm robot.

2.2.2 Probabilistic Modelling

Probabilistic modelling techniques may be employed. A probability distribution is able to represent different robot states. This section looks at probabilistic modelling techniques which may be used to develop representative models of the inchworm robot's state.

2.2.2.1 Gaussian Distributions

One of the most common techniques for modelling a state with uncertainty is through representing the uncertainty as a Gaussian distribution. A Gaussian distribution describes the uncertainty using a bell curve and is parameterised by a mean and a variance. The mean indicates the most likely location of the uncertainty while the variance indicates the spread of values.

Uncertainty is often considered by injecting Gaussian noise into the transitional models of wheeled robots [123–125] and walking robots [46]. When the transitional model is invoked, the error will grow according to a Gaussian distribution. However, injecting Gaussian noise during each transitional step does not accurately describe the effects of the uncertainty in the inchworm robot. While the uncertainty may differ at consecutive time steps it does not grow. Instead the same state at non-consecutive time steps will result in roughly the same level of uncertainty.

Gaussian distributions can be used to describe the variance in the system state or in certain parameters. Komendera and Correll [126] determined the errors due to thermal expansion and structural deformation in truss structures during construction using an Extended Kalman Filter (EKF) based approach. Rucker and Webster [127] used an EKF to predict external forces acting on a continuum robot with Gaussian distributions used to describe torque uncertainty. Bloesch et al. [48] considered a Gaussian distribution to describe the uncertainty in the torque for a legged robot. Dogar et al. [47] and Pilania and Gupta [128] used Gaussian distributions to describe both the joint and base locations for mobile manipulators.

When multiple states must be modelled, it may be prudent to use a multivariate Gaussian [129]. This is generally parameterised using a mean vector and a covariance matrix. The covariance matrix allows the variance between different state variables to be described. Multivariate Gaussians are often used in localisation and SLAM algorithms [44, 130–132].

2.2.2.2 Uniform Representations

A uniform distribution is used when the probability of a state being at any point of a distribution is equally as probable as it being at any other point along the same distribution. The use of uniform distributions is less prolific than Gaussian distributions and are often suitable when the range of the uncertainty is not biased.

Melchior and Simmons [133] used a uniform distribution to describe the unknown coefficient of friction for a rover operating over rough terrain. Bai et al. [134] considered a uniform distribution for learning parameters in two different cases; a two-link articulated robot which resembles a swinging acrobat and pedestrian avoidance for an autonomous vehicle. The uniform distribution was used to represent the mass of an acrobat and the intention of the pedestrian.

2.2.2.3 Bounded Uncertainty

In some instances a state can simply be represented as being within given bounds. While this representation does not provide a distribution of the probability of the state along the bound, the true state will lay within the bounds [135]. By maintaining only the boundary conditions often the computational costs are reduced [64].

Guibas et al. [64] used uncertainty bounds to represent the position of a robot in a bounded uncertainty roadmap algorithm. Paths are weighted based on the chances of collision occurring within the bound with bounds re-evaluated over the course of planning. Dehghani and Moosavian [90] modelled the uncertainty in a continuum robot as a bound determined from a factor of the module of elasticity. Trajectory tracking algorithms may consider the uncertainty in the system state as bounded [136]. Tripathy et al. [137] considered an unknown payload with a bounded uncertainty. Nazari and Notash [135] used bounded intervals to represent manufacturing errors, round-off errors, and measurement and control errors. Nazari and Notash continued using interval analysis to perform motion analysis on various DOF manipulators.

A set-membership approach is used to allow the uncertainty in the system to be considered in a bounded-error context which reduces the system to a set of non-linear equations and inequalities and reduces the overall complexity of the system [138]. Jaulin [139, 140] used this approach to perform SLAM for an underwater robot and a pose-based SLAM algorithm for a wheeled robot. Yu et al. [141] extended the set-membership approach to use ellipsoids to define the bounded uncertainty. This allows the approach to be used in large scale, online problems.

2.2.2.4 Sample Based Models

In some cases the uncertainty is modelled from samples gathered through trials or experiments. Using real-life data allows a better representation of the uncertainty; however, this might not necessarily be attainable for all system states especially those with higher dimensionality. Roveda et al. [142, 143] determined the uncertainty in a mobile manipulator from analysis of measurement inaccuracies. Karydis et al. [144] developed a method which extends deterministic models to stochastic models which include uncertainty due to the robot interacting with the environment.

2.2.2.5 Summary of Probabilistic Modelling

Probabilistic representations can be used to model a wide variety of uncertainties and are often used to account for multiple sources of uncertainty. These representations are used in many different facets of robotics including sensor and transitional models, SLAM and path planning.

From the literature it is apparent that a probabilistic distribution is suitable for modelling the uncertainty in the inchworm robot. Sample based models would be difficult to develop due to the high dimensionality of the state. While bounded uncertainty and uniform representations may be suitable, Gaussian distributions provide a better representation of the uncertainty distribution. The hand pad is more likely to land closer to the desired surface location. A Gaussian distribution would most likely account for variables about this state.

2.3 Path Planning

2.3.1 Non-Probabilistic Path Planners

A non-probabilistic path planning algorithm provides safe motion for a robot from a start location to a goal location when the state is known and predictable. While these path planners do not consider uncertainty in their design, they may be adapted to do so.

2.3.1.1 Potential Field Methods

Potential Field (PF) based planning algorithms [145] have been developed for years. Using a potential function to determine the gradient throughout an environment, paths are generated from a high potential position at the start point towards a low potential position at the goal point. Obstacles are considered as high potential areas causing the path to tend away from them. A variation of the PF was developed called the virtual force field algorithm (VFF) [146]. Instead of evaluating the entire map as PF algorithms do, VFF algorithms use sensor data to determine repulsive force based on the distance to obstacles surrounding the robot. However, basic PF and VFF planners may encounter regions known as local minima. This occurs when the sum of repulsive forces oppose the attractive force resulting in a zero sum and stalling further motion [147]. Unless avoidance techniques are implemented it is difficult for robots to escape local minima [148].

Literature regarding PF and VFF algorithms is prolific with implementations often considering local minima handling techniques. Wang et al. [149] used sensor scans to determine subgoals throughout the environment. Olunloyo and Ayomoh [150] used virtual obstacles to block potential local minima areas caused by concave shaped obstacles. Li et al. [148] varied the direction of the attractive force when entering local minima allowing the robot to escape. Mujahed et al. [151] used a variation of follow the wall algorithm to move towards the closest gap once an obstacle had been reached.

Other solutions are less concerned with local minima. Ghazal et al. [152] used a PF algorithm in a feature based environment for a manipulator; as the robot approaches obstacles the repulsive potential exponentially increases. If the environment cannot be represented with features this algorithm cannot be used. Hargas et al. [153] developed a PF algorithm for mobile manipulators. The potential function is applied offline to the environment from which trajectories for the base and each joint are found. Huptych and Röck [154] modified generated paths based on repulsive forces generated by dynamic obstacles. As the distance between an obstacle and generated path segments decreases, the repulsive force increases pushing the path segment away. Flacco et al. [61] introduced the concept of depth space for controlling a manipulator in dynamic environments. An external depth sensor measures the distances between known dynamic obstacles and the manipulator. This information is projected in $2\,{}^1/_2$ dimensional depth space allowing repulsive forces to be quickly generated using simplified distance calculations and pivoting the robot away from obstacles faster than it could otherwise. Chotiprayanakul et al. [155] used a 3D force field (3D-F^2) algorithm, a variation of the VFF algorithm, in conjunction with a haptic device and a Human-Machine-Interface to teleoperate a manipulator. The haptic device allows the operator to control the attractive force acting on the end-effector while repulsive forces are applied when an obstruction enters force fields surrounding the robot body. As this implementation takes advantage of a human operator, no local minima avoidance technique is implemented. These algorithms could be successfully used provided that a sufficient amount of environmental knowledge was available and it could be adapted to use a probabilistic model. Rather than directly considering distances, Ouyang and Zhang [156] used an offline, velocity-based variation to the VFF algorithm for avoidance of constant velocity, known dynamic obstacles. This algorithm allows for a minimum distance between the robot and obstacles to be maintained which improves the time-efficiency of generated paths. However, as the application scenario is considered static, dynamic obstacle avoidance is of limited use.

2.3.1.2 Rapidly-exploring Random Tree

Rapidly-exploring random tree (RRT) algorithms offer a method for effectively planning in known environments. From an initial position, a tree is grown by attempting to connect randomly generated states to an existing portion of the tree; if there is no collision, a connection is made. This continues until the end position is connected to the tree. RRT algorithms may have difficulties finding paths through narrow gaps in the environment. Clifton et al. [157] developed a method to overcome this by establishing a new tree at generated states which cannot be connected to an existing tree. This variation has improved convergence rates over the standard RRT algorithm. Ferguson et al. [58] developed an online RRT algorithm capable of replanning sections of the current tree based on new sensor information or obstacles. Invalid sections are removed with the affected branches reassessed and regrown until a solution is found. Karaman and Frazzoli [158] presented an RRT* algorithm which finds asymptotically optimal solutions by finding optimal paths to every state in the problem domain. As this can be expensive in high dimensional state spaces, Gammell et al. [159] used an Informed RRT* algorithm to limit the search space once an initial solution has been found within a prolate hyperspheroid encompassing the initial and goal states. By focusing on this limited search space, the Informed RRT* algorithm approaches optimal solutions sooner in comparison to regular RRT* algorithms.

RRT algorithms are successfully used to generate paths for higher dimensional models. Park et al. [160] considered an RRT algorithm for control of a camera mounted on a manipulator during surgery. Benevides and Grassi [161] used a bi-directional RRT algorithm to generate paths for a free-floating manipulator in space. Modelling of such a manipulator is complex; however, Benevides and Grassi simplified the dynamics by considering the base as a passive spherical joint. Zhu et al. [21] developed a bi-directional RRT algorithm for a biped pole climbing robot in a spatial truss environment. Trajectories are planned between hand holds positioned on trusses and are described by parametric equations. This approach has limited application with the inchworm robot as collision detection is purely a function of the distance between the robot and poles.

2.3.1.3 Geometric Methods

Geometric methods develop paths by geometrically de-constructing an environment. These paths are usually global paths, with secondary local planners required to generate robot motion. Lozano-Pérez and Wesley [162] introduced visibility graphs which place nodes at obstacle points and edges between unobstructed nodes. Rashid et al. [163] built graphs based on the visibility between start and end points, and a low level trajectory planner for mobile robots. If the visibility is hindered, nodes are created around the obstacle with edges introduced. This process is repeated until the end point is found with the path determined by the shortest route. Voronoi diagrams based planners partition the environment by placing nodes equidistant from obstacles with edges connecting unobstructed nodes. Garrido et al. [164, 165], Fedorenko and Gurenko [166] used Voronoi diagrams to plan global paths for local planners. As Voronoi diagrams maximise the distance from obstacles the generated paths allow greater leeway with regards to obstacle avoidance. Kalra et al. [167] used a Dynamic Brushfire algorithm to update Voronoi diagrams in real time based on new sensor data.

2.3.1.4 Optimisation Based Path Planning

In well-defined systems optimisation techniques provide a unique method for path planning. They work on the principle of minimising an objective function while conforming to equations which describe equalities or inequalities.

Menasri et al. [168] developed a trajectory planning approach for redundant manipulators using a bilevel optimisation. The first level optimisation seeks to minimise the distance between the end-effector position and a goal position while maximising the distance from obstacles. The second level finds a configuration which corresponds to the end-effector position found in the first level. Schuetz et al. [169] improved end-effector paths generated online for a 9DOF manipulator using an offline optimisation method. As online paths tended to result in undesirable joint velocities, the optimisation method sought to minimises the dynamic costs while maintaining the joint constraints and avoiding collisions. Chen et al. [170] utilised an offline optimisation path planning method for a 10

DOF manipulator used to paint long ducts. The optimisation sought to minimise the distance between the position of special points on the robot and the duct's centre axis while ensuring no singularities occur and the velocity and acceleration of each joint are within allowable limits. Marchese et al. [94] found optimal trajectories using a model for the dynamics of an elastomer manipulator. Driven by high-capacity fluidic drive cylinders, the dynamics in the system were modelled through energy equations while the kinematics were described using bending curvatures. An optimisation algorithm iteratively adjusts the robot's end-effector towards the end goal whilst minimising the energy in the system. HashemZadeh et al. [171] presented a sliding mode controller for a robot manipulator using an online variant of particle swarm optimisation. This approach augments offline estimates of uncertain parameters with sensor data to optimise the control signal.

While these techniques provide accurate solutions they are reliant on well-defined systems. The inchworm robot's state is not well-defined especially with the uncertainties.

2.3.1.5 Hybrid Path Planners

Hybrid planning algorithms often provide additional benefits in comparison to conventional planning algorithms. They can provide a greater chance of finding a successful path or overcome specific problems such as local minima.

Tanzmeister et al. [172] combined an RRT algorithm with an A* algorithm to develop a planning algorithm capable of finding multiple poses for given requirements. The A* algorithm finds an initial path to satisfy the given requirements then an RRT algorithm further explores the space finding additional possible solutions. Qureshi et al. [173] improved the efficiency of an RRT* algorithm by using a potential function to connect nodes within a known environment. Results showed a reduction in the number of iterations and in the time required to compute optimal solutions in comparison with RRT* algorithms. Both hybrid algorithms use kinematic models.

Haddadin et al. [174] developed a hybrid circular fields and PF algorithm for use in dynamic and partially unknown environments. Circular fields are used in lieu of the traditional methods of generating repulsive forces for the PF algorithm and are generated around an obstacle by using its centre of mass. Robots move around the obstacle in the direction of the circular field allowing local minima to be escaped. This method requires the dimensions to be known a priori in order to determining the centre of mass of an obstacle. Jaradat et al. [175] created a PF algorithm in conjunction with a fuzzy logic controller to navigate a robot through a 2D environment with dynamic obstacles and targets. Two fuzzy logic subsystems are used to reduce the force models and subsequently simplify the required computing power.

Wen et al. [117] introduced a VFF algorithm which uses a neural network to compensate for uncertainties in the dynamics of a wheeled robot. This is extended to create a neural network based fuzzy algorithm to perform self-tuning for variables, such as the force field size, and to further reduce any uncertainties. Nourani-Vatani et al. [57] used a multi-level software architecture to utilise different path planning algorithms for an autonomous mower in dynamic, real-time environments. At the highest level a spiral planner is used to develop a basic global path. The local motion planner employs an RRT algorithm, which takes into consideration vehicle kinematics, to modify the generated path when obstacles are present or a way point is infeasible. A VFF algorithm is implemented at a lower level to avoid dynamic obstacles and correct any errors in the estimated robot pose. Nia et al. [176] combined a VFF algorithm with a fuzzy logic controller and boundary-follow algorithm on a simulated, circular mobile robot. In normal operation the VFF algorithm supplies information, such as obstacle positions and direction to the target, for the fuzzy logic controller to modulate the velocity of the robot. If the robot enters a local minimum, the boundary-follow algorithm is invoked and supplies information to the fuzzy logic controller instead. The boundary-follow algorithm decides to follow either the left or right wall at a specific distance until the robot escapes the local minimum. Ranjbar et al. [177] used a combination of grids, potential fields and fuzzy logic to determine paths for manipulators. An occupancy grid is used to evaluate the distance from obstacles with a potential field algorithm used to find a global path for the end-effector to travel. A fuzzy controller considers the kinematics of the robot and determines the repulsive forces.

2.3.2 Planning with Uncertainty

Probabilistic path planning involves planning a path for a robot from a start location to a goal location when the system models are either unknown or partially known. These planners account for uncertainty in their design with different algorithms accounting for different types of uncertainty.

2.3.2.1 Potential Field Methods

Potential field methods generally consider uncertainty in the system through varying the effect of the generated forces. In a tactile mapping routine Bierbaum et al. [178] adjusted the attractive potential to guide a robot towards unknown space. Uncertainty due to environmental changes are considered through the repulsive forces. Guan-chen et al. [179] include a virtual disturbance force to represent uncertainties in the environment allowing high changes of the potential field function. Gillham and Howells [180] simply use a repulsive force to react to unknown dynamic obstacles. Potential field methods provide high utility with environmental uncertainty, especially VFF algorithms which readily consider dynamic obstacles.

In some research, potential field methods are used to consider uncertainty in the system. Wang et al. [181] employ a repulsive force to move a manipulator away from potential singularity poses. Owan et al. [182] used potential fields to generate repulsive forces for HMI control with a haptic device. This method considers uncertainty through both a priori knowledge and the probability of failure to increase the level of control an operator is given. The repulsive force affects the haptic interaction force applied to the operator by increasing the stiffness relative to the distance to obstacles. The maximum stiffness applied increases as the level of control an operator is given increases.

2.3.2.2 Sampling Based Methods

Sampling based methods involve approximating the state to develop nodes with edges connecting them. Roadmaps present a method for decomposing an environment to provide global paths; however, they generally require secondary local planning methods. RRT algorithms on the other hand can provide local trajectories although they take time to explore all states.

Probabilistic roadmaps (PRM) are often used for planning with uncertainty. Huang and Gupta [183] presents a PRM algorithm which considers uncertainty in a mobile manipulator's base. In this algorithm particles were used to represent base locations at each node. Collision checks are performed by sweeping the manipulator along the generated edges. A weighting is provided based on the number of collision-free edges. The shortest path above a threshold is then found through the graph. Agha-mohammadi et al. [184] presented a belief-space variant of the PRM algorithm for motion and sensor uncertainty which considers the uncertainty at each node as independent from the connecting nodes. A local controller utilises sensor feedback to develop paths along edges which will result in the belief at the connecting node. Pilania and Gupta [128] constructed a belief-based PRM for a mobile manipulator assuming a Gaussian distribution for the base location. An EKF algorithm is used to keep the robot localised while travelling between nodes. The algorithm will reconfigure the manipulator only as necessary to bypass obstacles.

Uncertainty can be used in RRT algorithms in many ways. A particle RRT algorithm, presented by Melchior and Simmons [133], considered uncertainty in a motion model when connecting a node to the tree. Nodes represent a cluster of states and are associated with a probability of being reached. Paths are found by following the highest probable path. Vasquez-Gomez et al. [62] developed an RRT algorithm for developing paths between the next best views. As the robot moves along paths generated through an RRT algorithm the size of the hyper volume representing the robot increases with the increase of uncertainty. The hyper volume is randomly sampled for collision checks which, if failed, disregards the view being moved towards.

2.3.2.3 Sensor Based Solutions

Often the uncertainty in the state can be mitigated by incorporating sensor readings.

A Kalman filter is a method for incorporating uncertainty in planning algorithms and is comprised of two main steps; a prediction step and an update step. The prediction step is performed to estimate the next states based on propagating uncertainty though the transitional model. The update step then compares the estimated states to sensor measurements and redistributes the uncertainty in the system to the most likely states. This process is recursive. An EKF extends the Kalman filter to nonlinear systems. Gonzalez and Stentz [185] used a first-order motion model for a robot with an EKF. A single error parameter is used to simplify the representation of the uncertainty with the resulting boundary region representing possible robot positions. Localisation regions are used throughout the map to reduce the uncertainty. An optimal path is found which minimises the distance to the target and the uncertainty in the state. Gonzalez and Stentz [186] extended the approach to linear features which minimises the uncertainty along a single dimension, e.g. along the x-axis for a mobile wheeled robot. Bloesch et al. [48] presented an EKF which fuses data from IMU measurements and encoder readings to predict the state of the body of legged robots. While capable of handling unknown terrains and arbitrary gaits, the approach fails to observe the absolute position and yaw in the body in some cases. Rotella et al. [187] extend this research to a bipedal robot by including a rotational constraint provided by flat feet. Although these approaches could be used by considering uncertainty after a step has occurred they do not allow reactive collision avoidance to occur. Benallegue and Lamiraux [49] used IMU measurements and contact information to estimate the deformation due to a flexible section between ankle and sole of a bipedal robot using an EKF. The deformation in the flexible section is represented by a 6DOF homogeneous transformation matrix effectively adding a virtual joint between the global coordinate frame and the robot. Two IMUs, one positioned in the chest and a virtual IMU in the foot, are used to model the position of the foot in the global coordinate frame. Using an EKF, predictions of the deformation are made based on deviations in the hand position of the bipedal robot. As this approach assumes the contact position and the relative location of each joint within the system state to be known, it cannot be used with the inchworm robot. Ma et al. [46] developed an EKF algorithm to maintain accuracy in a 6DOF (position and orientation) state of the body of a quadruped walking robot which utilises GPS position updates, a stereo camera, IMU measurements and leg odometry. The integration of multiple sensors into the EKF algorithm allows the state to be known with greater accuracy. Roveda et al.

[142, 143] considered a manipulator mounted upon a compliant base which considers the dynamic parameters of the environment. In this strategy the base is estimated using a Kalman Filter, while the dynamic environment parameters use an EKF. Estimates are used in conjunction with the measured end-effector force to update the impedance control setpoint. An impedance controller maintains this setpoint by updating the forces acting on the system. An unscented Kalman filter (UKF) is a variation of the EKF which use sample points representative of the probability distribution in the system and are generally used in highly non-linear systems [188]. Kaelbling and Lozano-Pérez [189] used a UKF for an integrated controller for a mobile manipulator with uncertainty. A step of an initial path is executed and the state observed. A new path is then created based on this observed state. This is repeated until the observed state satisfies the goal condition. As the uncertainty in the inchworm does not propagate over time, EKF based solutions are of limited use. Sensor reading can be used to reduce the uncertainty at a single transitional step; however, at the next transitional step the state would be unknown again.

Non-EKF based solutions using sensor readings exist. Dogar et al. [47] presented a method for localising a mobile manipulator performing a task with 2D laser scans and proprioceptive collision data. Particles are used to represent possible base locations which have their beliefs updated through an Adaptive Monte Carlo Localisation based on laser scans and torque readings which are both used to detect collisions. De Schutter et al. [190] presented a systematic approach to complex sensor-based robot tasks with geometric uncertainty. This is achieved by creation of a new set of reference frames. These frames are attached to the robot, sensors and features, according to a set of defined rules, to form a kinematic chain. The robot dynamics and modelled uncertainties are then established according to these new reference frames. While this framework may provide a means to control the inchworm robot with uncertainty, it may be difficult to explicitly define all required motion for the inchworm robot through task based control.

2.3.2.4 Optimisation Based Methods

Given an objective function and constraints, optimisation methods provide locally or globally optimal solutions for path planning with uncertainty.

A Linear-quadratic Gaussian (LQG) algorithm is an optimisation technique for developing task-based motion which accounts for uncertainty. Van den Berg et al. [26] presented a LQG motion planning (LQG-MP) algorithm which takes into account motion and sensor uncertainty. Given a set of stochastic models, a priori probability distributions of the robot state are determined along a set of generated discrete paths. By using Gaussian models for the uncertainty, a Kalman filter is used to determine the a priori distributions and can be used to determine probability of collisions. The best path is then found based on minimising an objective function. Simulations were performed on a car-like robot, in multi-robot planning and a 6DOF serial manipulator. Van den Berg et al. [191] further applied the LQG-MP algorithm for a steerable needle to compute sensor placement to find paths which minimise the probability of intersecting with obstacles. The LQG framework is evaluated, with a large set of random positions for the given sensor, to determine the optimal sensor placement. Yadav and Singh [192] considered uncertainty through high-order cost statistics applied to the LQG framework. As opposed to minimising an explicit cost function, this method used a risk-sensitive decision process to vary the uncertainty to create a more accurate model and improved trajectories. The LQG-MP algorithms determine locally optimal solutions based on a priori knowledge of distributions along a pre-generated path and localisation methods. If the path is not established prior to execution, optimisation is not possible.

2.3.2.5 Markov Decision Process

A Markov decision process (MDP) is another framework for accounting for uncertainty in a system. This framework considers a finite set of states, a finite set of actions and an action model. This action model specifies the probability of a new state occurring when an action is applied. State and actions have associated costs which are dependent on the application. Through iterative methods, a policy is generated which finds the optimal cost of reaching the target. The MDP assumes that the state is observable (known) which is not always the case. Partially observable MDP can be used if the state can be partially observed. Hauser [193] presented a belief space replanning strategy within a POMDP framework. The belief state is represented using a weighted set of particles with state estimation performed using a particle filter. Random exploration is performed using open-looped actions to determine partial paths. This strategy selects the path with the highest likelihood of success. It is acknowledged, however, that this implementation is not reliable in high dimensional state space. Van den Berg et al. [194] extended previous work with LQG algorithms to belief space within a POMDP framework. Value iteration is performed using a belief space variant of the LQG algorithm to find a locally optimal control policy over a continuous trajectory. A new trajectory is determined and followed from this policy and the process is repeated until a locally optimal solution is found. This approach overcomes the limitation of other POMDP solutions as it does not require discrete state space, however it does require an a priori trajectory. If an a priori trajectory was available to the inchworm robot further optimisation of the trajectory may not be necessary. While algorithms within the MDP and POMDP framework are powerful they can become computationally intractable for highly dimensional states and long planning horizons. Du et al. [195] presented ideas for improving the efficiency in POMDPs. Reductions in computational costs are achieved by reducing the dimensionality of the space by separating observable and partially observable states; only beliefs over the partially observed states are maintained. Additional reductions are achieved by sampling only after a sequence of action-observations pairs have been performed as, in some cases, different pairs may result in similar effects on the belief space. Kurniawati et al. [196] developed a point-based POMDP, Milestone Guided Sampling, which reduces the complexity of planning in long-time horizons. Milestone Guided Sampling constructs a roadmap of milestones by sampling the robot's state space and develops a series of actions for planning between milestones. Planning efficiency is improved by sampling the belief space based on this roadmap. Kurniawati et al. [24] improved the efficiency of POMDPs with feature-based maps whereby the inclusion of feature uncertainty in the state variable can greatly increase the difficulty of the problem. Kurniawati et al. introduced Guided Cluster Sampling which reduces the belief space into smaller sub-spaces and transforms robot uncertainty into the map uncertainty. While this technique may be used with the inchworm robot it relies on feature based maps which is not applicable with the current system.

Through the POMDP framework, research has been performed to learn unknown parameters while planning. Ross et al. [197] presented a method for overcoming unknown state models during planning in POMDPs by using Bayesian reinforcement learning. Assuming known action and observation spaces, the robot is modelled as a Gaussian system with a particle filter used to estimate the next system state. Fixed planning horizon paths are generated online with actions and observations selected to maximise the reward function. This algorithm is tested on a robot in 2D attempting to learn its encoder drift parameters. Ross et al. [198] continues to addresses the intractability of the belief space by approximating the model for the POMDP model. Ross et al. [197] research is also extended by Dallaire et al. [199] by incorporating Gaussian Process Dynamic Models to learn the transition, observation and reward functions. Bai et al. [134] presented a model-based Bayesian reinforcement learning as a POMDP. The POMDP is tailored to both plan a path to the goal and learn unknown parameters offline. The results are used to create a finite-state controller which is then implemented online.

2.4 Discussion on the Related Work

Path planning for an inchworm robot with uncertainty in the given application scenarios is challenging. The inchworm robot suffers from a number of uncertainties. Existing literature has shown that many of these uncertainties can be accounted; however, the unique combination of structural uncertainty and hand position uncertainty, coupled with the high dimensionality of the state, is not addressed. It is established that a probability distribution should be used to develop a representative model of the inchworm robot under the influence of uncertainty. As the uncertainty in the state does not propagate each time step, the uncertainty should be dependent on the current state.

The 3D-F^2 algorithm [155] allows smooth path planning for multi-link serial robots. This algorithm allows point-to-point paths to be generated without the need for inverse kinematics. While the 3D-F^2 algorithm was used without consideration for uncertainty, the method lends itself to be used with state uncertainty. The force fields surrounding the robot could increase as the uncertainty in the state increases which would allow for reactive collision avoidance. Additionally, by modifying the methods used to generated forces, surface transitions can be facilitated on any orientated surface. A graph-based algorithm is developed which generates longer time-horizon paths. This would improve the performance of the developed uncertainty-based 3D-F^2 algorithm.

Chapter 3

Modelling Structural and Hand Position Uncertainty

Safe motion is a requirement for robotic systems. If the state is known and accurate, sound decisions can be made during path planning to facilitate safe motion. However, in real robotic applications collisions may occur due to uncertainties. Uncertainty should be considered in the model describing the robot state. This then allows the path planner to compensate for the uncertainty and develop collision-free paths.

This chapter first defines the deterministic kinematic model of a multi-link serial robot. Several factors cause modelling the uncertainty for multi-link serial robots to be a challenge: the configuration of the robot, the materials used, the design of each component and the large number of states. The proposed model interpolates empirical knowledge of the uncertainty at the end-effector to describe the effect of the uncertainty at the coordinate frame of reference of each joint.

Two major sources of uncertainty are identified with the inchworm robot - structural uncertainty and hand position uncertainty. These uncertainties are used to describe a kinematic model of a multi-link serial robot which is representative of the inchworm type robots. This model can be extended to describe kinematically similar robots such as soft robots.

The outline of the chapter is as follows: definitions and models of the structural and hand position uncertainties are presented; a robot model with both structural and hand position uncertainties is defined; and methods used for parameter verification are described.

3.1 System Definition

A kinematic model describing the inchworm robot without uncertainty is described using Denavit-Hartenberg (DH) parameters. These parameters relate coordinate frames of reference through homogeneous transformation matrices, T. The parameters a_i and d_i are the distances along the x- and z-axis from the preceding frame of reference, and q_i and α_i are the rotations about the x- and z-axis from the previous frame of reference. The homogeneous transformation matrix ${}^{i-1}T_i$ describes the coordinate frame of reference of a link, i, associated with a joint, q_i, with respect to the coordinate frame of reference of the preceding link, $i-1$ (Equation 3.1).

$$
{}^{i-1}T_i(q_i) = \begin{bmatrix} \cos q_i & -\sin q_i \cos\alpha_i & \sin q_i \sin\alpha_i & a_i \cos q_i \\ \cos q_i & \cos q_i \cos\alpha_i & -\cos q_i \sin\alpha_i & a_i \sin q_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}
$$

Any homogeneous transform matrix, T, is comprised of two components; a 3×3 rotation matrix, $\mathbf{R}$, and a 3×1 position vector, $\mathbf{P}$ (Equation 3.2). The rotation matrix can be further decomposed to 3×1 normal, approach and orientation vectors, $\mathbf{R}_n$, $\mathbf{R}_a$ and $\mathbf{R}_o$ respectively, which describe the orientation of the x-, y- and z-axis of the homogeneous transform matrix relative to the preceding coordinate frame.

$$
T = \begin{bmatrix} \mathbf{R} & \mathbf{P} \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_n & \mathbf{R}_a & \mathbf{R}_o & \mathbf{P} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & a_x & o_x & x \\ n_y & a_y & o_y & y \\ n_z & a_z & o_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}
$$

The base of the robot, T_0, is described relative to the origin point in a global coordinate frame, Ψ_G. The base describes the location of the foot pad. This is the pad currently adhering to the surface. The terms "foot pad" and "base" are used synonymously in the rest of this thesis (Figure 3.1).

A vector, $\mathbf{q}$, of n joints is used to describe the state of the joint angles, q_i, in radians (Equation 3.3). For the inchworm robot, $n = 7$.

$$\mathbf{q} := \begin{bmatrix} q_1, & q_2, & ..., & q_n \end{bmatrix}^T \tag{3.3}$$

The location of a robot link, i, and associated joint, q_i, within the global coordinate frame, Ψ_G, is described using a homogeneous transform matrix (Equation 3.4).

$${}^0T_k(\mathbf{q}) = T_0 \prod_{i=1}^{k} {}^{i-1}T_i(q_i) \qquad k = 1, ..., n \tag{3.4}$$

The location of the inchworm robot's end-effector, ${}^0T_e$, is described using a homogeneous transform matrix (Equation 3.5). The end-effector location describes the location of the

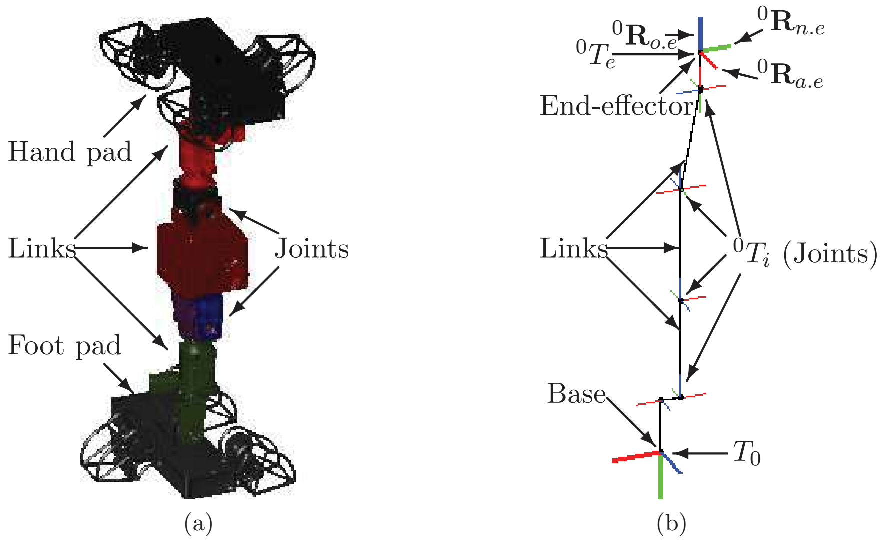


FIGURE 3.1: (a) Representation of the inchworm robot. (b) Multi-link serial robot representation of the inchworm robot.

pad which is currently free to move. This pad is called the hand pad. The terms "hand pad" and "end-effector" are used synonymously in the rest of this thesis.

$$^{0}T_e = {}^{0}T_7{}^{7}T_e \tag{3.5}$$

The following sections describe the effect the uncertainty has on this deterministic model.

3.2 Structural Uncertainty

The body of a robot under external forces can suffer from structural deformation; the greater the force the larger the deformation. Under normal operation, and provided that no collisions occur, the deformation on a robot generally occurs from gravitational forces, impulses and dynamic motions (Figure 3.2). For a multi-link serial robot the deformation leads to the coordinate frame of reference, $^{0}T_i$, to differ from its expected value, which may lead to collisions.

In this research the inchworm robot is designed to move slowly due to the complexity of the environment and the requirements of the operation, such as safety. Henceforth when discussing deformation only the effect of gravitational forces are considered.

In regards to the inchworm robot, its symmetrical design induces high moment loads due to the adhesion mechanism on the pads at both ends of the robot. This cannot be avoided as both pads must be capable of supporting the entire weight of the robot even in a cantilever

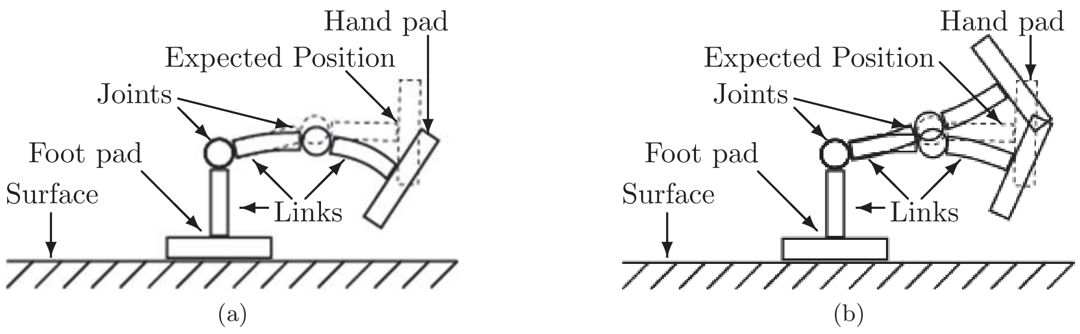


Figure 3.2: Uncertainty in links due to *a)* gravitational forces and *b)* impulses or dynamic motions.

state. This state generates the highest loads on the adhesion mechanisms. Additionally, as the first and last joints must both be capable enough to ensure smooth motion, they are heavier in comparison to the other joints. These other joints are progressively lighter and weaker the closer they are to the centre of the inchworm robot. These factors induce significant gravitational loads on the inchworm robot causing structural deformation.

For a given pose and base location, the structural deformation in a multi-link serial robot can be quantifiable. However, computing the structural deformation for all possible robot states is challenging. The difficulty is compounded by factoring in the configuration of the robot, the materials used and the design of each component. A probabilistic representation is used to estimate the effect of the structural deformation.

3.2.1 Modelling Structural Uncertainty

This section formulises the effect of the structural uncertainty. The coordinate frame of reference for a link is described using a homogeneous transform matrix (Equation 3.4). The structural deformation causes the coordinate frame of reference to differ from the expected one because of the interaction between gravity and four major factors: the configuration of the robot, the materials used, the design of each component and the high number of states.

Gravitational loads in the inchworm robot can induce various deformations as shown in Figure 3.3. While calculating these loads may be possible, calculating their effect on the deformation in all possible states is non-trivial. Instead a probability distribution can be used to estimate the effect of deformation at the coordinate frame of reference of a joint. This is calculated based on empirical knowledge of the structural deformation at the end-effector in the worst case robot state.

The worst case structural deformation occurs in a cantilever state. In this state the robot's base is positioned on a vertical surface; this state has the greatest distance between the base and end-effector, l_c, and the coordinate frame of reference for the majority of joints have their y-axis aligned with gravity (Figure 3.4). This structural deformation at the end-effector has two components; a translational component aligned with gravity, E_c,

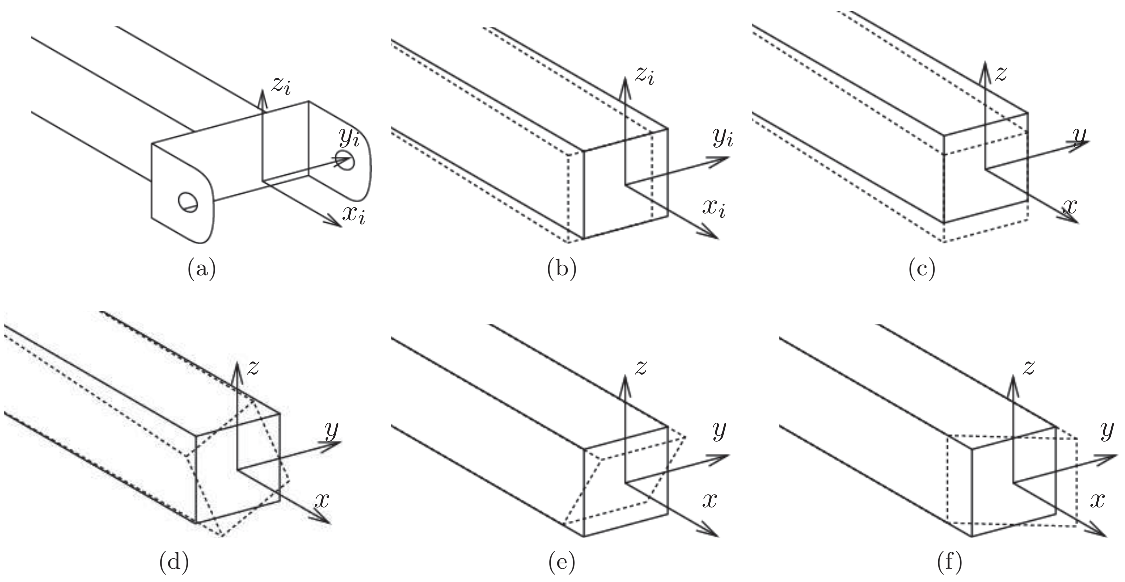


FIGURE 3.3: (a) The coordinate frame of a link, i, (b) deformation along the y-axis, (c) deformation along the z-axis, (d) deformation about the x-axis, (e) deformation about the y-axis, (f) deformation about the z-axis.

and rotational component, γ_c. These are considered the allowed maximum values. The rotational component is considered perpendicularly to both gravity and the vector between the origins of the base and end-effector of the robot. Both components are considered from the coordinate frame of reference of the base. From these components an estimate is determined for the structural deformation at a joint's coordinate frame of reference.

Before determining the structural deformation at a joint's coordinate frame, a number of assumptions should be stated:

1. The coordinate frame of reference of a joint will deform in the same direction as, but not exceed, the allowed maximum translational deformation, E_c.
2. The coordinate frame of reference of a joint will rotate perpendicularly about both a vector created between the base and the end-effector, and gravity.
3. The rotation at a coordinate frame of reference of a joint due to the structural deformation will not exceed the allowed maximum rotational deformation, γ_c.
4. The structural deformation at the base is zero.

5. The structural deformation at the coordinate frame of reference of a joint will then lie between zero and the allowed maximum deformation depending on the distance between the foot pad and the joint.

The maximum distance along the xy-plane, $d_{i.xy}$, is found from the position of the coordinate frame of reference of a joint, ${}^{0}\mathbf{P}_{i.xy}$, to the position of each preceding joint's coordinate frame of reference including the base, ${}^{0}\mathbf{P}_{k.xy}$ (Equation 3.6). This is the maximum cantilever distance and can account for poses where the robot is completely parallel with gravity (Figure 3.5a) or when the robot is wrapped upon itself (Figure 3.5b). A joint distance ratio, $\beta_{c.i}$, is defined to relate the xy-distance over the maximum distance from the base (Equation 3.7). Two functions are used which relate the joint distance ratio to the allowed maximum translational and rotational deformations, G_c and F_c. These functions allow a conservative estimate for the translational and rotational components of the structural deformation at the coordinate frame of reference of a joint to be determined, $E_{c.i}$ and $\gamma_{c.i}$ (Equation 3.8 and 3.9). As the allowed maximum translational and rotational deformations are considered relative to the base coordinate frame, the components at each joint are similarly considered relative to the base coordinate frame.

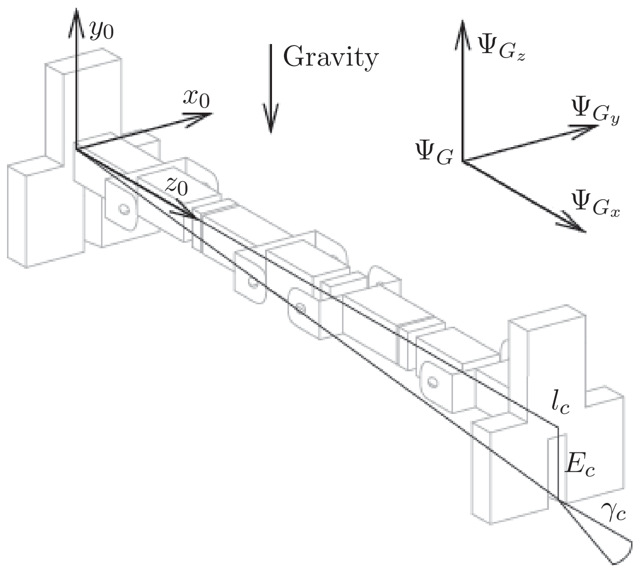


FIGURE 3.4: Visualisation of inchworm robot in cantilever with two toes aligned with gravity.

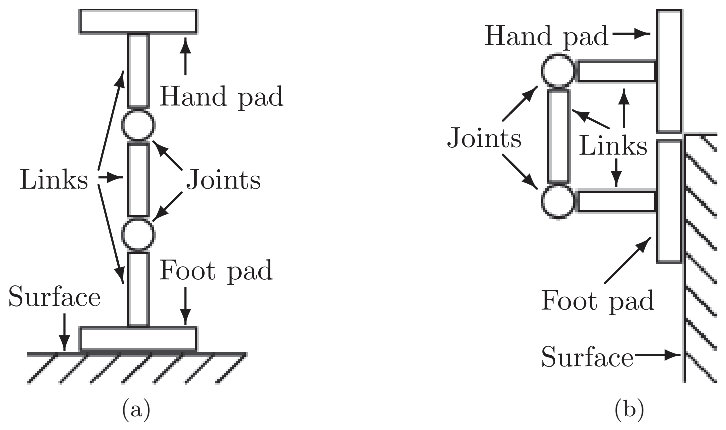


FIGURE 3.5: (a) Robot with joints directly above each other. (b) Robot in a wrapped pose with the base and end-effector above each other.

$$d_{i.xy} = \max_{k \in 0,...,i-1} \|{}^0\mathbf{P}_{k.xy} -{}^0 \mathbf{P}_{i.xy}\| \qquad i = 1, .., n \tag{3.6}$$

$$\beta_{c.i} = \frac{d_{i.xy}}{l_c} \qquad i = 1, .., n \tag{3.7}$$

$$E_{c.i} = G_c(\beta_{c.i}) \qquad i = 1, .., n \tag{3.8}$$

$$\gamma_{c.i} = F_c(\beta_{c.i}) \qquad i = 1, .., n \tag{3.9}$$

The functions only allow a conservative estimate as the joint distance ratio does not account for the configuration of the robot, the materials used nor the design of each component. A probability distribution may be applied to account for these factors with a Gaussian distribution the most likely representation; however, in this study, statistical modelling is not used. Instead, worst case values are used to represent the structural deformation and are parameterised by the allowed maximum means and variances. The allowed maximum values provide a conservative, safer solution for navigation for the inchworm robot presented within this study.

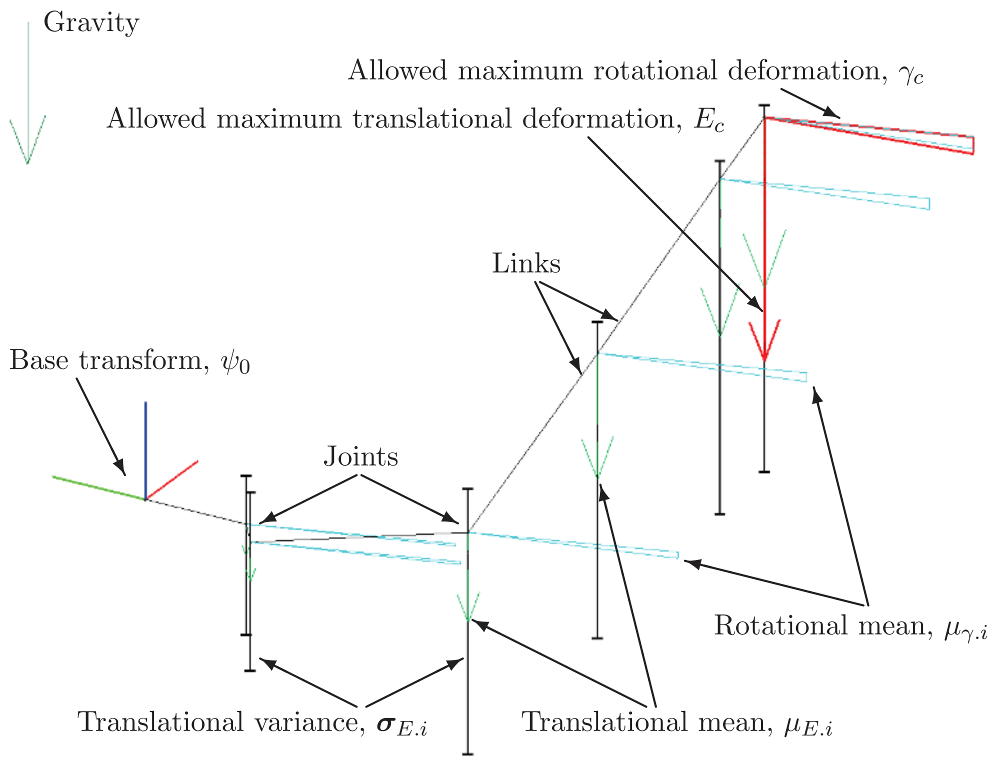
The means, $\mu_{E.i}$ and $\mu_{\gamma.i}$, are defined by the translational and rotational components of the structural deformation at the coordinate frame of reference for a joint, $E_{c.i}$ and $\gamma_{c.i}$ (Equations 3.10 and 3.11). There exists variance in both components for the structural uncertainty. However, the variance in the rotational component at a joint is relatively small and is assumed to be zero. The variance in the translational component can be considered as a portion of the structural deformation's allowed maximum translational variance. The method applied to calculate this portion is the same as calculating the components of the structural deformation of a joint. This calculation results in the variance at each joint's coordinate frame of reference. The joint distance ratio, $\beta_{c.i}$, is applied to a function, J_c, which relates the ratio to the allowed maximum translational variance (Equation 3.12) to find the variance at the joint's coordinate frame of reference, $\sigma_{E.i}$. A matrix is developed, based on the convention described with Equation 3.2, to represent this variance in 3D, $\boldsymbol{\sigma}_{E.i}$ (Equation 3.13). As the structural translational variance is taken along the z-axis, only the $\mathbf{R}_o$ portion of the convention is considered. Figure 3.6 shows the inchworm robot in an arbitrary pose with the resulting mean and variance visualised in 3D.

$$\mu_{E.i} = E_{c.i} \qquad i = 1,..,n \tag{3.10}$$

$$\mu_{\gamma.i} = \gamma_{c.i} \qquad i = 1,..,n \tag{3.11}$$

$$\sigma_{E.i} = J_c(\beta_{c.i}) \qquad i = 1,..,n \tag{3.12}$$

$$\boldsymbol{\sigma}_{E.i} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_{E.i} \end{bmatrix} \qquad i = 1,..,n \tag{3.13}$$


Gravity
Allowed maximum rotational deformation, γ_c
Allowed maximum translational deformation, E_c
Links
Base transform, ψ_0
Joints
Rotational mean, $\mu_{\gamma.i}$
Translational variance, $\boldsymbol{\sigma}_{E.i}$
Translational mean, $\mu_{E.i}$

(a)

Figure 3.6: The mean and variance of the structural deformation for a multi-link serial robot in an arbitrary pose. Green arrows and cyan arcs show the mean in the translational and rotational components of the structural uncertainty at a joint's coordinate frame of reference. The variance in the translational component is shown as bars extending either direction from the mean. The red arrow and arc show the allowed maximum translational and rotational structural uncertainty imposed on the current end-effector position.

3.3 Hand Position Uncertainty

The permanent magnet adhesion mechanisms situated on both ends of the inchworm robot are used to adhere to a surface and provide a means for locomotion in the environment. When the foot pad is attached to a surface the hand pad is free to move. Locomotion is performed by manoeuvring the hand pad to a goal location on the surface. The hand pad is then attached and the foot pad detached which allows it to move freely. The base coordinate frame of reference is updated to reflect the new foot pad location and the pads swap designations.

A number of factors cause an uncertain hand pad location. The surface in real environments is not always perfectly uniform. Rivets, poor surface condition (i.e. rust or flaking paint), step changes or curved surfaces may cause inaccurate pad location (Figure 3.7a).

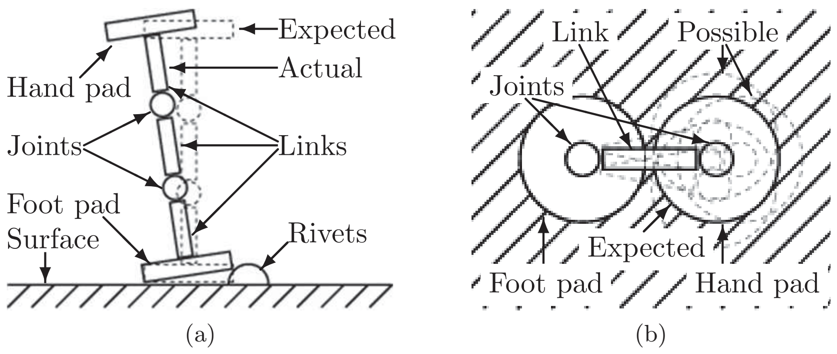


FIGURE 3.7: (a) Uncertainty in the hand pad location due to stepping on a rivet (side view). (b) Uncertainty in the hand pad location due to the hand position uncertainty shown with dotted outlines (top-down view).

These concerns may be mitigated provided that a surface goal location is selected which is not on top of these surface irregularities and localisation can be performed to ascertain the foot pad location.

A major concern may arise during surface attachment. Attachment occurs by adjusting the orientation of the permanent magnets to increase the magnetic force acting towards the surface. Once the force becomes high enough the foot pad is forced to comply to the surface. In an ideal situation the pad will attach to the desired goal location; however, the magnetic force exerted by the permanent magnets may result in the pad deviating from the goal location (Figure 3.7b). This deviation is considered as the hand position uncertainty. When the hand pad deviates, the body may be forced to comply to allow the deviation which may result in collisions along the length of the robot.

No sensors or techniques are currently available on the robot to precisely measure or predict the effects of this uncertainty. As there exists a balance between the magnet strength, adhesion and the weight of the robot there are limited hardware modifications possible to alleviate this issue. Localisation may be used to determine the actual location after attachment occurs but it is challenging to implement while in action due to high computational costs. Instead a probability distribution is used to estimate possible landing positions around the surface goal location.

3.3.1 Modelling Hand Position Uncertainty

The goal location on a surface is parameterised by a 3×1 position vector, ${}^{0}\mathbf{P}_{g}$, and a 3×1 orientation vector, ${}^{0}\mathbf{R}_{o.g}$. This follows the transform convention described in Equation 3.2. The hand pad is considered at the goal when both its position, ${}^{0}\mathbf{P}_{e}$, and orientation vector, ${}^{0}\mathbf{R}_{o.e}$, are aligned with the goal location. If it is assumed that the surface at the goal position is flat and the hand pad is attached, the position along the z-axis, $P_{z.e}$, and rotations about the x- and y-axes, $\mathbf{R}_{n.e}$ and $\mathbf{R}_{a.e}$, at the coordinate frame of reference of the hand pad are not affected by the hand position uncertainty (Figure 3.8). The hand position uncertainty then affects the position along the x- and y- axes, $P_{x.e}$ and $P_{y.e}$, and the rotation about the z-axis, $\mathbf{R}_{o.e}$. A probability distribution can be used to represent the hand position uncertainty.

A single variable 2D Gaussian is the most likely representation for the hand position uncertainty; however, in this study, statistical modelling is not used. Instead, worst case values are used to represent the hand position uncertainty along the x- and y-axes of the hand pad's coordinate frame of reference. The worst case values provide a conservative, safer solution for navigation for the inchworm robot presented in this study. These are parameterised by an allowed maximum mean for each axis, $\mu_{x.h}$ and $\mu_{y.h}$, and an allowed maximum variance, σ_h. As the variance is centred around the goal position both mean values are zero; the variance is empirically determined. While it is acknowledged that the hand position uncertainty affects the rotation about the z-axis, $\mathbf{R}_{o.e}$, it is minimal in comparison to the effect along the surface and is assumed to be zero.

The hand position's allowed maximum variance can be described in 3D using a matrix, $\boldsymbol{\sigma}_h$ (Equation 3.14). This matrix is considered from the hand pad's coordinate frame of reference (Figure 3.8).

$$\boldsymbol{\sigma}_h = \begin{bmatrix} \sigma_h & 0 & 0 \\ 0 & \sigma_h & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.14}$$

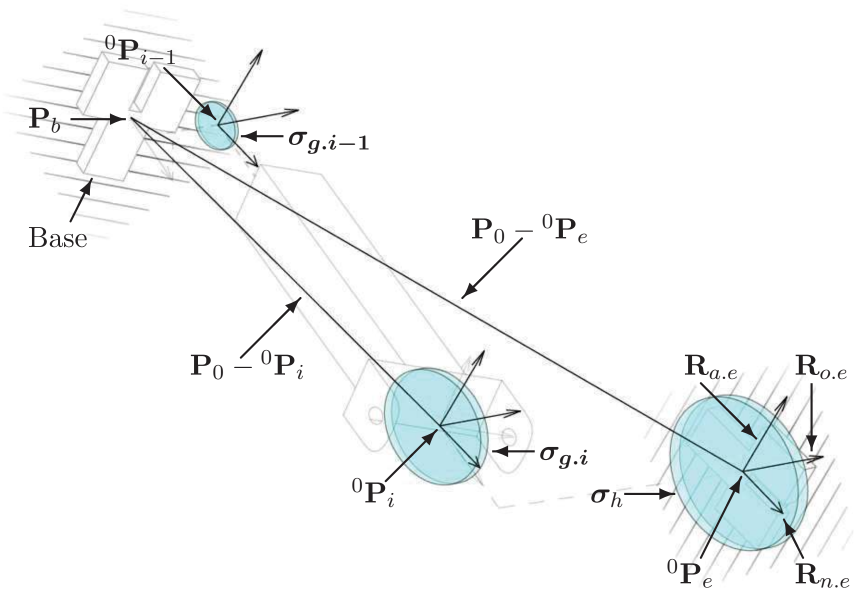

$^0\mathbf{P}_{i-1}$
$\mathbf{P}_b$
$\boldsymbol{\sigma}_{\boldsymbol{g.i-1}}$
Base
$\mathbf{P}_0 - {}^0\mathbf{P}_e$
$\mathbf{P}_0 - {}^0\mathbf{P}_i$
$\mathbf{R}_{a.e}$
$\mathbf{R}_{o.e}$
$\boldsymbol{\sigma}_{\boldsymbol{g.i}}$
$^0\mathbf{P}_i$
$\boldsymbol{\sigma}_h$
$^0\mathbf{P}_e$
$\mathbf{R}_{n.e}$

FIGURE 3.8: A representation of a multi-link serial manipulator with hand position uncertainty. The blue region represents the variance at the coordinate frame of reference of joints and at the end-effector. The blue regions have been shown with height to improve the visualisation only.

The robot body will move to conform to the inaccurate hand position caused by attachment which may lead to collisions. The hand position uncertainty should then be considered at the coordinate frame of reference of each joint to give an indication of the expected deviation along the robot body. It is then assumed that the coordinate frame of reference of a joint will conform to allow the inaccurate hand position at the surface caused by attachment. It is assumed that the hand position uncertainty does not affect the position of the base which is known.

The effect of hand position uncertainty at the coordinate frame of reference of a joint is characterised by two means, $\mu_{x.i}$ and $\mu_{y.i}$, and a variance, $\sigma_{h.i}$. As the hand position uncertainty deviates around the goal location, the robot position will deviate around the pose at this location resulting in the mean hand position uncertainty at each joint being zero. The variance at a joint will be between zero and the hand position's allowed maximum variance, σ_h, depending on the joint's relative position; as the position of the joint tends towards the base, the variance tends towards zero; as the position of the joint tends towards the hand pad location the variance tends towards the hand position variance. A ratio is found to this effect which relates the distance between the position of the base, $\mathbf{P}_0$, and a joint, ${}^{0}\mathbf{P}_i$, over the distance between the position of the base and the end-effector, ${}^{0}\mathbf{P}_e$ (Figure 3.8). A function, $L(..)$, is established which relates the ratio to the variance at each joint's coordinate frame of reference, $\sigma_{h.i}$ (Equation 3.15 and Figure 3.8).

$$\sigma_{h.i} = L\left(\frac{\|\mathbf{P}_0 - {}^{0}\mathbf{P}_i\|}{\|\mathbf{P}_0 - {}^{0}\mathbf{P}_e\|}\right) \qquad i = 1, .., n \tag{3.15}$$

The hand position uncertainty should only be considered when approaching a goal location. To facilitate this a growth factor, K_g, is used which modifies the variance based on the distance from the end-effector position to the goal location (Equation 3.16). Here d_a is an preparatory approach distance, d_b is a surface approach distance and d_e is the distance from the end-effector to the goal location (Figure 3.9). These are used to determine points which are described by distances from the positional component of the goal location along the goal location's rotational component. These points are used for the surface attachment procedure. The point described by the preparatory approach distance is used as the goal location to begin the surface attachment procedure. The complete procedure is explained in Section 4.2.8. The end-effector is then moved towards the goal location. The growth factor will increase between the points described by the preparatory and surface approach distances then remain constant at a maximum between the surface approach distance and the goal location. The growth factor is applied to modify the variance at each joint, $\sigma_{g.i}$ (Equation 3.17).

$$K_g = \begin{cases} \frac{d_a - d_b}{d_i - d_b} & d_e \geq d_b \\ 1 & d_e < d_b \end{cases} \tag{3.16}$$

$$\sigma_{g.i} = K_g \sigma_{h.i} \qquad i = 1, .., n \tag{3.17}$$

The variance due to the hand position uncertainty at each joint's coordinate frame of reference, $\sigma_{g.i}$, can be described using a 3D matrix with respect to the coordinate frame of reference of the end-effector, $\boldsymbol{\sigma}_{g.i}$ (Equation 3.18 and Figure 3.10). To describe the variance relative to the joint's coordinate frame of reference, $\boldsymbol{\sigma}_{h.i}$, instead of the pad's coordinate frame of reference the rotational component of the coordinate frame of reference of the hand pad, ${}^{0}\mathbf{R}_e$ is applied (Equation 3.19).

$$\boldsymbol{\sigma}_{g.i} = \begin{bmatrix} \sigma_{g.i} & 0 & 0 \\ 0 & \sigma_{g.i} & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad i = 1, .., n \tag{3.18}$$

$$\boldsymbol{\sigma}_{h.i} = {}^{0}\mathbf{R}_e \boldsymbol{\sigma}_{g.i} {}^{0}\mathbf{R}_e^{-1} \qquad i = 1, .., n \tag{3.19}$$

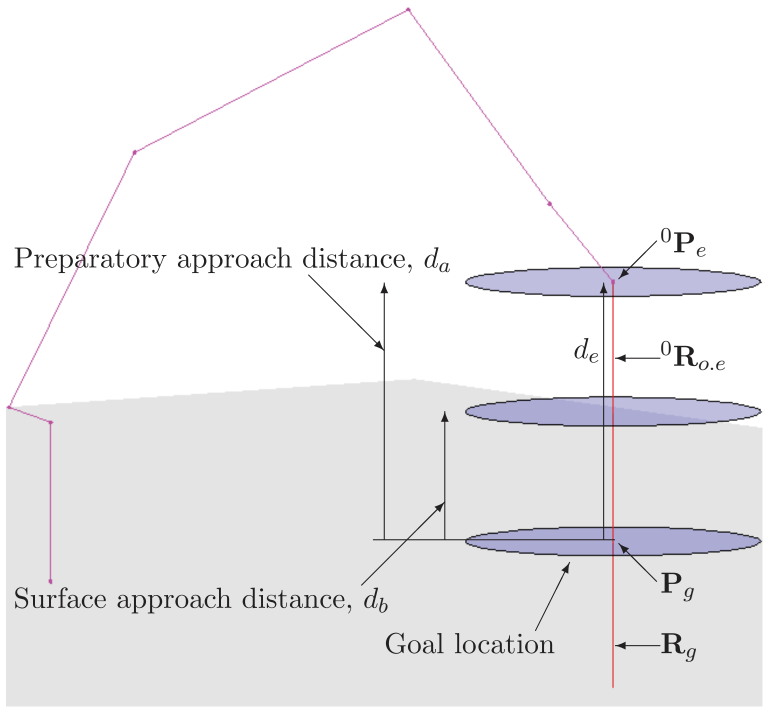


FIGURE 3.9: Different distances relating to different growth factors. At the preparatory approach distance the growth factor is zero, while at the surface approach distance the growth factor is at a maximum.

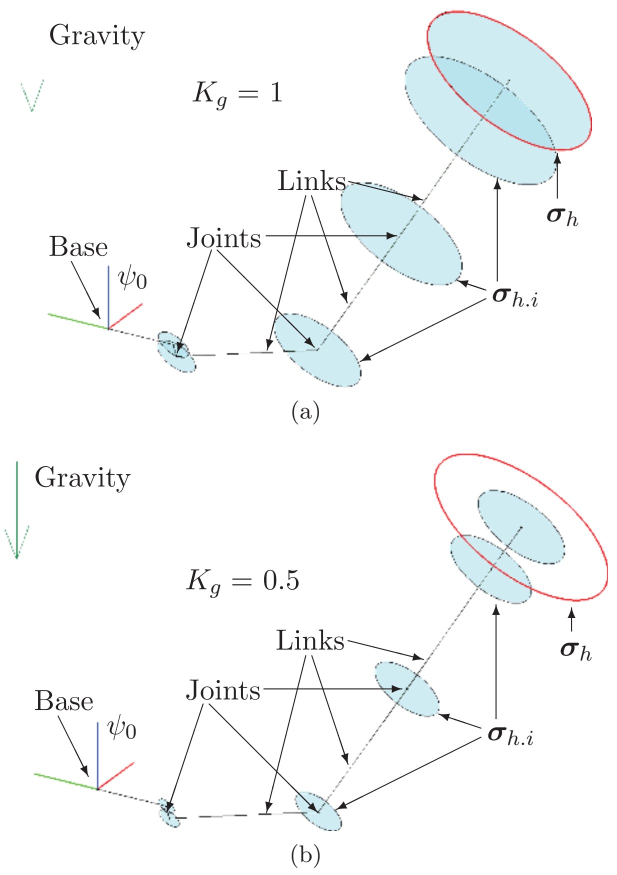


FIGURE 3.10: (a) The variance at a maximum growth factor (b) and with a growth factor of 0.5. The red circle at the hand pad represents the maximum variance with the cyan discs showing the variance at each link's coordinate frame of reference.

3.4 Robot Model with Structural and Hand Position Uncertainties

The presented model describes the most likely position of each joint in a multi-link serial robot in the global coordinate frame of reference. It is defined using a 4x4 mean homogeneous transformation matrix which relates the deterministic joint coordinate frames of reference to their mean location. This mean homogeneous transformation matrix is defined using the mean values of the uncertainty.

For the structural and hand position uncertainties the means $\mu_{E.i}$ and $\mu_{\gamma.i}$ and variances $\boldsymbol{\sigma}_{E.i}$ and $\boldsymbol{\sigma}_{h.i}$ are established for each joint. A homogeneous transformation matrix, ${}^{i}\mathcal{T}_{\mu.i}$, is found which describes the means with respect to the coordinate frame of reference of the joint (Equation 3.20). The matrix considers the mean $\mu_{E.i}$ in the positional vector, ${}^{i}\mathcal{P}_{\mu.i}$, and the mean $\mu_{\gamma.i}$ in the rotational vector, ${}^{i}\mathcal{R}_{\mu.i}$. As these means are considered with respect to a vector between the base and the end-effector, a transformation matrix is applied which relates this vector to the coordinate frame of reference of the link, $\psi_{R.i}$ [200]. The matrix, ${}^{i}\mathcal{T}_{\mu.i}$, is then applied to the transformation describing the joint coordinate frame of reference, ${}^{0}T_{i}$, resulting in a coordinate frame of reference for the joint, ${}^{0}\mathcal{T}_{i}$, describing its mean position in the global coordinate frame (Equation 3.21). The coordinate frame of reference for a joint has both a 3×3 rotational, $\mathcal{R}$, and 3×1 positional, $\mathcal{P}$, components (Equation 3.22).

$$ {}^{i}\mathcal{T}_{\mu.i} = \begin{bmatrix} {}^{i}\mathcal{R}_{\mu.i} & {}^{i}\mathcal{P}_{\mu.i} \\ 0_{1\times3} & 1 \end{bmatrix} \psi_{R.i} = \begin{bmatrix} \cos\mu_{\gamma.i} & -\sin\mu_{\gamma.i} & 0 & 0 \\ \sin\mu_{\gamma.i} & \cos\mu_{\gamma.i} & 0 & 0 \\ 0 & 0 & 1 & \mu_{E.i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \psi_{R.i} \qquad i = 1, ..., n \tag{3.20} $$

$$ {}^{0}\mathcal{T}_{i}(q) = {}^{0}T_{i}(q) {}^{i}\mathcal{T}_{\mu.i} \qquad i = 1, ..., n \tag{3.21} $$

$$ \mathcal{T} = \begin{bmatrix} \mathcal{R} & \mathcal{P} \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{n} & \mathcal{R}_{a} & \mathcal{R}_{o} & \mathcal{P} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.22} $$

The variances are applied to this coordinate frame of reference. As both consider translational variance and are taken with respect to the same coordinate frame, the total variance, $\boldsymbol{\sigma}_{i}$, at a joint is a sum of the variances for each uncertainty, $\boldsymbol{\sigma}_{E.i}$ and $\boldsymbol{\sigma}_{h.i}$ (Equation 3.23). The variance represents the probabilistic spread of the distance a joint's position will lie from its mean position; however, for collision avoidance this thesis does not use the probabilistic approach. Instead the variance is used to determine the maximum bounds of the uncertainty. This maximum bound can be represented using an ellipsoid. Here the ellipsoid's centre point would be parameterised by the coordinate frame of reference, ${}^{0}\mathcal{P}_{i}$, and radii along each axis by the maximum bounds.

$$\boldsymbol{\sigma}_i = \boldsymbol{\sigma}_{E.i} + \boldsymbol{\sigma}_{h.i} \tag{3.23}$$

Figure 3.11 shows a kinematic model of the multi-link serial robot. Figures 3.11a and 3.11b show the effect on the deterministic kinematic model of the structural and hand position uncertainties respectively. Figure 3.11c shows the coordinate frame of reference with both structural and hand position uncertainties. Figure 3.11d shows the variance at the mean coordinate frame of reference of the joints of a multi-link serial robot.

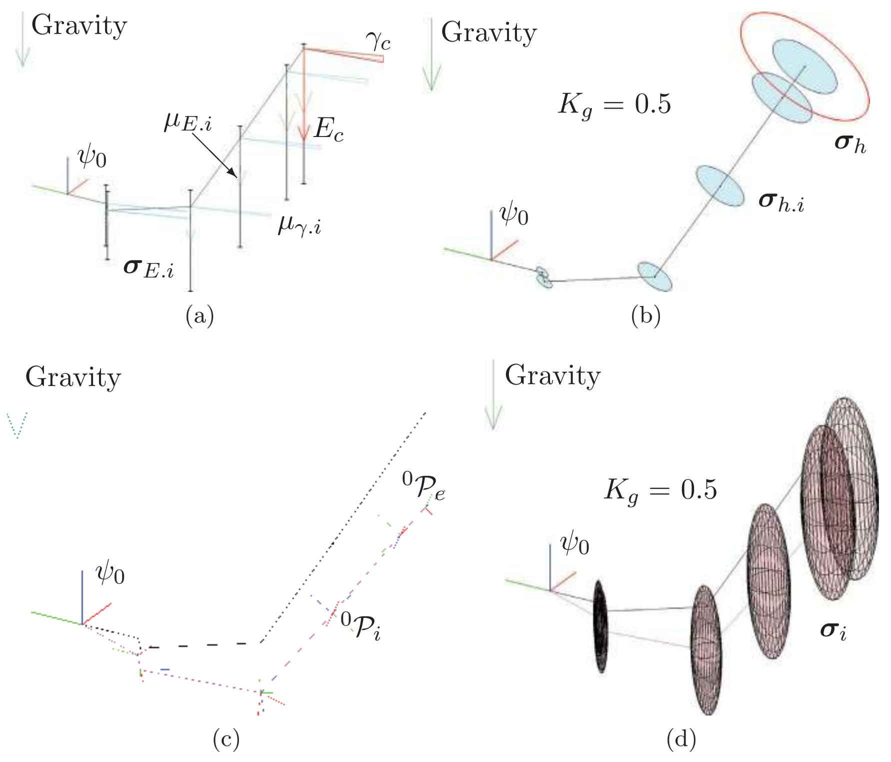


FIGURE 3.11: Visualisation of the (a) structural uncertainty and (b) the hand position uncertainty at a growth factor of 0.5. (c) The coordinate frame resulting from structural uncertainty. (d) The maximum variance represented as ellipsoids at the model's coordinate frame of reference.

3.5 Parameter Verification

This model is reliant on a number of main parameters; the allowed maximum mean and variance for both the structural and hand position uncertainties. These parameters were verified through a number of laboratory and site trials.

3.5.1 Structural Uncertainty Parameters

The values for the allowed maximum mean and variance for the structural uncertainty were determined by placing the inchworm robot in a cantilever state on a vertical surface and measuring the deformation. This state generates the highest loads on the adhesion mechanisms. The adhesion mechanism is comprised of a central housing with three permanent magnet housings; two positioned on one side of the central housing and one positioned on the opposite side.

The allowed maximum variance is measured with the base's adhesion system orientated at three discrete 90° orientations on a vertical plane. First case has the pad orientated with the adhesion mechanism aligned with two toes aligned towards the positive z-direction (Figure 3.12a); second case has the two toes aligned towards the negative z-direction (Figure 3.13a); and the final case has the toes aligned perpendicularly to the z-direction (Figure 3.14a). Figures 3.12b, 3.13b and 3.14b show the deformed inchworm robot superimposed on a non-deformed inchworm robot image for each case respectively. This non-deformed image was captured with the inchworm robot in an upright state. Table 3.1 shows the measured deformation for each case. The allowed maximum translational mean is shown to be 65mm, while the allowed maximum rotational mean is shown to be 8.5°. Both allowed maximum values were found in case 2 with two toes aligned in the negative z-direction. The allowed maximum translational variance was found by taking the difference between the highest and lowest measured translational deformations. This value was found to be 40mm. The functions relating the allowed maximum translational and rotational deformations and the allowed maximum translational variance to the joint distance ratio, G_c, F_c and J_c, were estimated to be linear (Equations 3.24 to 3.26).

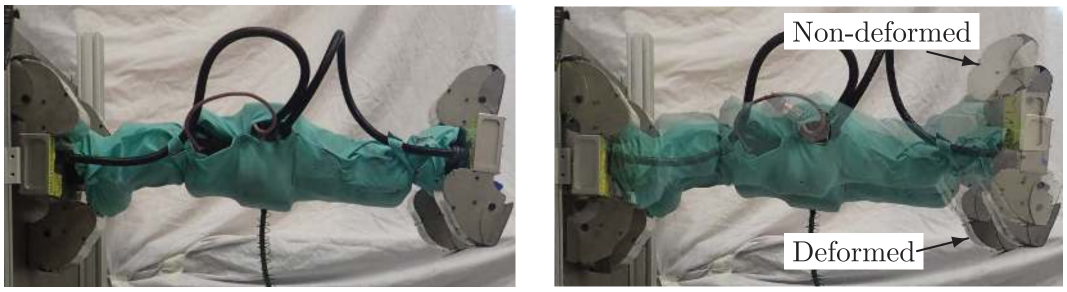
$$G_c(\beta_{c.i}) = \beta_{c.i} E_c \qquad i = 1, .., n \tag{3.24}$$

$$F_c(\beta_{c.i}) = \beta_{c.i} \gamma_c \qquad i = 1, .., n \tag{3.25}$$

$$J_c(\beta_{c.i}) = \beta_{c.i}\omega_E \qquad i = 1, .., n \tag{3.26}$$

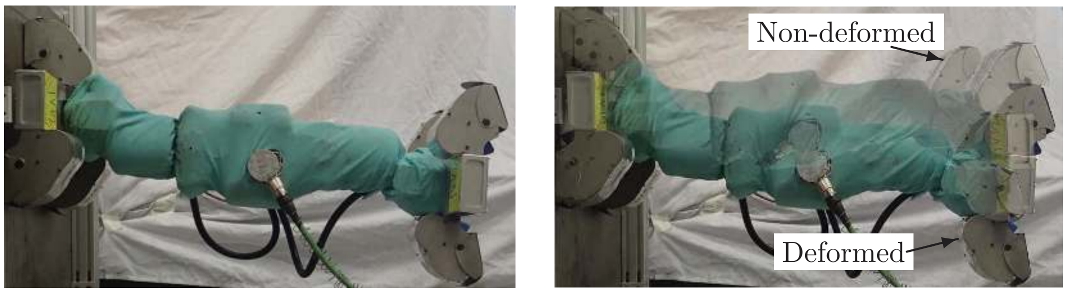
Table 3.1: Measured structural deformation mean and variance for the inchworm robot at difference base orientations

Base pad orientation case	1	2	3
Measure translational deformation (mm)	36	65	25
Measure rotational deformation (°)	6.8	8.5	1.8



(a) (b)

Figure 3.12: The inchworm robot in case 1 (a) deforming due to the weight of the permanent magnet adhesion system orientated with two magnet housings pointing in the positive z-direction and (b) superimposed on a non-deformed inchworm robot. Covers are attached to the robot for protection.



(a) (b)

Figure 3.13: The inchworm robot in case 2 (a) deforming due to the weight of the permanent magnet adhesion system orientated with the two magnet housings pointing in the negative z-direction and (b) superimposed on a non-deformed inchworm robot. Covers are attached to the robot for protection.

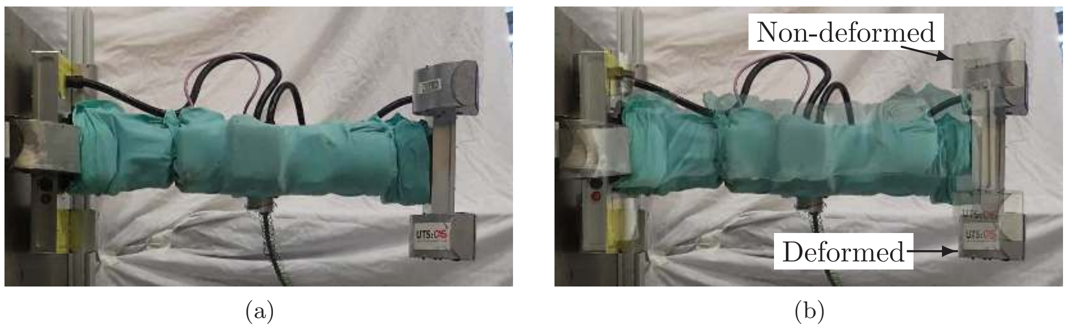


FIGURE 3.14: The inchworm robot in case 3 (a) deforming due to the weight of the permanent magnet adhesion system orientated with the magnet housing orientated perpendicular to the z-axis and (b) superimposed on a non-deformed inchworm robot. Covers are attached to the robot for protection.

3.5.2 Hand Position Uncertainty Parameters

The value for the allowed maximum variance for the hand position uncertainty was determined by measuring the variation between the position of the hand pad prior to and after attachment. This was performed in three states; first with the base positioned on a horizontal floor plane and attaching to a goal location on the same plane; second with the base positioned on a horizontal floor plane attaching to a goal location on a vertical wall plane; and finally with the base positioned on a vertical plane attaching to a horizontal ceiling plane.

Figures 3.15a to 3.15c show the variance from each initial state. Table 3.2 collates the measured variances with the maximum variance measured to be 38mm. While these states give an indication of the allowed maximum variance in laboratory conditions, this is not an exhaustive experiment. Instead based on the measured variance and empirical data from site trials a factor of safety was applied. The final allowed maximum variance for the inchworm robot's hand position variance is 50mm. While only a total of three laboratory trials were performed, the values measured were indicative of measured values found during site trials.

Table 3.2: Measured hand position variance for the inchworm robot at difference hand position attachment surfaces

Hand pad attachment surface	Floor	Wall	Ceiling
Measured variance (mm)	2	5	38

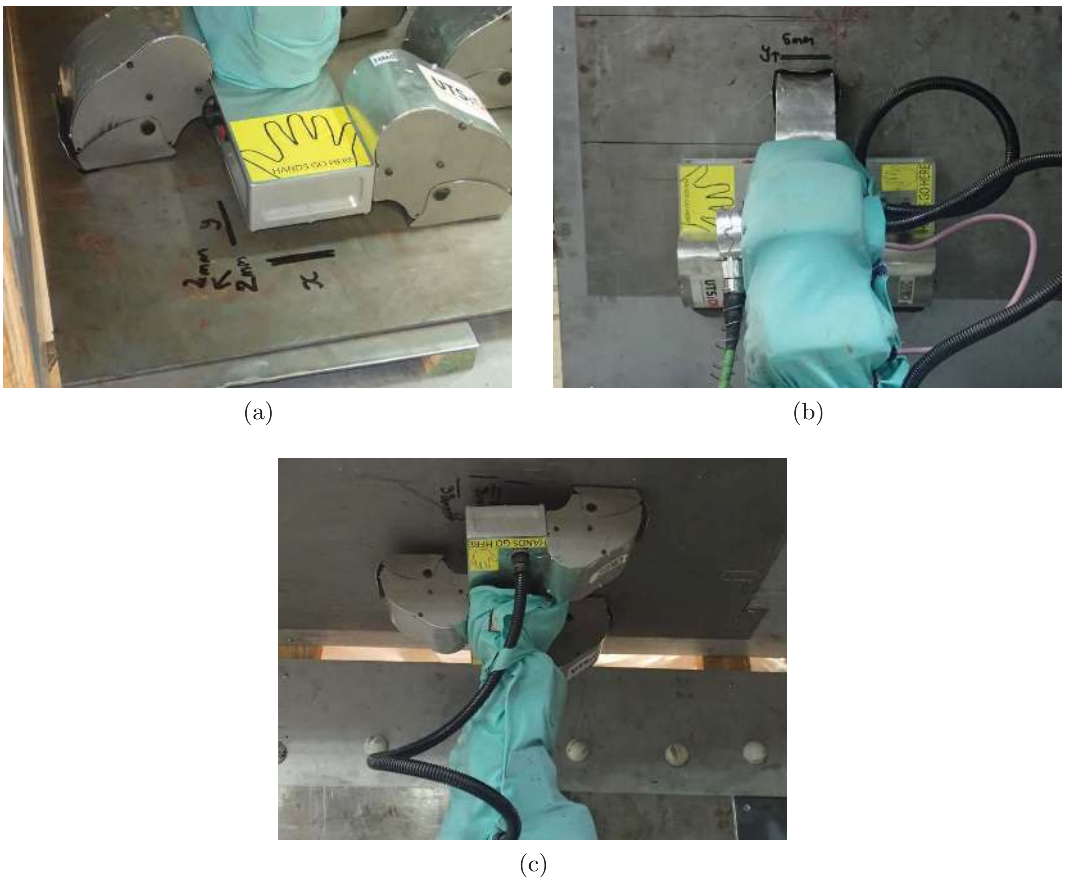


Figure 3.15: The hand position variance due to the permanent magnets in the adhesion system with the hand pad attaching to (a) the floor, (b) the wall and (c) the ceiling (pad detached from the surface for clarity).

3.6 Discussion

The presented model describes the structural and hand position uncertainties of a multi-link serial robot. The uncertainties are described relative to the coordinate frame of reference of the joints and are represented with both mean and variance. The mean is used to determine the most likely robot position relative to the deterministic coordinate frame of reference while the variance represents the probabilistic spread of the distance a joint's position will lie from its mean position. A concern may arise from combining the variance representing different types of uncertainty as the final probabilistic spread is difficult to calculate especially in 3D and when different variances are transformed from different coordinate frames of reference. In this thesis exact knowledge of the probabilistic spread is not required. Instead the variance is used to determine the maximum bounds of the uncertainty which is used for collision avoidance.

The model uses interpolation to determine the mean and variance values of the uncertainty. Using interpolation decouples the need for forward kinematic calculations to determine the uncertainty. This provides a quick method for determining the effect of the uncertainty at any point along the robot without relying on the position of any previous point. However, this method is then reliant on the functions used to interpolate the mean and variance. The curve should then be generated by considering the worst case state. For the inchworm robot, the empirical estimates for the uncertainty in the worst case state were found from multiple site and laboratory trials.

While only a limited number of laboratory trials were performed the measured parameters for the uncertainties were indicative of those found during other laboratory and site trials. However, with limited understanding of the uncertainties which affect the physical state of the robot, the validity of the measured parameters of the uncertainties may not be guaranteed. This may be the case if the poses which result in the highest and lowest deformations were not known or incorrectly identified.

Provided these poses are identified, the uncertainty in the system itself, other than those which affect the physical state of the robot, would not have a large effect on the measured parameters of the uncertainties as comparisons are only made against static instances of the robot state. Also, by considering the static instances of the robot state, the parameters of the uncertainties can more readily be measured with a greater degree of repeatability. As for the uncertainties that do affect the physical state, this method precludes a need to understanding their exact affect on the robot once the poses have been identified.

3.7 Conclusions

This chapter described a method for representing the uncertainty in a multi-link serial robot specifically considering the structural and hand position uncertainties. The effect of these uncertainties results in a model which describes the mean joint locations as a variation from the deterministic model. From these mean joint locations, matrices are used to represent the maximum variance in possible robot locations. While this model only provides a conservative estimate of the uncertainty, it is suitable for this study. The following chapters use this model to perform path planning for a multi-link serial robot.

Chapter 4

Path Planning with Structural and Hand Position Uncertainty

Chapter 3 described a kinematic model to represent multi-link serial robots with structural and hand position uncertainty. This model is incorporated into a 3D probabilistic force field (3D-PF^2) algorithm in this Chapter to develop smooth, collision-free paths. The effect of the uncertainty on the position of a joint is taken into account in an ellipsoid surrounding the link and the joint. Based on this ellipsoid, a force field is defined for motion planning and collision avoidance. A 3D-PF^2 algorithm is then developed based on the 3D force field (3D-F^2) algorithm [155]. This chapter first details the 3D-F^2 algorithm. This is followed by the 3D-PF^2 algorithm including the modifications required to incorporate the model with structural and hand position uncertainty. The algorithm is then verified through simulations and experiments.

The outline of the chapter is as follows: the 3D-F^2 algorithm is defined; the 3D-PF^2 algorithm is described which incorporates the structural and hand position uncertainties; and simulations and methods for verification of the 3D-PF^2 algorithm are explained.

4.1 Three-Dimensional Force Field (3D-F^2) Algorithm

A 3D-F^2 algorithm [155] was developed for finding smooth, collision-free paths for a multi-link serial robot. This algorithm uses a combination of attractive and repulsive forces to guide the end-effector towards a goal position while simultaneously moving away from obstacles within the environment. This section has been included to ensure the thesis is self-contained with pertinent information extracted from Chotiprayanakul et al. [155].

The basic flow of the 3D-F^2 algorithm is detailed in Figure 4.1 and Algorithm 1. First the attractive force and repulsive forces acting on the multi-link serial robot are calculated. The attractive force is calculated based on the difference between the current end-effector location and the goal location. The repulsive forces are based on the force fields surrounding the links and joints at the current pose; if an obstruction enters a force field a virtual force is generated. Obstructions may occur from parts of the environment or parts of the robot itself. Both attractive and repulsive forces are expressed as a magnitude and an application point on the robot. These forces are transformed through a force control algorithm to joint torques which are applied to the robot's dynamic model to change the pose. The 3D-F^2 algorithm continues until the goal location is reached or the algorithm fails.

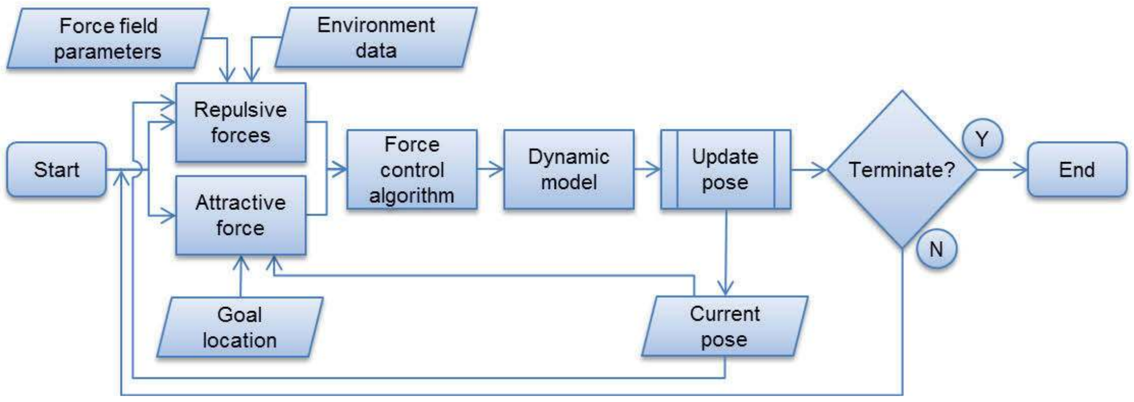


FIGURE 4.1: Basic 3D-F^2 Algorithm flowchart.

Algorithm 1: 3D-F^2 Algorithm

```
Input: $Goal \longleftarrow$ Goal location, $Pose \longleftarrow$ Current pose
Output: $Path \longleftarrow$ Final path

while $EndEffector \neq Goal$ do                    ▷ End-effector not at the goal location
    $(AF, AP) \leftarrow AttractiveForce(Pose, Goal)$          ▷ Find attractive force and point (Section 4.1.4)
    $(RF, RP) \leftarrow RepulsiveForces(Pose)$     ▷ Find repulsive forces and points (Section 4.1.2 and 4.1.3)
    $Torques \leftarrow ForceController(AF, AP, RF, RP)$  ▷ Transform forces to joint torques (Section 4.1.5)
    $Pose \leftarrow Dynamics(Torques)$                  ▷ Update the pose (Eq. 4.12)
    $Path \mathrel{+}= Pose$                              ▷ Update the path
end while
return $Path$                                         ▷ Successful
```

4.1.1 Force Fields

The 3D-F^2 algorithm employs force fields to prevent collisions with possible obstructions. Each link of the multi-link serial robot is surrounded by a force field which fully envelops the physical volume of the link. A force field is described as an ellipsoid parameterised with both a centre point and radius in 3D with respect to the link's coordinate frame of reference (Figure 4.2). When an obstruction enters a force field a repulsive force is generated which pushes the link away.

4.1.2 Repulsive Forces Due to Potential Collision with the Environment

Repulsive forces can be used to prevent the multi-link serial robot from colliding with the environment. In this algorithm, obstacles may be represented as a series of voxels. Repulsive forces are generated when a voxel enters within range of a force field surrounding a link. The further a voxel penetrates a force field, the higher the repulsive force generated.

A repulsive force is generated for each robot link, $\mathbf{F}_{repE_i}$ (Figure 4.3). The shortest distance, C_x, between the closest voxel, $\mathbf{P}_{obs}$, to a point on the link, $\mathbf{P}_{cls}$, is calculated. If this distance is less than an ellipsoid coverage constant, K_P, plus a distance factor, K_{de}, then a repulsive force, $\mathbf{f}_{repE_i}$, is calculated based on a sigmoid function (Equation 4.1).

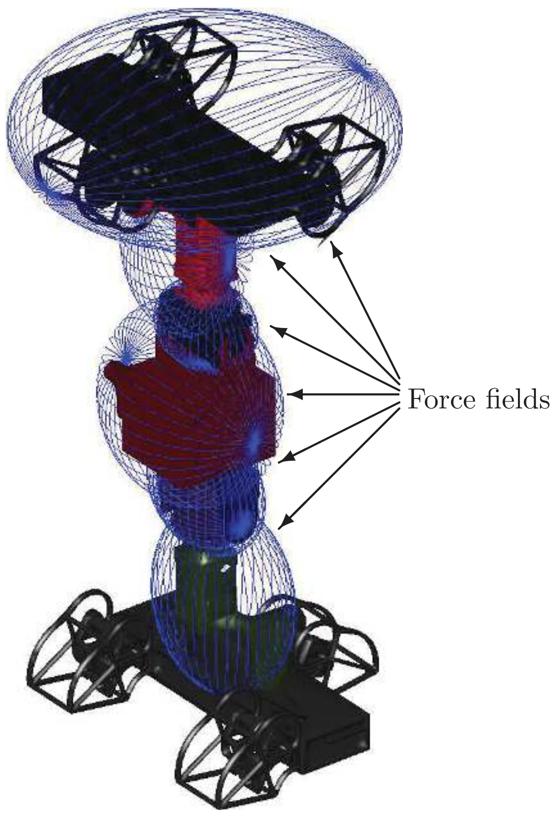


Figure 4.2: Blue ellipsoids surrounding links of the multi-link serial robot to represent the force fields.

Here K_P is a constant greater than 1 and K_f is the coefficient of amplitude. The final repulsive force, $\mathbf{F}_{repE_i}$, only considers translational repulsive forces (Equation 4.2).

$$\mathbf{f}_{repE_i} = K_f - K_f / \left(1 + \exp \left(-10 \frac{Cx - K_P - 0.5K_{de}}{K_{de}} \right) \right) \frac{\mathbf{P}_{cls} - \mathbf{P}_{obs}}{\|\mathbf{P}_{cls} - \mathbf{P}_{obs}\|} \tag{4.1}$$

$$\mathbf{F}_{repE_i} = \begin{bmatrix} 0_{3\times 1} \\ \mathbf{f}_{repE_i} \end{bmatrix} \tag{4.2}$$

4.1.3 Repulsive Forces Due to Potential Self-Collision

Repulsive forces can also be used to prevent the multi-link serial robot from colliding with itself. Once the distance between two links is significantly small a repulsive force is generated. The repulsive force increases as the distance between two links decreases.

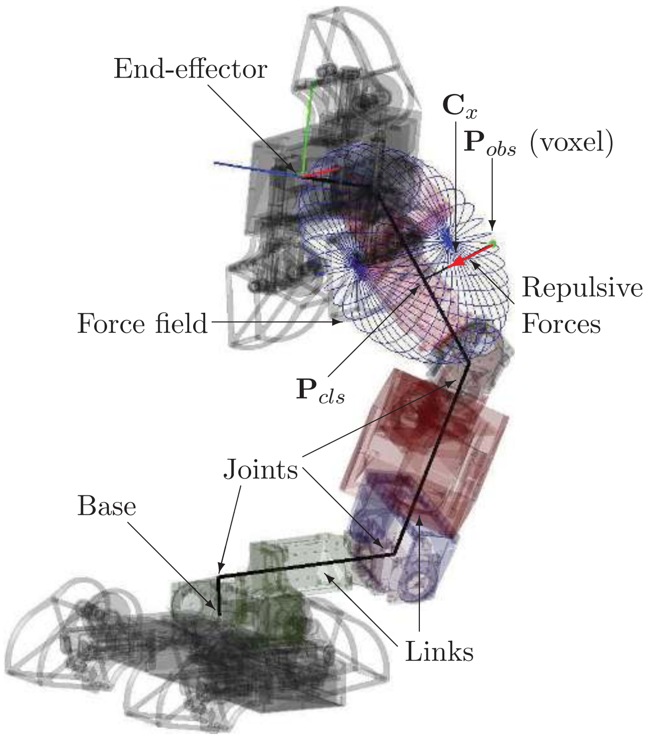


FIGURE 4.3: A repulsive force acting on an inchworm robot.

The self-collision repulsive force is calculated based on the distance, d_0, between the closest points on a link i, ${}^0\mathbf{P}_{i.cls}$, and a preceding link j, ${}^0\mathbf{P}_{j.cls}$. As the link directly preceding link i, that is link $i-1$, is in direct contact it is excluded from the calculation. If this distance, d_0, is within a distance factor, K_{ds}, a sigmoid function is used to calculate the self-collision repulsive force (Equation 4.3). This force is applied to the closest point, $\mathbf{P}_{i.cls}$, on the link i. Here K_f is the coefficient of amplitude. The final repulsive force, $\mathbf{F}_{repE_i}$, only considers translational repulsive forces not rotational forces (Equation 4.4). This method defines force fields using a swept-sphere method which differs from the ellipsoid definition for force fields used to determine the repulsive forces due to the environment.

$$\mathbf{f}_{repS_i} = K_f - K_f / \left(1 + \exp\left(-10 \frac{d_0 - 0.5K_{ds}}{K_{ds}} \right) \right) \frac{{}^0\mathbf{P}_{i.cls} - {}^0\mathbf{P}_{j.cls}}{\|{}^0\mathbf{P}_{i.cls} - {}^0\mathbf{P}_{j.cls}\|} \tag{4.3}$$

$$\mathbf{F}_{repS_i} = \begin{bmatrix} 0_{3\times1} \\ \mathbf{f}_{repS_i} \end{bmatrix} \tag{4.4}$$

4.1.4 Attractive Force

The attractive force is used to guide the multi-link serial robot's end-effector to a goal location. The end-effector's position, ${}^{0}\mathbf{P}_e$, and orientation component of its rotation matrix, ${}^{0}\mathbf{R}_{e.o}$, are aligned with the goal location which is comprised of a goal position, $\mathbf{P}_g$ and rotational vector, $\mathbf{R}_g$ (Figure 4.4). The goal location can either be user supplied or determined through additional methods.

The attractive force is calculated by determining the differences between the end-effector and goal positions and normals and applying a sigmoid function to the resultant (Equations 4.5 and 4.6). Here $K_{att.R}$ and $K_{att.T}$ are the maximum rotational and translational attractive forces, K_{zero} is a small non-zero positive constant and K_s is a constant which determines how the attractive force varies over distance. The position component, $\mathbf{f}_{att}$, and the rotation component, $\boldsymbol{\omega}_{att}$, are appended to determine the attractive force, $\mathbf{F}_{att}$ (Equation 4.7). This force is applied at the end-effector (Equation 4.8).

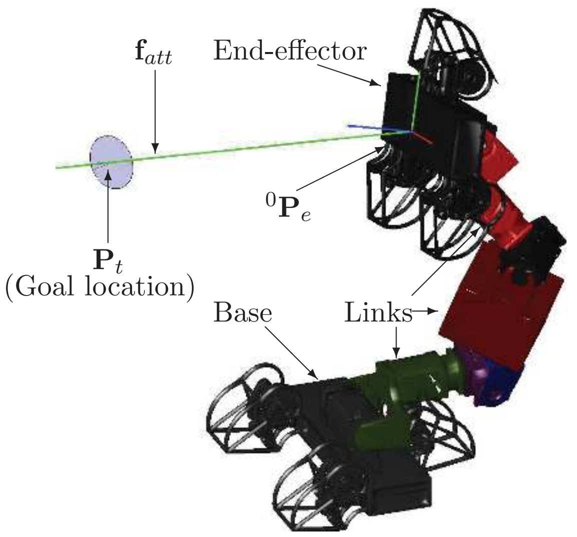


Figure 4.4: The attractive force acting on the inchworm robot.

$$\mathbf{f}_{att} = K_{att.T} \Big/ \left(1 + \frac{\exp\left(-K_s \|\mathbf{P}_t - {}^{0}\mathbf{P}_e\|\right)}{K_{zero}}\right) \frac{\mathbf{P}_t - {}^{0}\mathbf{P}_e}{\|\mathbf{P}_t - {}^{0}\mathbf{P}_e\|} \tag{4.5}$$

$$\boldsymbol{\omega}_{att} = K_{att.R} / \left(1 + \frac{\exp\left(-K_s \sin^{-1}((\|{}^0\mathbf{R}_{e.o} \cdot \mathbf{R}_g\|)/(\|{}^0\mathbf{R}_{e.o}\| \cdot \|\mathbf{R}_g\|))\right)}{K_{zero}} \right) \frac{{}^0\mathbf{R}_{e.o} \cdot \mathbf{R}_g}{\|{}^0\mathbf{R}_{e.o} \cdot \mathbf{R}_g\|} \tag{4.6}$$

$$\mathbf{F}_{att} = \begin{bmatrix} \boldsymbol{\omega}_{att} \\ \mathbf{f}_{att} \end{bmatrix} \tag{4.7}$$

$${}^0\mathbf{P}_{att_n} = {}^0\mathbf{P}_e \tag{4.8}$$

4.1.5 Dynamic Model

The dynamic model of a multi-link serial robot is given in Equation 4.9. In this research the dynamic model is used to calculate equivalent joint torques based on the applied attractive and repulsive forces and is used to iteratively calculate the next pose in the path. Here $\mathbf{I}$ is the link inertia matrix, $\boldsymbol{\beta}$ is the joint damping-friction constant matrix, $\mathbf{k}_{sp}$ is the joint spring constant matrix and $\ddot{\mathbf{q}}$, $\dot{\mathbf{q}}$, $\mathbf{q}$ are the joint accelerations, velocities and positions respectively. $\boldsymbol{\Gamma}$ is the torque-force matrix which contains the joint torque components, τ_i, for each joint, i (Equation 4.10). The values of $\mathbf{I}$, $\boldsymbol{\beta}$ and $\mathbf{k}_{sp}$ are based on the design of the robot.

$$\boldsymbol{\Gamma} = \mathbf{I}\ddot{\mathbf{q}} + \boldsymbol{\beta}\dot{\mathbf{q}} + \mathbf{k}_{sp}\mathbf{q} \tag{4.9}$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} \tau_1, & \tau_2, & ..., & \tau_n \end{bmatrix}^T \tag{4.10}$$

The torque-force matrix for the multi-link serial robot is calculated by combining the torque-force matrices generated from the attractive force, $\mathbf{F}_{att}$, and the repulsive forces, $\mathbf{F}_{rep}$ (Equation 4.11). Equation 4.11 is substituted into Equation 4.9 to form Equation 4.12.

$$\mathbf{\Gamma} = \mathbf{\Gamma}_{att} + \sum_{i=1}^{n} \mathbf{\Gamma}_{rep_i} \tag{4.11}$$

$$\mathbf{\Gamma}_{att} + \sum_{i=1}^{n} \mathbf{\Gamma}_{rep_i} = \mathbf{I}\ddot{\mathbf{q}} + \boldsymbol{\beta}\dot{\mathbf{q}} + \mathbf{k}_{sp}\mathbf{q} \tag{4.12}$$

Assuming the velocity, $\dot{\mathbf{q}}$, and acceleration, $\ddot{\mathbf{q}}$, are initially zero, the 3D-F^2 algorithm iteratively solves Equation 4.12 for the pose, $\mathbf{q}$. This process continues until the goal location has been reached. The resulting series of poses is the collision-free path.

4.1.6 Force Control Algorithm

The torque-force matrix, $\mathbf{\Gamma}$, defines an equivalent torque resulting from transforming a force, $\mathbf{F}$, from Cartesian space to joint space using a transformation matrix, $\mathbf{H}(\mathbf{q}, {}^0\mathbf{P}_f)$ (Equation 4.13). Here $\mathbf{P}_f$ is the position of the force in Cartesian space, $\mathbf{L}_j$ is a matrix representing the axis of rotation of a joint and ${}^i\mathbf{M}_0$ is a rotation skew-symmetric matrix.

$$\mathbf{\Gamma} = \mathbf{H}_i(\mathbf{q}, {}^0\mathbf{P}_f)\mathbf{F} \tag{4.13}$$

$$\mathbf{H}_i(\mathbf{q}, {}^0\mathbf{P}_f) = \begin{bmatrix} \mathbf{h}_j, & ..., & \mathbf{h}_n \end{bmatrix}^T \qquad j = 1, ..., n \tag{4.14}$$

$$\mathbf{h}_j = \begin{cases} 0_{1\times 6} & \text{if i > j} \\ \mathbf{L}_j{}^i\mathbf{M}_0 & \text{if i} \leq \text{j} \end{cases} \qquad j = 1, ..., n \tag{4.15}$$

$$\mathbf{L}_j = \begin{bmatrix} l_{r.x_j} & l_{r.y_j} & l_{r.z_j} & l_{t.x_j} & l_{t.y_j} & l_{t.z_j} \end{bmatrix} \tag{4.16}$$

$$l_{r.k_j} = \begin{cases} 1 & \text{if joint } j \text{ is rotational about axis } r \\ 0 & \text{otherwise} \end{cases} \qquad k \in \{x, y, z\} \tag{4.17}$$

$$l_{t.k_j} = \begin{cases} 1 & \text{if joint } j \text{ is translational along axis } r \\ 0 & \text{otherwise} \end{cases} \qquad k \in \{x, y, z\} \tag{4.18}$$

The rotation skew-symmetric matrix, ${}^{i}\mathbf{M}_0$, is used to transfer a force in the global coordinate frame of reference to the coordinate frame of reference of joint i (Equation 4.19). Here ${}^{0}\mathbf{R}_i$ is the rotation matrix of joint i, and $\mathbf{S}(\Delta\mathbf{P})$ is a skew-symmetric matrix based on the variation between the position of the coordinate frame of reference, ${}^{0}\mathbf{P}_i$, and the position of the force, ${}^{0}\mathbf{P}_f$.

$${}^{i}\mathbf{M}_0 = \begin{bmatrix} {}^{0}\mathbf{R}_i^T & 0_{3\times3} \\ 0_{3\times3} & {}^{0}\mathbf{R}_i^T \end{bmatrix} \begin{bmatrix} 1_{3\times3} & 0_{3\times3} \\ -\mathbf{S}(\Delta\mathbf{P}_i) & 1_{3\times3} \end{bmatrix} \tag{4.19}$$

$$\mathbf{S}(\Delta\mathbf{P}_i) = \begin{bmatrix} 0 & -\Delta P_z & \Delta P_y \\ \Delta P_z & 0 & -\Delta P_x \\ -\Delta P_y & \Delta P_x & 0 \end{bmatrix} \tag{4.20}$$

$$\Delta\mathbf{P}_i = \begin{bmatrix} \Delta P_x \\ \Delta P_y \\ \Delta P_z \end{bmatrix} =^0 \mathbf{P}_f -^0 \mathbf{P}_i \tag{4.21}$$

The dynamic model (Equation 4.12) is rewritten to include the force-torque matrices for the attractive and repulsive forces (Equation 4.22).

$$\mathbf{H}(\mathbf{q}_{t-1}, {}^{0}\mathbf{P}_{att_n})_n\mathbf{F}_{att} + \sum_{i=1}^{n} \mathbf{H}(\mathbf{q}_{t-1}, {}^{0}\mathbf{P}_{rep_i})_i\mathbf{F}_{rep_i} = \mathbf{I}\ddot{\mathbf{q}} + \boldsymbol{\beta}\dot{\mathbf{q}} + \mathbf{k}_{sp}\mathbf{q} \tag{4.22}$$

4.2 3-Dimensional Probabilistic Force Field (3D-PF^2) Algorithm

The 3D-PF^2 algorithm is used to find smooth, collision-free paths for a multi-link serial robot with structural and hand position uncertainty. This algorithm builds on the 3D-F^2

algorithm [155].

The 3D-PF^2 algorithm follows the same flow and algorithm of the 3D-F^2 algorithm with three main differences (Figure 4.5 and Algorithm 2). 1) The model of the multi-link serial robot described in Chapter 3 is used in lieu of the deterministic model. This model is used to calculate the attractive and repulsive forces. The force control algorithm and the dynamic model have been updated to reflect this change. 2) The creation of the force fields. In the 3D-F^2 algorithm these were statically sized to encompass the links. In the 3D-PF^2 algorithm the sizes of the force fields also consider the uncertainty in the coordinate frame of reference of the joints. 3) Finally the termination criteria. The 3D-F^2 algorithm does not consider local minima. A method has been developed to allow the 3D-PF^2 algorithm to detect local minima. The 3D-PF^2 algorithm terminates when either the goal location is reached or a local minimum is entered.

4.2.1 Force Fields with Structural and Hand Position Uncertainties

In the 3D-F^2 algorithm presented in Chotiprayanakul et al. [155] force fields are sized to encompass links and joints. Due to the structural and hand position uncertainties deterministically sized force fields are not suitable. Instead force fields should be used which

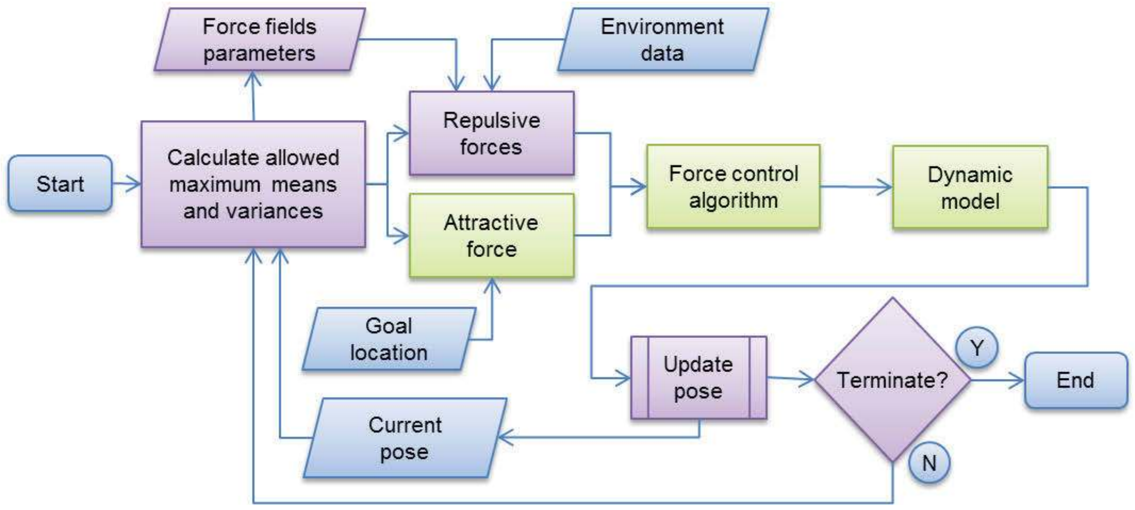


FIGURE 4.5: Basic 3D-PF^2 algorithm flowchart. Green objects are modified based on the model described in Chapter 3, purple have added algorithms and blue are unchanged.

Algorithm 2: 3D-PF^2 Algorithm

Input: *Goal* ⟵ Goal location, *Pose* ⟵ Current pose
Output: *Path* ⟵ Final path

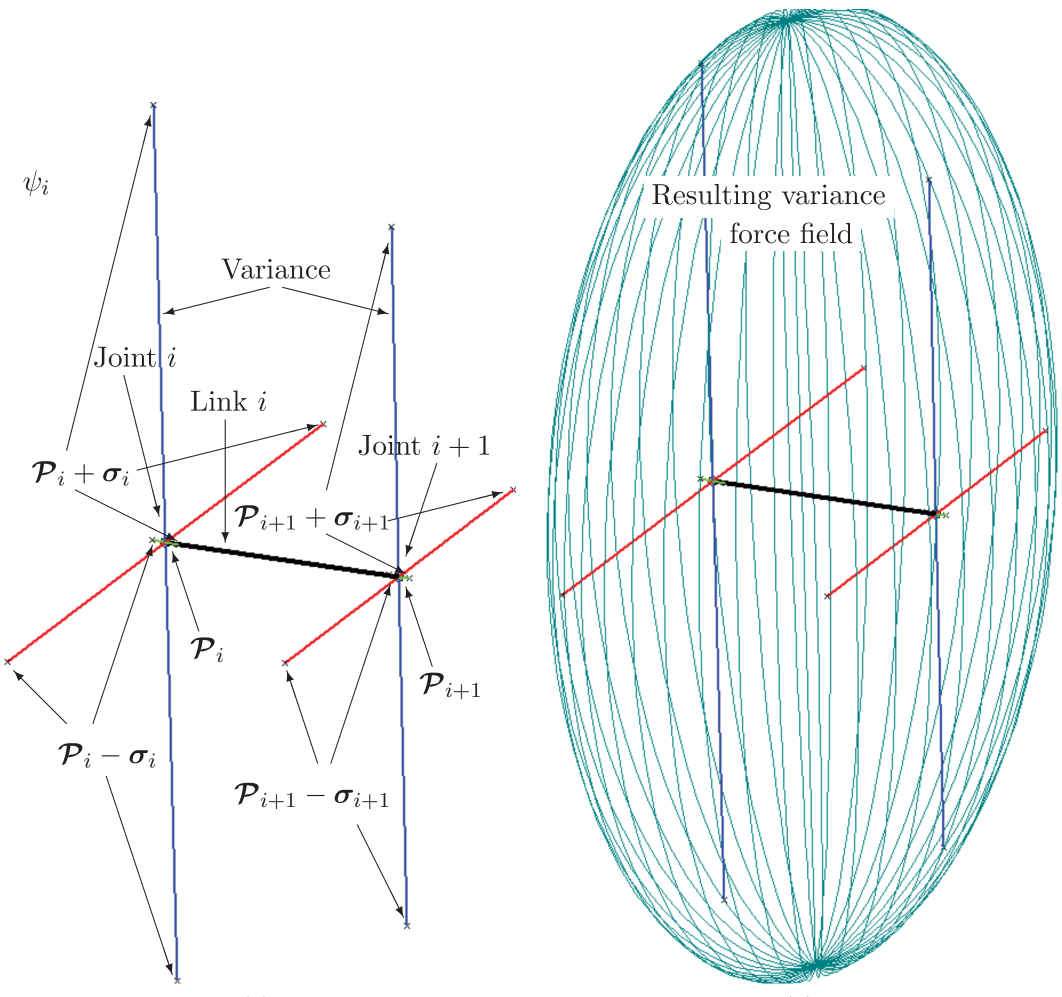
```
while $EndEffector \neq Goal$ do                                   ▷ End-effector not at the goal location
    $Uncert \leftarrow Uncertainty(Pose)$       ▷ Calculate the allowed maximum mean and variance (Section 3.4)
    $UpdateForceField(Uncert, Pose)$            ▷ Update the force fields (Section 4.2.1)
    $(AF, AP) \leftarrow ProbAttractiveForce(Pose, Goal)$    ▷ Find the attractive force and point (Section 4.2.4)
    $(RF, RP) \leftarrow ProbRepulsiveForces(Pose)$    ▷ Find the repulsive forces and points (Section 4.2.2 and 4.2.3)
    $Torques \leftarrow ProbForceController(AF, AP, RF, RP)$    ▷ Find the equivalent torques (Section 4.2.6)
    $Pose \leftarrow ProbDynamics(Torques)$     ▷ Update the pose (Eq. 4.58)
    $Path$ += $Pose$                            ▷ Update the path
    if InLocalMinimaPath then                   ▷ If a local minima is entered (Section 4.2.7)
        return $Path \leftarrow \emptyset$      ▷ The algorithm fails
    end if
end while
return $Path$                                   ▷ Successful
```

Lines highlighted in green are modified based on the model described in Chapter 3 while lines highlighted in purple have added or modified algorithms.

encompasses the variance at each joint. Guan-chen et al. [179] uses a similar technique to account for system uncertainty in an unmanned aerial vehicle.

For a given link, i, the force field needs to be formed to envelop the variance, $\boldsymbol{\sigma}_i$, at its associated coordinate frame of reference, i, and the variance, $\boldsymbol{\sigma}_{i+1}$, at the proceeding coordinate frame of reference, $i+1$ (Figure 4.6). The variance of a joint is described as an ellipsoid with the ellipsoid's major axes describing the maximum allowed variance. By considering the variance at a link's associated joint's positions, $\boldsymbol{\mathcal{P}}_i$ and $\boldsymbol{\mathcal{P}}_{i+1}$, and along the joint's axes provides a series of points which represent the variance for that link, $\boldsymbol{\mathcal{P}}_{\boldsymbol{\sigma}_i}$ (Equation 4.23 and Figure 4.6a). A force field needs to be sized to envelop these points. This is done by using a Khachiyan Algorithm [201] - an optimisation routine which iteratively minimises a convex function - to find a bounding ellipsoid which will encompass these point (Figure 4.6b). While the kinematic model considers a link as a line, this is generally not the case for most multi-link serial robots.

$$\boldsymbol{\mathcal{P}}_{\boldsymbol{\sigma}_i} = \begin{bmatrix} \boldsymbol{\mathcal{P}}_i + \boldsymbol{\sigma}_i & \boldsymbol{\mathcal{P}}_i - \boldsymbol{\sigma}_i & \boldsymbol{\mathcal{P}}_{i+1} + \boldsymbol{\sigma}_{i+1} & \boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\sigma}_{i+1} \end{bmatrix} \tag{4.23}$$


ψ_i
Variance
Joint i
Link i
Joint $i+1$
$\boldsymbol{\mathcal{P}}_i + \boldsymbol{\sigma}_i$
$\boldsymbol{\mathcal{P}}_{i+1} + \boldsymbol{\sigma}_{i+1}$
$\boldsymbol{\mathcal{P}}_i$
$\boldsymbol{\mathcal{P}}_{i+1}$
$\boldsymbol{\mathcal{P}}_i - \boldsymbol{\sigma}_i$
$\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\sigma}_{i+1}$
Resulting variance force field

(a) (b)

FIGURE 4.6: (a) The variance in a link at its coordinate frame of reference, i, and its proceeding coordinate frame of reference, $i+1$ (b) The resulting variance force field surrounding a link to include the variance.

The force field should take into consideration the physical components of a link such as linkages, motors and sensors (Figure 4.7). As previously described, the force field needs to be sized to encompass the variance at the coordinate frame of reference; this needs to be extended to include the link's components. The components of a link can be represented using voxels; for the purposes of this work this can be further simplified by using only the voxels at the extreme points of the components, $\mathbf{V}$, such as at corners (Figure 4.7a). The variance is now considered from these voxels instead of the joint's coordinate frame of reference; however, calculating the exact uncertainty at each voxel would be computationally expensive. Instead it is assumed that the variance at a vertex is proportional to its relative position along the link. For each vertex the closest point on the link defined through the kinematic model - that is the line $\boldsymbol{\mathcal{P}}_i$ to $\boldsymbol{\mathcal{P}}_{i+1}$ - is found, $\mathbf{P}_{\mathbf{V}}$ (Equation 4.24). The variance at a vertex is proportionally found based on the closest point (Equation 4.25). As the variance is described using an ellipsoid with the major axes describing the variance, the major axes of the variance ellipsoid proportionally found for each voxel is added to the voxels position to determine a series of points, $\mathbf{P}_{\boldsymbol{\sigma}_{\mathbf{V}}}$ (Equation 4.26 and Figure 4.7b). A Khachiyan Algorithm [201]is used to find a bounding ellipsoid. This ellipsoid represents the force field which full encompasses the variance in the link (Figure 4.7c).

$$\mathbf{P}_{\mathbf{V}} = \frac{(\mathbf{V} - \boldsymbol{\mathcal{P}}_i) \cdot (\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i)}{\|\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i\|} \tag{4.24}$$

$$\boldsymbol{\sigma}_{\mathbf{V}} = \mathbf{P}_{\mathbf{V}}(\boldsymbol{\sigma}_{i+1} - \boldsymbol{\sigma}_i) + \boldsymbol{\sigma}_{i+1} \tag{4.25}$$

$$\mathbf{P}_{\boldsymbol{\sigma}_{\mathbf{V}}} = \begin{bmatrix} \mathbf{V} + \boldsymbol{\sigma}_{\mathbf{V}} & \mathbf{V} - \boldsymbol{\sigma}_{\mathbf{V}} \end{bmatrix} \tag{4.26}$$

Singular value decomposition is used to extract the force field parameters. This force field - the variance force field, $\hat{\mathbf{D}}_i$ - is parameterised using a 3×1 array describing the centre position, $\mathbf{c}_i$, a 3×1 array describing radius, $\mathbf{r}_i$, and a 3×3 rotation matrix, $\mathbf{R}_{ff.i}$ relative to the coordinate frame of a given link, i.

In the 3D-F^2 algorithm [155] an inner and an outer force field is used to control the amplitude of repulsive forces. The 3D-PF^2 algorithm also employs two force fields; a minimum and a maximum variance force field. These force fields are parameterised with the same centre, $\mathbf{c}_i$, and rotation matrix, $\mathbf{R}_{ff.i}$ as the variance force field, $\hat{\mathbf{D}}_i$; however, their radii are considered as factors of the variance force field's radius, $\mathbf{r}_i$. A minimum and a maximum variance factor, ξ_{min} and ξ_{max}, is used to define the radius of the minimum and the maximum variance force field respectively (Equation 4.27). If an obstacle penetrates

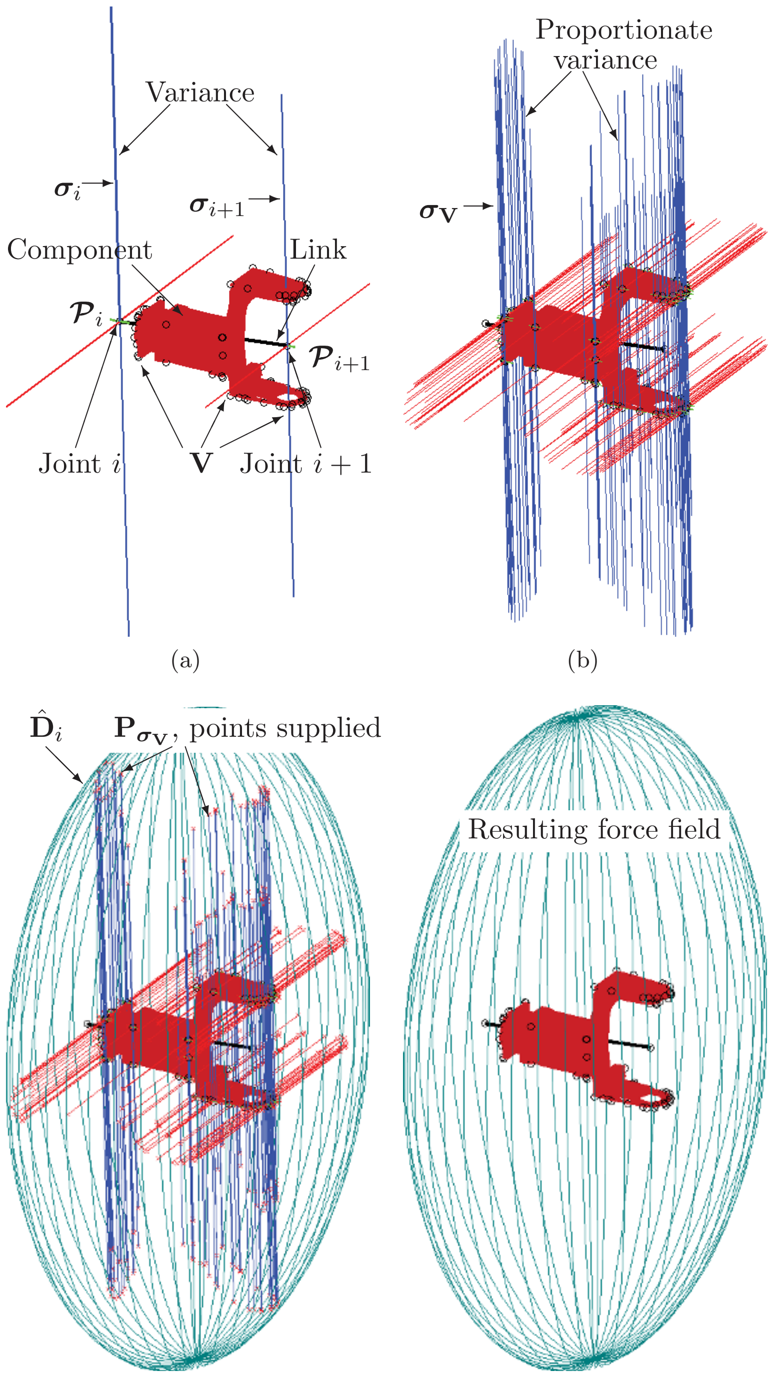

Variance
$\boldsymbol{\sigma}_i$
$\boldsymbol{\sigma}_{i+1}$
Component
Link
$\boldsymbol{\mathcal{P}}_i$
$\boldsymbol{\mathcal{P}}_{i+1}$
Joint i
$\mathbf{V}$
Joint $i+1$
(a)
Proportionate variance
$\boldsymbol{\sigma}_{\mathbf{V}}$
(b)
$\hat{\mathbf{D}}_i$
$\mathbf{P}_{\boldsymbol{\sigma}_{\mathbf{V}}}$, points supplied
(c)
Resulting force field
(d)

FIGURE 4.7: (a) The link components superimposed on the link with the variance at a link's joint and the proceeding joint. Black circles show the link vertices, $\mathbf{V}$ (b) The proportionate variance at each vertex, $\boldsymbol{\sigma}_{\mathbf{V}}$ (c) The resulting force field surrounding the link. The red asterisks at the end of the proportionate variances are the points supplied to the Khachiyan Algorithm. (d) The resulting force field and link shown for clarity.

the minimum variance force field, the chances of a collision with the robot is considered to be significantly high and the 3D-PF^2 algorithm fails. Once the maximum variance force field is penetrated by an obstacle, repulsive forces are generated as there exists a chance that collisions with the robot are possible.

$$0 \leq \xi_{min} < \xi_{max} \tag{4.27}$$

The variance factors should be selected to allow obstacles to be bypassed while ensuring safe motion. Lower values reduce the radius of the force fields and increase the chances of collisions. Higher variance factors will increase the radius of the force fields and reduce the chances of collision; larger force fields may prevent the robot from bypassing certain obstacles. For variance factors less than 1 the type of distribution used to represent the variance will affect the number of possible joint locations encompassed by the force fields; for example at a variance factor of 0.8 the number of possible joint locations encompassed will vary between a Gaussian distribution and a uniform distribution. These concerns should be considered when selecting the variance factors.

In Chapter 3 it was mentioned that while a Gaussian distribution was the most likely representation for the structural and hand position of uncertainties the allowed maximum values are used instead. These values will generate relatively larger force fields allowing conservative, safer navigation solutions to be found.

The variance factor cannot be directly multiplied by the radius of the variance force field as a variance factor of zero would result in a force field with a zero radius. Instead with a variance factor of zero, a force field should be defined which only envelopes the link with zero variance (Figure 4.8a). To determine this first a 3×3 transformation matrix is found which transforms the variance force field, $\hat{\mathbf{D}}_i$, to a unit sphere, $\psi_{\hat{\mathbf{D}}_i}$ (Figure 4.8b). This matrix is applied to a link's component's vertices, $\mathbf{V}$, to determine their positions, $\hat{\mathbf{V}}$, relative to the unit sphere (Equation 4.28). The distance from each vertex to the origin of the unit sphere is calculated and the furtherest distance selected, $\xi_{dist.i}$ (Equation 4.29). As the unit sphere has a radius of 1 and the vertex is within this radius, the distance can be expressed as a ratio which, when applied to the variance force field's radius, defines the minimum force field radius surrounding a link that encompasses all vertices, $r_{min.i}$ (Equation 4.30). The radius of a force field will then be sized based on the difference between the minimum force field radius, $r_{min.i}$, and the variance force field radius, $\mathbf{r}_i$, at a variance factor of 0 and 1 respectively (Figure 4.8c). The minimum variance force field, $\hat{\mathbf{D}}_{i.minV}$, is then described using the minimum variance factor, ξ_{min} (Equation 4.31) while the maximum variance force field, $\hat{\mathbf{D}}_{i.maxV}$, uses the maximum variance factor, ξ_{max} (Equation 4.32). The variance force fields for a given link are visualised in Figure 4.9.

$$\hat{\mathbf{V}} = \mathbf{V} \cdot \psi_{\hat{\mathbf{D}}_i} \tag{4.28}$$

$$\xi_{dist.i} = \max(\|\hat{\mathbf{V}}\|) \tag{4.29}$$

$$\mathbf{r}_{min.i} = \xi_{dist.i}\mathbf{r}_i \tag{4.30}$$

$$\mathbf{r}_{minV.i} = \xi_{min}(\mathbf{r}_i - \mathbf{r}_{min.i}) + \mathbf{r}_{min.i} \tag{4.31}$$

$$\mathbf{r}_{maxV.i} = \xi_{max}(\mathbf{r}_i - \mathbf{r}_{min.i}) + \mathbf{r}_{min.i} \tag{4.32}$$

4.2.2 Repulsive Force Due to Potential Collisions with the Environment

The repulsive force can be used to prevent collisions with the environment. The 3D-PF^2 algorithm cannot use a direct implementation of the environmental repulsive force algorithm used in the 3D-F^2 algorithm for a number of reasons. First the 3D-F^2 algorithm parameterised the force fields without a rotation component. The radius of the force field is defined relative to the axes of the coordinate frame of reference of the joint and link. Without a rotation component, optimal coverage of the links can not be achieved. Secondly the distance between the inner and outer force fields was constant. The position of the

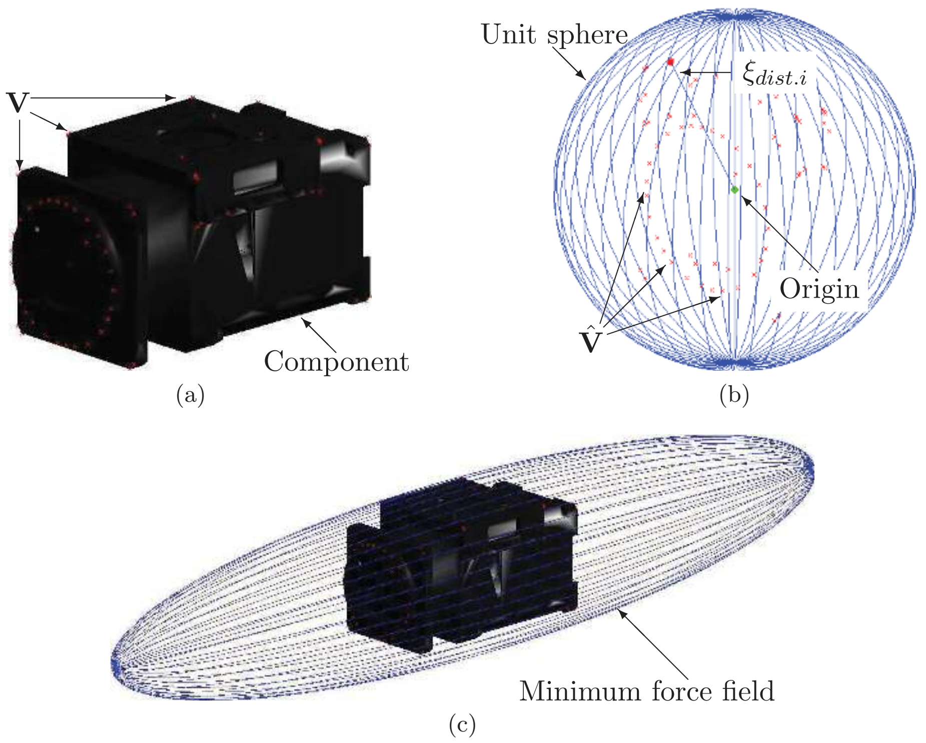


FIGURE 4.8: (a) A link's component with vertices shown as red asterisks. (b) The vertices transformed relative to the unit sphere shown as red asterisks. The furthest vertex from the origin is found. (c) The final minimum force field encompassing the link's component.

obstruction relative to the two force fields has no bearing on the amplitude of the repulsive force; that is, the same distance an obstruction penetrates the force field will result in the same repulsive force amplitude irrespective of the position. If the distance between the two force fields was not constant additional calculations would be necessary. Finally only the closest voxel to the link was considered, which may be a concern when force fields were penetrated from multiple directions. The repulsive force algorithm used in the 3D-PF^2 algorithm addresses these concerns for the multi-link serial robot with uncertainty.

The magnitude of the repulsive force is based on the distance a voxel penetrates between two force fields irrespective of the relative position. As the difference between the inner and outer force fields was constant it was relatively straightforward to determine this penetration distance. With the minimum and maximum variance force field, this can no longer be directly applied as the difference between the radius of the variance force fields may vary along each major axis. Instead the variance force fields first need to be converted

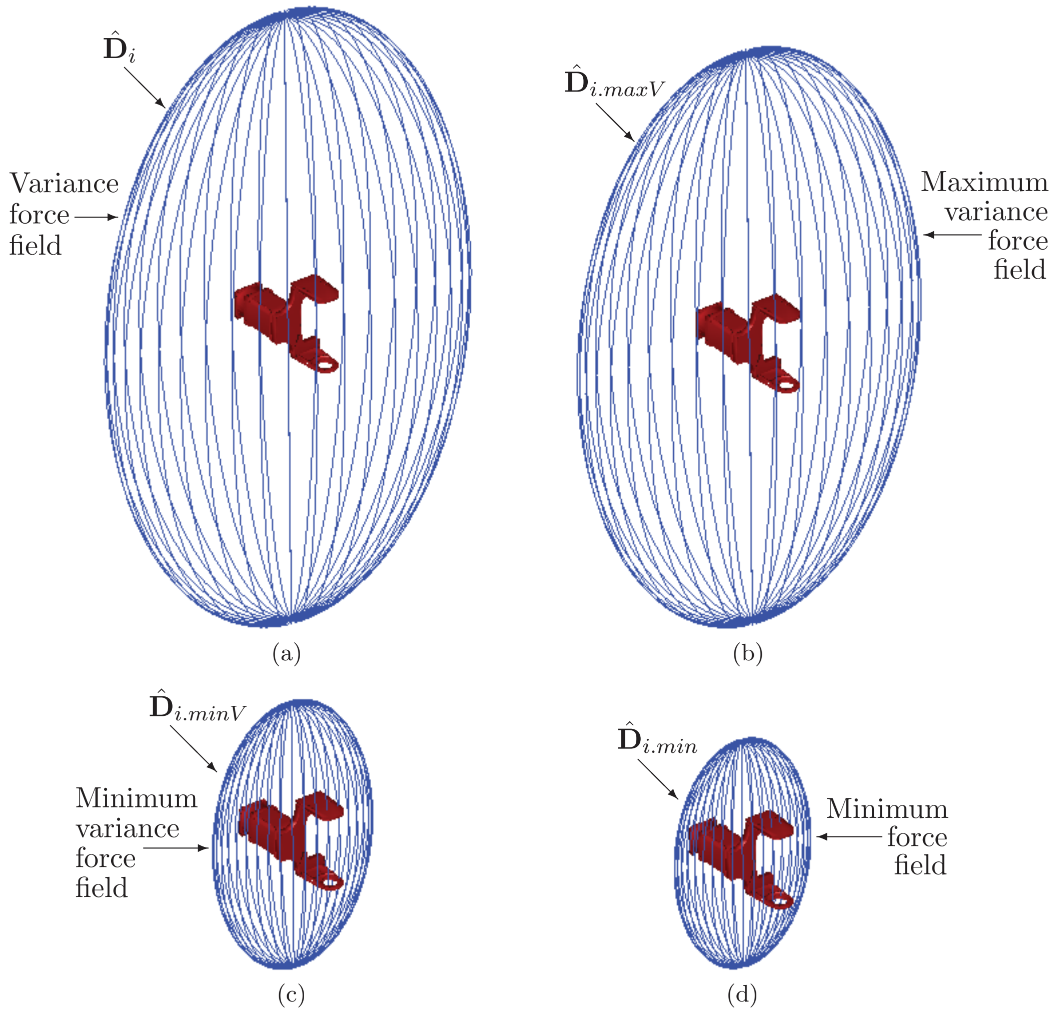

$\hat{\mathbf{D}}_i$
Variance force field
(a)
$\hat{\mathbf{D}}_{i.maxV}$
Maximum variance force field
(b)
$\hat{\mathbf{D}}_{i.minV}$
Minimum variance force field
(c)
$\hat{\mathbf{D}}_{i.min}$
Minimum force field
(d)

FIGURE 4.9: (a) The variance force field, $\hat{\mathbf{D}}_i$. (b) The maximum variance force field, $\hat{\mathbf{D}}_{i.maxV}$, at a maximum variance factor of 90%. (c) The minimum variance force field, $\hat{\mathbf{D}}_{i.minV}$ at a minimum variance factor of 10%. (d) The minimum force field, $\hat{\mathbf{D}}_{i.min}$

to spheres which allows a constant distance between the two to be established. This is possible as both variance force fields have the same centre and rotation.

Let $\mathbf{P}_{obs}$ be the voxels which comprise the environment. For a given link, i, and minimum variance force field, $\hat{\mathbf{D}}_{i.minV}$, a 3×3 transformation matrix, ψ_{E_i}, is found which transforms the minimum variance force field to a unit sphere. Applying the same transformation, ψ_{E_i}, to the maximum variance force field will find a maximum variance sphere relative to the minimum variance unit sphere. The radius of the maximum variance unit sphere is defined by $\hat{\mathbf{r}}_{maxV.i}$. The voxels are then transformed relative to the minimum variance unit sphere, $\hat{\mathbf{P}}_{obs_i}$, using the same transformation, ψ_{E_i} (Equation 4.33 and Figure 4.10).

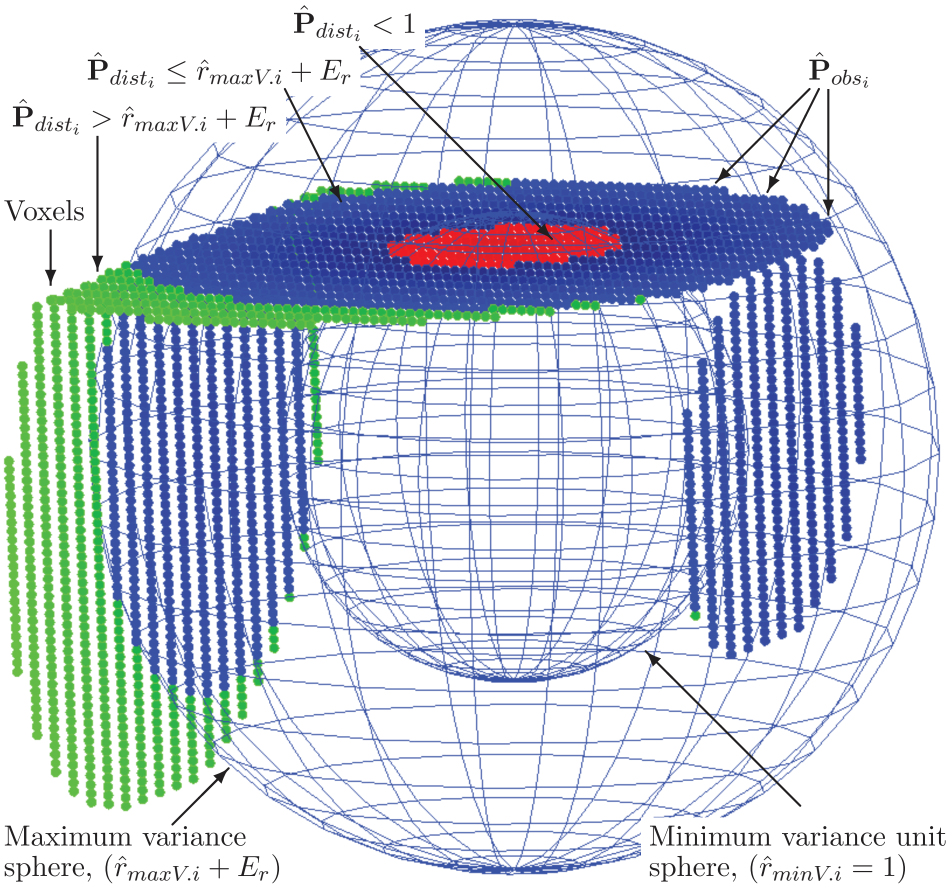

$\hat{\mathbf{P}}_{dist_i} < 1$
$\hat{\mathbf{P}}_{dist_i} \leq \hat{r}_{maxV.i} + E_r$
$\hat{\mathbf{P}}_{dist_i} > \hat{r}_{maxV.i} + E_r$
$\hat{\mathbf{P}}_{obs_i}$
Voxels
Maximum variance sphere, $(\hat{r}_{maxV.i} + E_r)$
Minimum variance unit sphere, $(\hat{r}_{minV.i} = 1)$

FIGURE 4.10: Transformed voxels relative to the unit sphere. Red voxels are within the minimum variance unit sphere, blue voxels are between the minimum variance unit sphere and maximum variance sphere, green voxels are outside the maximum variance sphere.

The distance from the origin of the spheres to each transformed voxel is found, $\hat{\mathbf{P}}_{dist_i}$ (Equation 4.34). All voxels within the maximum variance sphere plus an error factor, E_r, are determined (Equation 4.35). The voxels related to these within the original coordinate frame of reference are considered as the closest voxels, $\mathbf{P}_{cls_i}$, and are used to determine the repulsive forces. The error factor, E_r, is based on the speed of the end-effector [155].

$$\hat{\mathbf{P}}_{obs_i} = \mathbf{P}_{obs} \cdot \psi_{E_i} \tag{4.33}$$

$$\hat{\mathbf{P}}_{dist_i} = \|\hat{\mathbf{P}}_{obs_i}\| \tag{4.34}$$

$$\mathbf{P}_{cls_i} = \{\mathbf{P}_{obs_i} : \hat{\mathbf{P}}_{dist_i} \leq \hat{r}_{maxV.i} + E_r\} \tag{4.35}$$

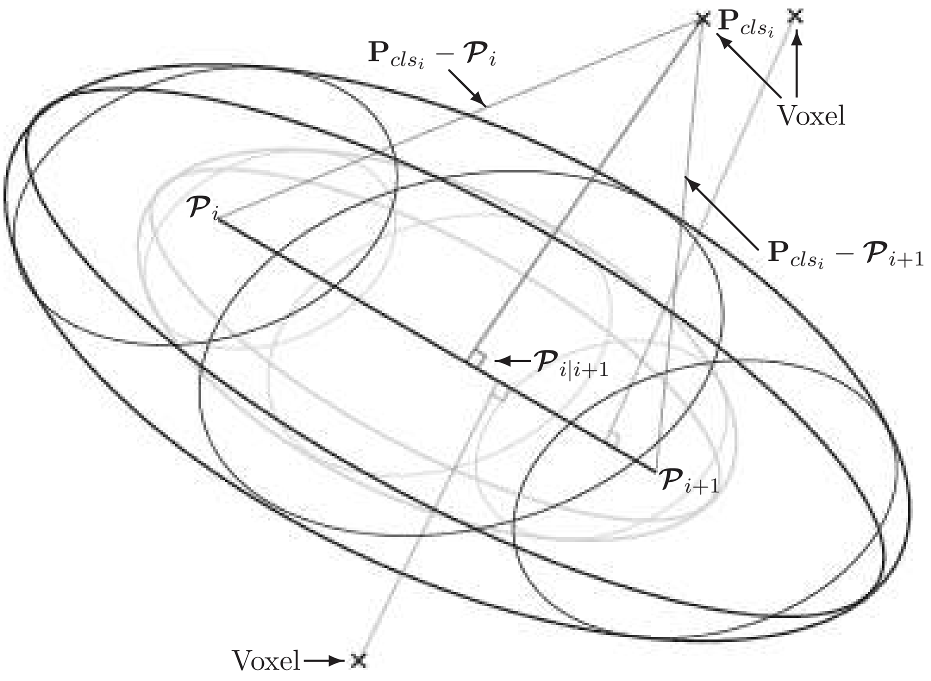
The relative distance to the minimum variance unit sphere, $\hat{\mathbf{P}}_{dist_i}$, is related to the relative penetration between the two variance force fields. If the distance is less than 1 the voxel is within the minimum variance force field and the algorithm fails; if it is between 1 and the maximum variance unit sphere radius plus an error factor a repulsive force is generated; if it is larger than the maximum variance unit sphere plus an error factor no repulsive force is generated (Figure 4.10).

Before a repulsive force is generated the application point on the link needs to be determined. For each of the closest voxels, $\mathbf{P}_{cls_i}$, the closest point on the link defined by the position of the coordinate frame of references $\boldsymbol{\mathcal{P}}_i$ and $\boldsymbol{\mathcal{P}}_{i+1}$ is calculated, $\boldsymbol{\mathcal{P}}_{i|i+1}$ (Equation 4.36 and Figure 4.11). This point is the application point for the repulsive force. The repulsive force, $\boldsymbol{f}_{rep.cls_i}$, for each of the closest voxels is found (Equation 4.37 and Figure 4.12). Here K_f is the repulsive force amplitude.

$$\boldsymbol{\mathcal{P}}_{i|i+1} = \boldsymbol{\mathcal{P}}_i + \left(\frac{\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i}{\|\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i\|}\left((\mathbf{P}_{cls_i} - \boldsymbol{\mathcal{P}}_i) \cdot \frac{\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i}{\|\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i\|}\right)\right) \tag{4.36}$$

$$\boldsymbol{f}_{rep.cls_i} = K_f - K_f/(1 + \exp(-10\frac{\hat{\mathbf{P}}_{dist_i} - r_{maxV.i} - 0.5E_r}{E_r}))\frac{\boldsymbol{\mathcal{P}}_{i|i+1} - \mathbf{P}_{cls_i}}{\|\boldsymbol{\mathcal{P}}_{i|i+1} - \mathbf{P}_{cls_i}\|} \tag{4.37}$$

The 3D-F^2 algorithm only considered a single closest voxel which, in many situations, would be sufficient. When moving through a tunnel or close to a corner it may be necessary to apply repulsive forces in multiple directions to prevent collision. A repulsive force, $\boldsymbol{f}_{rep.cls_i}$, is comprised of a component along each axis (Equation 4.38). It is suggested that the highest repulsive force along each axis, in both the positive and negative directions, be considered, $\boldsymbol{f}_{rep_i}$ (Equation 4.39 and Figure 4.13). While it is acknowledged that this may lead to similar repulsive forces being applied, it is preferential to any collisions. Directionality along the same axis is considered to allow motion through gaps where repulsive forces may be generated from opposing walls.



$\mathbf{P}_{cls_i} - \boldsymbol{\mathcal{P}}_i$
$\mathbf{P}_{cls_i}$
Voxel
$\boldsymbol{\mathcal{P}}_i$
$\mathbf{P}_{cls_i} - \boldsymbol{\mathcal{P}}_{i+1}$
$\boldsymbol{\mathcal{P}}_{i|i+1}$
$\boldsymbol{\mathcal{P}}_{i+1}$
Voxel

FIGURE 4.11: The closest point, $\boldsymbol{\mathcal{P}}_{i|i+1}$ on a link, $\boldsymbol{\mathcal{P}}_i$ to $\boldsymbol{\mathcal{P}}_{i+1}$ to a given voxel, $\mathbf{P_{cls_i}}$.

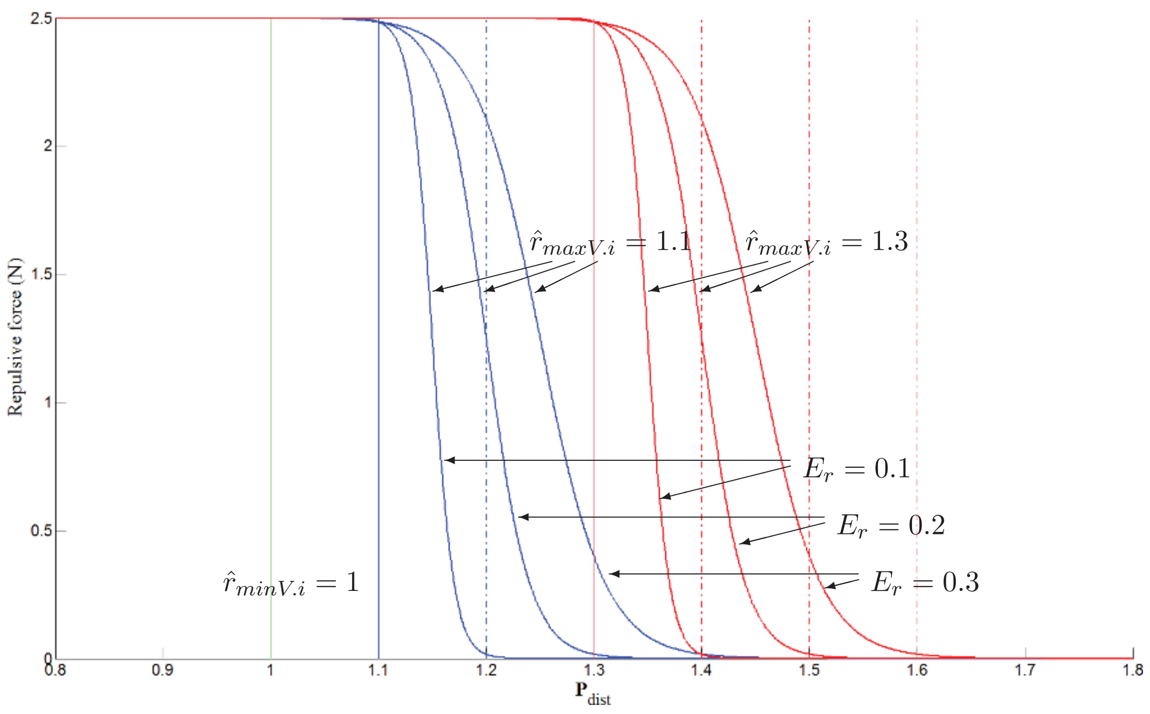


FIGURE 4.12: The repulsive force versus unit sphere distances at varying error factors. Each of the blue and red lines show the amplitude of the generated repulsive forces for maximum variance spheres with radii of 1.1 and 1.3 respectively.

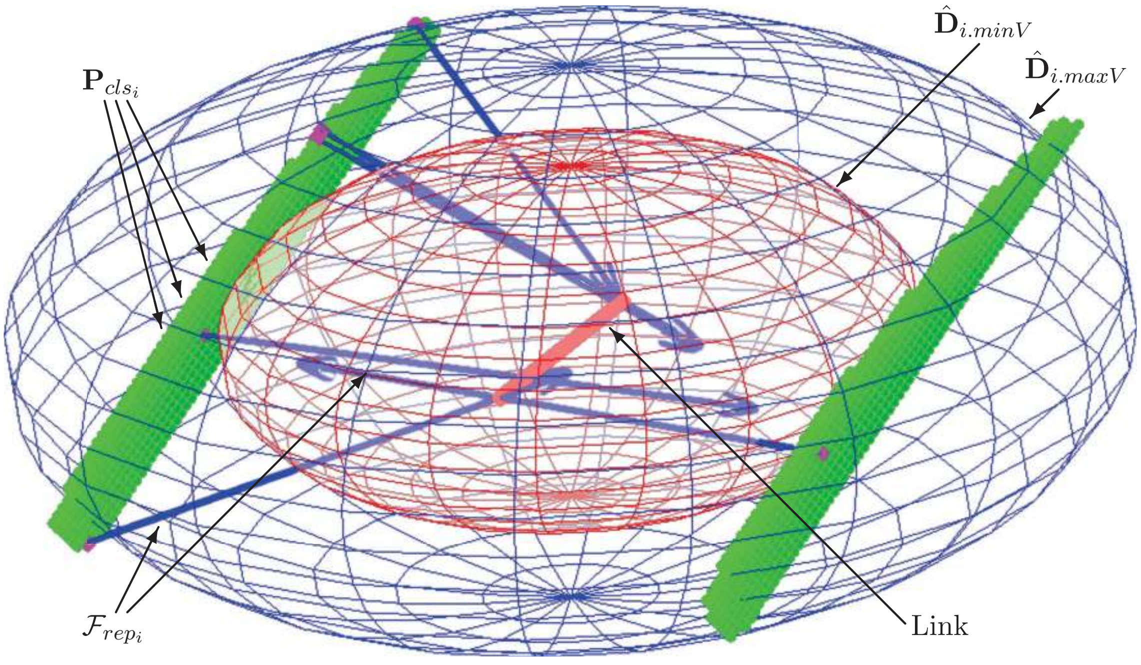


FIGURE 4.13: Multiple repulsive forces acting on a link. Green dots are voxels, the red and blue ellipsoids are the minimum and maximum variance force fields respectively, the red line is the link, the blue vectors are the repulsive forces with their related voxels coloured magenta.

$$\boldsymbol{f}_{rep.cls_i} = \begin{bmatrix} \boldsymbol{f}_{rep.cls_i.x} & \boldsymbol{f}_{rep.cls_i.y} & \boldsymbol{f}_{rep.cls_i.z} \end{bmatrix}^T \tag{4.38}$$

$$\boldsymbol{f}_{rep_i} : \Big[\min_{\boldsymbol{f}_{rep.cls_i}} (\boldsymbol{f}_{rep.cls_i.\{x|y|z\}} | \boldsymbol{f}_{rep.cls_i.\{x|y|z\}} < 0) \\ \max_{\boldsymbol{f}_{rep.cls_i}} (\boldsymbol{f}_{rep.cls_i.\{x|y|z\}} | \boldsymbol{f}_{rep.cls_i.\{x|y|z\}} > 0)\} \Big] \tag{4.39}$$

The final repulsive force, $\boldsymbol{\mathcal{F}}_{rep_i}$, only considers the translational repulsive forces along each axis (Equation 4.40). As multiple repulsive forces may be generated from different voxels, j represents the total number of repulsive forces generated for a link i.

$$\boldsymbol{\mathcal{F}}_{rep_i} = \begin{bmatrix} 0_{3\times j} \\ \boldsymbol{f}_{rep_i} \end{bmatrix} \tag{4.40}$$

4.2.3 Repulsive Force Due to Self-Collision

Repulsive forces can also be used to prevent the multi-link serial robot from colliding with itself. The 3D-F^2 algorithm used sphere-swept bounding to determine the self-collision repulsive force. While this is an effective method, it is incompatible with the variance force fields. Instead self-collision forces should be generated when two maximum variance force fields intersect. If either minimum variance force fields are penetrated the algorithm fails.

Self-collision checks are performed between a link, i, and any preceding link, j. As link $i-1$ is in direct contact with link i it is excluded from the calculations. The closest point on the maximum variance force field to the minimum variance force field is found [202]. For each link, i, the closest point to the preceding maximum variance force fields is found $\boldsymbol{\mathcal{P}}_{i.maxV}$; while for the preceding links, j, the closest points on the maximum variance force field of link i are found, $\boldsymbol{\mathcal{P}}_{j.maxV}$ (Figure 4.14). This is done for both the link and the preceding links as each force field may be penetrated to different degrees.

In a similar manner to the environmental repulsive forces, collision checks are performed with unit spheres. Transformation matrices are found which when applied to the maximum

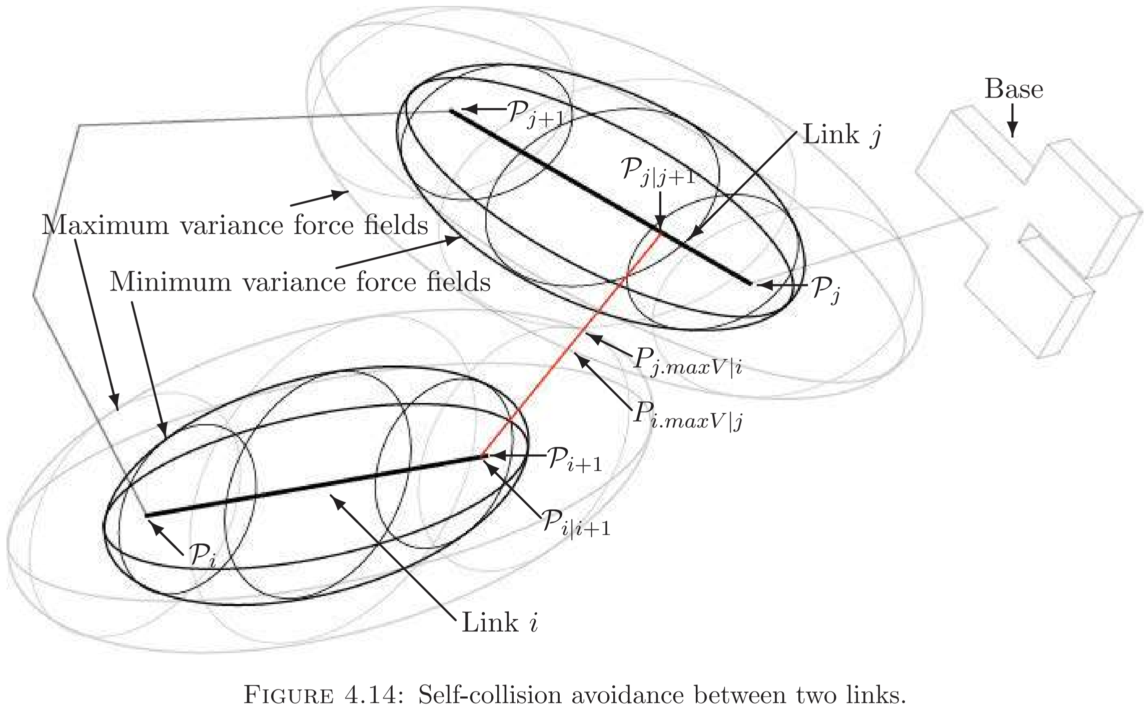


FIGURE 4.14: Self-collision avoidance between two links.

variance force fields will transform them to unit spheres, ψ_{E_i} and ψ_{E_j}. The closet point on the preceding force field, $\boldsymbol{\mathcal{P}}_{j.maxV}$, is transformed with the matrix ψ_{E_i} relative to the minimum variance unit sphere of link i, $\boldsymbol{\mathcal{P}}_{j.maxV|i}$ (Equation 4.41). The distance from the transformed point to the unit sphere's origin is calculated, $d_{j.maxV|i}$ (Equation 4.42). The transformation matrix, ψ_{E_i}, is also applied to the maximum variance force field resulting in a sphere with a radius of $\hat{r}_{maxV.i}$. This process is repeated using the transformation matrix ψ_{E_j} to transform the closest point on the initial link to each preceding link's minimum variance unit sphere to find; the transformed point, $\boldsymbol{\mathcal{P}}_{i.maxV|j}$ (Equation 4.43), the distance to the unit sphere origin, $d_{i.maxV|j}$ (Equation 4.44), and the radius of their maximum variance unit spheres, $\hat{r}_{maxV.j}$.

$$\boldsymbol{\mathcal{P}}_{j.maxV|i} = \boldsymbol{\mathcal{P}}_{j.maxV} \cdot \psi_{E_i} \tag{4.41}$$

$$d_{j.maxV|i} = \|\boldsymbol{\mathcal{P}}_{j.maxV|i}\| \tag{4.42}$$

$$\boldsymbol{\mathcal{P}}_{i.maxV|j} = \boldsymbol{\mathcal{P}}_{i.maxV} \cdot \psi_{E_j} \tag{4.43}$$

$$d_{i.maxV|j} = \|\boldsymbol{\mathcal{P}}_{i.maxV|j}\| \tag{4.44}$$

A repulsive force is generated if either distance $d_{j.maxV|i}$ or $d_{i.maxV|j}$ is less than its opposing link's maximum variance unit sphere radius plus an error factor, E_r (Equation 4.45 and 4.46). If the distances are above these values no self-repulsive forces are generated. If either distance is below 1, the points have penetrated the minimum variance force field and the algorithm fails. The error factor, E_r, is based on the speed of the end-effector [155].

$$d_{i.maxV|j} < \hat{r}_{maxV.i} + E_r \tag{4.45}$$

$$d_{j.maxV|i} < \hat{r}_{maxV.j} + E_r \tag{4.46}$$

The strength of the self-repulsive force is based on the distance penetrated into a force field, d_{cls} (Equation 4.47). As self-collision repulsive forces are based on the penetration between two force fields with different radii, the distance of the force field most deeply penetrated should be used. Ratios which represent the relative penetration for each force field are developed, $\phi_{j.maxV|i}$ and $\phi_{i.maxV|j}$ (Equation 4.48 and 4.49). This is calculated by taking the norm of the transformed closest point, $\boldsymbol{\mathcal{P}}_{j.maxV|i}$ and $\boldsymbol{\mathcal{P}}_{i.maxV|j}$, relative to the radius of the opposing maximum variance unit sphere, $\hat{r}_{maxV.i}$ and $\hat{r}_{maxV.j}$. Equation 4.47 is then used to select the penetration distance related to the force field with the larger ratio. This penetration distance is used to generate the repulsive force.

$$d_{cls} = \begin{cases} d_{i.maxV|j} & \text{if } \phi_{j.maxV|i} \leq \phi_{i.maxV|j} \\ d_{j.maxV|i} & \text{if } \phi_{j.maxV|i} > \phi_{i.maxV|j} \end{cases} \tag{4.47}$$

$$\phi_{j.maxV|i} = \frac{\hat{r}_{maxV.i} - \|\boldsymbol{\mathcal{P}}_{j.maxV|i}\|}{\hat{r}_{maxV.i} - 1} \tag{4.48}$$

$$\phi_{i.maxV|j} = \frac{\hat{r}_{maxV.j} - \|\boldsymbol{\mathcal{P}}_{i.maxV|i}\|}{\hat{r}_{maxV.j} - 1} \tag{4.49}$$

The self-collision repulsive force is always applied to link i as motion in this link is less likely to affect the overall motion. The point, $\boldsymbol{\mathcal{P}}_{i|i+1}$, is the closest point on the link i to the preceding link's maximum variance force field, $\boldsymbol{\mathcal{P}}_{i.maxV|j}$ (Equation 4.50). As the closest point on the preceding link's maximum variance force field, $\boldsymbol{\mathcal{P}}_{i.maxV|j}$, was found relative to the unit sphere, the matrix $\psi_{\hat{D}_{i.minV}}$ is used to transform the point back to the original coordinate frame of reference (Equation 4.51).

$$\boldsymbol{\mathcal{P}}_{i|i+1} = \boldsymbol{\mathcal{P}}_i + \left(\frac{\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i}{\|\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i\|} \left((\boldsymbol{\mathcal{P}}_{j.maxV|i} - \boldsymbol{\mathcal{P}}_i) \cdot \frac{\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i}{\|\boldsymbol{\mathcal{P}}_{i+1} - \boldsymbol{\mathcal{P}}_i\|} \right) \right) \tag{4.50}$$

$$\boldsymbol{\mathcal{P}}_{j.maxV|i} = \psi_{\hat{D}_{i.minV}} \cdot \hat{\boldsymbol{\mathcal{P}}}_{j.maxV|i} \tag{4.51}$$

The self-collision repulsive force, $\boldsymbol{f}_{self_i}$, is then calculated using Equation 4.52. Here the maximum amplitude of the repulsive force is K_f. The ellipsoid coverage constant, K_p, is based on the radius of the maximum variance unit sphere with the higher penetration ratio (Equation 4.53).

$$\boldsymbol{f}_{self_i} = K_f - K_f/(1 + \exp(-10\frac{d_{cls} - K_P - 0.5E_r}{E_r}))\frac{\boldsymbol{\mathcal{P}}_{i|i+1} - \boldsymbol{\mathcal{P}}_{i.maxV|j}}{|\boldsymbol{\mathcal{P}}_{i|i+1} - \boldsymbol{\mathcal{P}}_{i.maxV|j}|} \tag{4.52}$$

$$K_p = \begin{cases} \hat{r}_{maxV.i} & \text{if } \phi_{j.maxV|i} \leq \phi_{i.maxV|j} \\ \hat{r}_{maxV.j} & \text{if } \phi_{j.maxV|i} > \phi_{i.maxV|j} \end{cases} \tag{4.53}$$

The final self-collision repulsive force, $\boldsymbol{\mathcal{F}}_{self_i}$, only considers the translational repulsive forces (Equation 4.54). Here k is the number of self-collision repulsive forces generated for a link i.

$$\boldsymbol{\mathcal{F}}_{self_i} = \begin{bmatrix} 0_{3\times k} \\ \boldsymbol{f}_{self_i} \end{bmatrix} \tag{4.54}$$

4.2.4 Attractive Force

The attractive force is used to guide the multi-link serial robot's end-effector to the goal location (Figure 4.15). This basic premise remains unchanged from the 3D-F^2 algorithm's implementation; however, the 3D-PF^2 algorithm's implementation differs by using the model developed in Chapter 3.

The goal location is comprised of a position vector, $\mathbf{P}_g$, and rotational vector, $\mathbf{R}_g$. The attractive force is derived to align the end-effector coordinate frame of reference position, ${}^0\boldsymbol{\mathcal{P}}_e$, to the goal position vector, $\mathbf{P}_g$, and the orientation component of its rotational matrix, ${}^0\boldsymbol{\mathcal{R}}_{e.o}$ to the goal rotational vector, $\mathbf{R}_g$. As only the orientation component of the

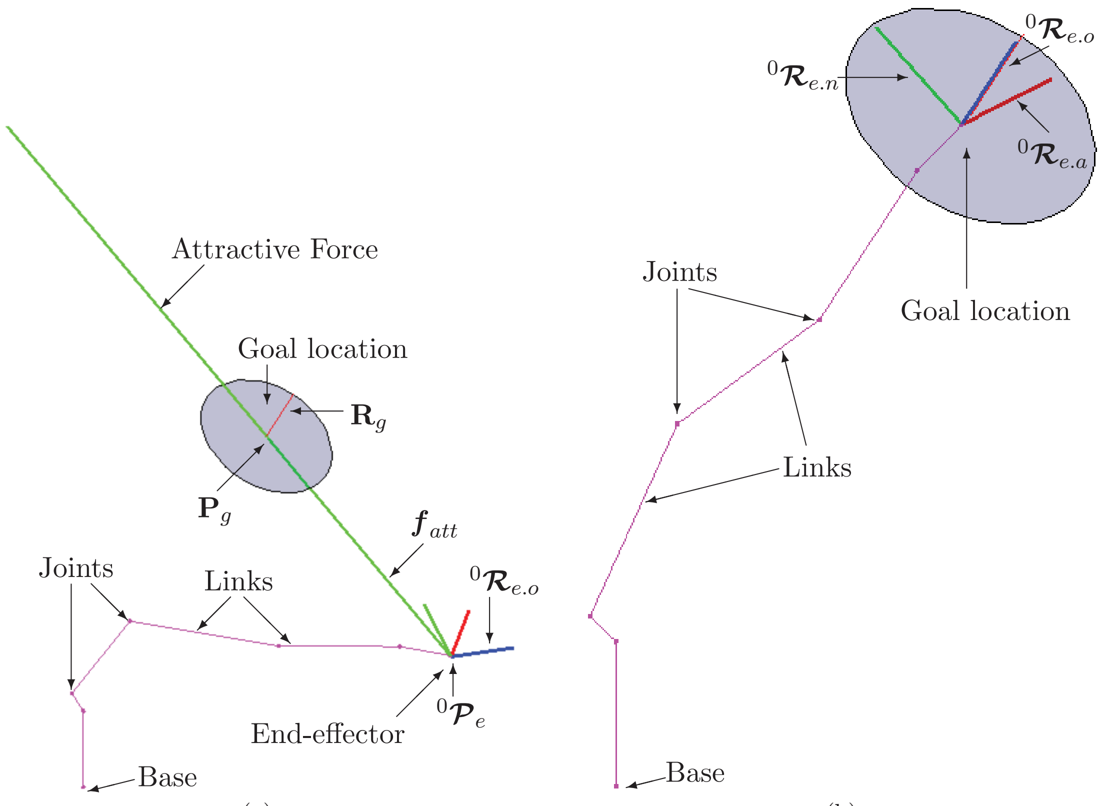


FIGURE 4.15: (a) Wireframe representation of the multi-link serial robot with the translational attractive force vector shown at a start pose and (b) at the goal location.

rotational matrix is used, both the normal, ${}^0\mathcal{R}_{e.n}$, and approach, ${}^0\mathcal{R}_{e.a}$, components are not constrained. The attractive force could be adjusted to function with any component of the rotational matrix; however, if all three components are used it should be assured that the goal rotational vectors are not mutually exclusive.

The attractive force, $\boldsymbol{\mathcal{F}}_{att}$, is comprised of two components; a translational, $\boldsymbol{f}_{att}$, and a rotational, $\boldsymbol{w}_{att}$ (Equation 4.55). The translational component is a linear force while the rotational component is an angular momentum. These components are determined through sigmoid functions with their amplitude based on the difference between the end-effector and goal locations (Equation 4.56 and 4.57). Here $K_{att.T}$ and $K_{att.R}$ are the maximum attractive forces for the translation and rotational components respectively, K_{zero} is a small non-zero positive constant and K_s is a constant which determines how the attractive force varies over distance (Figure 4.16). The attractive force is applied to the end-effector position, ${}^0\mathcal{P}_e$.

$$\boldsymbol{\mathcal{F}}_{att} = \begin{bmatrix} \boldsymbol{w}_{att} \\ \boldsymbol{f}_{att} \end{bmatrix} \tag{4.55}$$

$$\boldsymbol{f}_{att} = K_{att.T}/(1 + \frac{\exp\left(-K_s\|\mathbf{P}_g - {}^0\boldsymbol{\mathcal{P}}_e\|\right)}{K_{zero}})\frac{\mathbf{P}_g - {}^0\boldsymbol{\mathcal{P}}_e}{\|\mathbf{P}_g - {}^0\boldsymbol{\mathcal{P}}_e\|} \tag{4.56}$$

$$\boldsymbol{w}_{att} = K_{att.R}/(1+\frac{\exp\left(-K_s \sin^{-1}(\|{}^0\boldsymbol{\mathcal{R}}_{e.a}\cdot\mathbf{R}_g\|)/(\|{}^0\boldsymbol{\mathcal{R}}_{e.a}\|\cdot\|\mathbf{R}_g\|)\right)}{K_{zero}})\frac{{}^0\boldsymbol{\mathcal{R}}_{e.a}\cdot\mathbf{R}_g}{\|{}^0\boldsymbol{\mathcal{R}}_{e.a}\cdot\mathbf{R}_g\|} \tag{4.57}$$

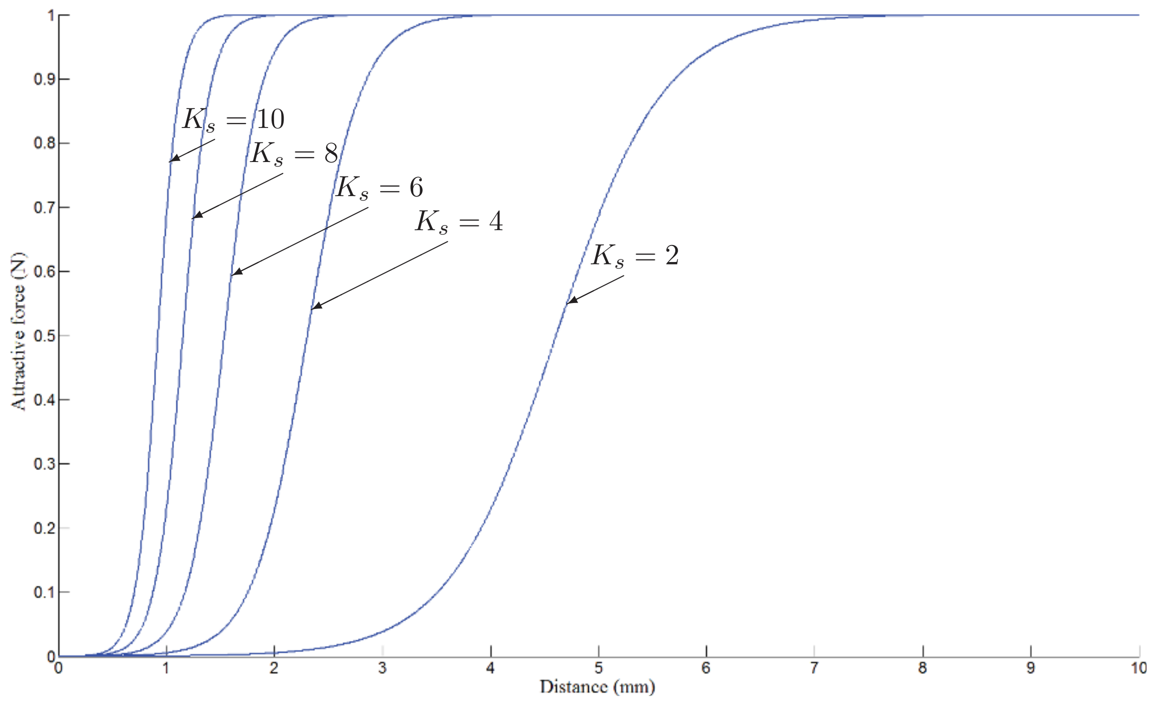
4.2.5 Dynamic Model

The dynamic model (Equation 4.22) is amended to be based on the model (Equation 3.21) described in Chapter 3. As the composition of the link is unchanged the mass-inertia matrix of the links, the damping coefficient and the stiffness coefficients also remain unchanged. The components of the equations are calculated based on the kinematic model (Equation 4.58); these components include the torque-force matrix, $\hat{\boldsymbol{\Gamma}}$, the attractive force, $\boldsymbol{\mathcal{F}}_{att}$, the repulsive forces, $\boldsymbol{\mathcal{F}}_{rep_i}$, and their application points, ${}^0\boldsymbol{\mathcal{P}}_{att_n}$ and ${}^0\boldsymbol{\mathcal{P}}_{rep_i}$. The joint accelerations, $\ddot{\mathbf{q}}$ and velocities, $\dot{\mathbf{q}}$, are updated each program cycle. This equation is iteratively solved for the pose, $\mathbf{q}$, until either the goal position is found or a local minimum is entered. As uncertainty is not considered in the joint pose, neither the joint velocities nor accelerations are considered to have uncertainty.

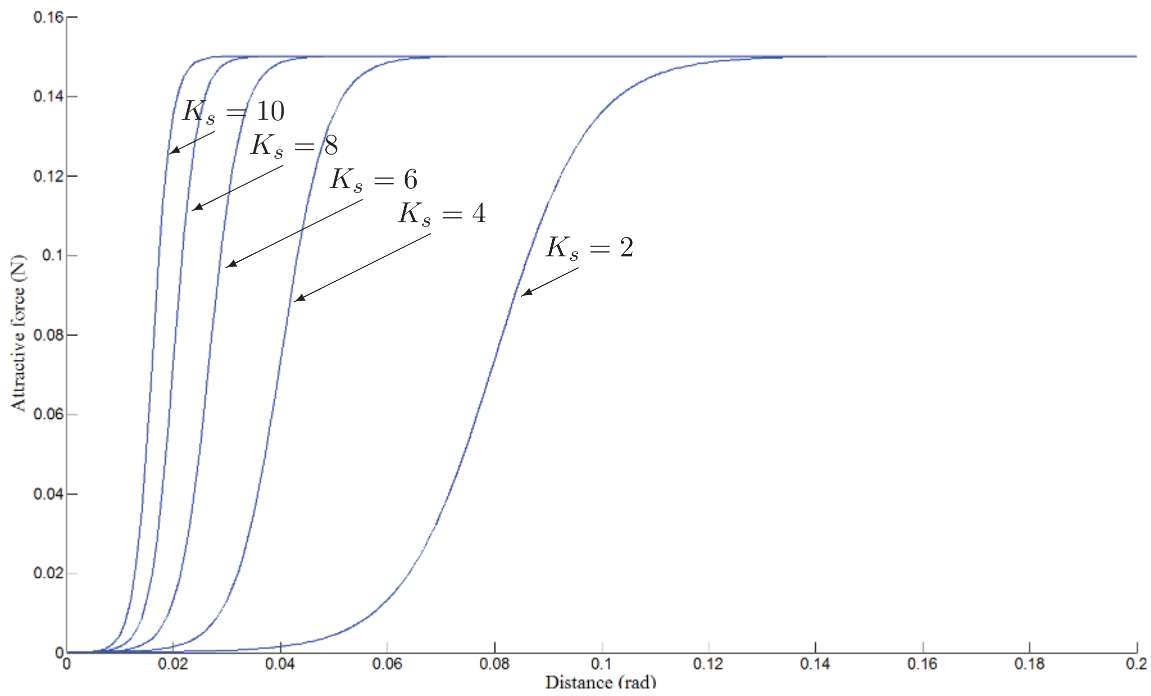
$$\hat{\boldsymbol{\Gamma}}_{att} + \sum_{i=1}^{n}\hat{\boldsymbol{\Gamma}}_{rep_i} = \boldsymbol{\mathcal{H}}(\mathbf{q}_{t-1}, {}^0\boldsymbol{\mathcal{P}}_{att_n})_n\boldsymbol{\mathcal{F}}_{att} + \sum_{i=1}^{n}\boldsymbol{\mathcal{H}}(\mathbf{q}_{t-1}, {}^0\boldsymbol{\mathcal{P}}_{rep_i})_i\boldsymbol{\mathcal{F}}_{rep_i} = \mathbf{I}\ddot{\mathbf{q}} + \boldsymbol{\beta}\dot{\mathbf{q}} + \mathbf{k}_{sp}\mathbf{q} \tag{4.58}$$

4.2.6 Force Control Algorithm

The torque-force matrix, $\hat{\boldsymbol{\Gamma}}$, is amended to consider the model (Equation 4.58). As the force, $\boldsymbol{\mathcal{F}}$, and the point the force is applied to, ${}^0\boldsymbol{\mathcal{P}}_f$, are calculated based on the model,



(a)



(b)

FIGURE 4.16: The (a) position and (b) rotation components of the attractive force with different K_s constants over different distances (Equations 4.56 and 4.57 respectively). Here $K_{att.T}$ is 1 and $K_{att.R}$ is 0.15.

the transformation matrix, $\boldsymbol{\mathcal{H}}_i$, needs to be changed to also be calculated based on the model (Equation 4.59).

$$\boldsymbol{\mathcal{H}}(\mathbf{q}, {}^0\boldsymbol{\mathcal{P}}_f)_i = \begin{bmatrix} \hat{\mathbf{h}}_i, & ..., & \hat{\mathbf{h}}_n \end{bmatrix}^T \qquad i = 1, ..., n \tag{4.59}$$

$$\hat{\mathbf{h}}_j = \begin{cases} 0_{1\times 6} & \text{if i > j} \\ \mathbf{L}_j {}^i\boldsymbol{\mathcal{M}}_0 & \text{if i} \leq \text{j} \end{cases} \qquad j = 1, ..., n \tag{4.60}$$

$$\mathbf{L}_i = \begin{bmatrix} l_{r.x_i} & l_{r.y_i} & l_{r.z_i} & l_{t.x_i} & l_{t.y_i} & l_{t.z_i} \end{bmatrix} \tag{4.61}$$

$$l_{r.k_i} = \begin{cases} 1 & \text{if joint } j \text{ is rotational about axis } r \\ 0 & \text{otherwise} \end{cases} \qquad k \in \{x, y, z\} \tag{4.62}$$

$$l_{t.k_i} = \begin{cases} 1 & \text{if joint } j \text{ is translational along axis } r \\ 0 & \text{otherwise} \end{cases} \qquad k \in \{x, y, z\} \tag{4.63}$$

The rotation skew-symmetric matrix, ${}^i\boldsymbol{\mathcal{M}}_0$, transfers the force in the global coordinate frame to the coordinate frame of joint i (Equation 4.64). Here the rotation component of the coordinate frame of reference, ${}^0\boldsymbol{\mathcal{R}}_i$, is used while the skew-symmetric matrix, $\mathbf{S}(\Delta\boldsymbol{\mathcal{P}})$, is based on the variation between the point the force is applied, ${}^0\boldsymbol{\mathcal{P}}_f$, and the position of the coordinate frame i, ${}^0\boldsymbol{\mathcal{P}}_i$.

$${}^i\boldsymbol{\mathcal{M}}_0 = \begin{bmatrix} {}^0\boldsymbol{\mathcal{R}}_i^T & 0_{3\times 3} \\ 0_{3\times 3} & {}^0\boldsymbol{\mathcal{R}}_i^T \end{bmatrix} \begin{bmatrix} 1_{3\times 3} & 0_{3\times 3} \\ -\mathbf{S}(\Delta\boldsymbol{\mathcal{P}}_i) & 1_{3\times 3} \end{bmatrix} \tag{4.64}$$

$$\mathbf{S}(\Delta\boldsymbol{\mathcal{P}}_i) = \begin{bmatrix} 0 & -\Delta\mathcal{P}_z & \Delta\mathcal{P}_y \\ \Delta\mathcal{P}_z & 0 & -\Delta\mathcal{P}_x \\ -\Delta\mathcal{P}_y & \Delta\mathcal{P}_x & 0 \end{bmatrix} \tag{4.65}$$

$$\Delta\boldsymbol{\mathcal{P}}_i = \begin{bmatrix} \Delta\mathcal{P}_x \\ \Delta\mathcal{P}_y \\ \Delta\mathcal{P}_z \end{bmatrix} = {}^0\boldsymbol{\mathcal{P}}_f - {}^0\boldsymbol{\mathcal{P}}_i \tag{4.66}$$

4.2.7 Local Minima Detection and Termination Criteria

As the 3D-F^2 algorithm [155] was used in conjunction with a human operator no methods were implemented which consider local minima. This section describes the local minima detection algorithm implemented with the 3D-PF^2 algorithm. When a local minimum is detected the 3D-PF^2 algorithm fails and terminates.

Local minima occur when the sum of forces acting on the multi-link serial robot equal zero, essentially stalling motion. However, this is not necessarily instantaneous. While the forces acting may equate to zero, the multi-link serial robot's velocity may cause it to continue to move through and potentially exit the local minimum. Also, the multi-link serial robot may be caught in an area which results in continuous, repeated motion such as bouncing between two obstacles. The multi-link serial robot is then considered to be in a local minimum when it is not at the end node and at least one of the following three conditions are met.

The first condition occurs when the multi-link serial robot has not moved significantly over a period. This occurs when the sum of the variation between consecutive joint poses, $\mathbf{q}$, along the path over a period, k_{period}, is less than a threshold, k_q (Equation 4.67). Taking the values over a period prevents erroneously detecting local minima when the multi-link serial robot changes direction as it may momentarily stop moving.

$$\sum_{l=2}^{k_{period}} |\mathbf{q}_{l-1} - \mathbf{q}_l| < k_q \tag{4.67}$$

The second local minima condition occurs when the joint velocities over a set period are insignificant (Equation 4.68) implying no motion has occurred. This detects local minima sooner than the first condition when joint accelerations are approaching zero. This occurs when the sum of the variation between consecutive joint velocities, $\dot{\mathbf{q}}$, over the period, k_{period}, is less than the velocity threshold, $k_{\dot{q}}$.

$$\sum_{l=2}^{k_{period}} |\dot{\mathbf{q}}_{l-1} - \dot{\mathbf{q}}_l| < k_{\dot{q}} \tag{4.68}$$

The threshold values of k_q and $k_{\dot{q}}$ should be determined based on the number of joints, the expected motion of joints and the desired response time of the algorithm to detect the local minimum. Higher thresholds would be required when the robot has more joints as the sum of the variation over a larger number of joints would, expectedly, be higher. Conversely, if the threshold is too high, erroneously classified local minimum may occur if the expected motion of the joints are slow. Finally the desired detection response time of the system, based on the application, further dictates the thresholds. Faster response times would occur with higher threshold values and slower responses with lower values.

The final condition is used to identify when the multi-link serial robot repeats the same pose and velocities (Equation 4.69) implying the robot is stuck in a loop from which it may not exit. This is defined as when both the current pose, $\mathbf{q}$, and velocity, $\dot{\mathbf{q}}$, have occurred at the same instance previously along the pose set, $\mathbf{q}_{set}$, and velocity set, $\dot{\mathbf{q}}_{set}$, within a set of tolerances, $k_{q_{set}}$ and $k_{\dot{q_{set}}}$.

$$|\mathbf{q} \in \mathbf{q}_{set}| < k_{q_{set}} \wedge |\dot{\mathbf{q}} \in \dot{\mathbf{q}}_{set}| < k_{\dot{q_{set}}} \tag{4.69}$$

4.2.8 Surface Attachment

Surface attachment is a special case for the 3D-PF^2 algorithm. As contact is made with the surface, considerations need to be made to ensure the generated forces do not prevent attachment and that attachment is handled in a safe manner. To this effect surface attachment occurs in two main stages.

In the first stage a preliminary goal is found at the preparatory approach distance, d_a, away from the surface goal position (Figure 4.17). This preliminary goal position is orientated in the same manner as the surface goal position, $\mathbf{R}_g$. The 3D-F^2 algorithm develops a path to this preliminary goal position prior to the final attachment motion.

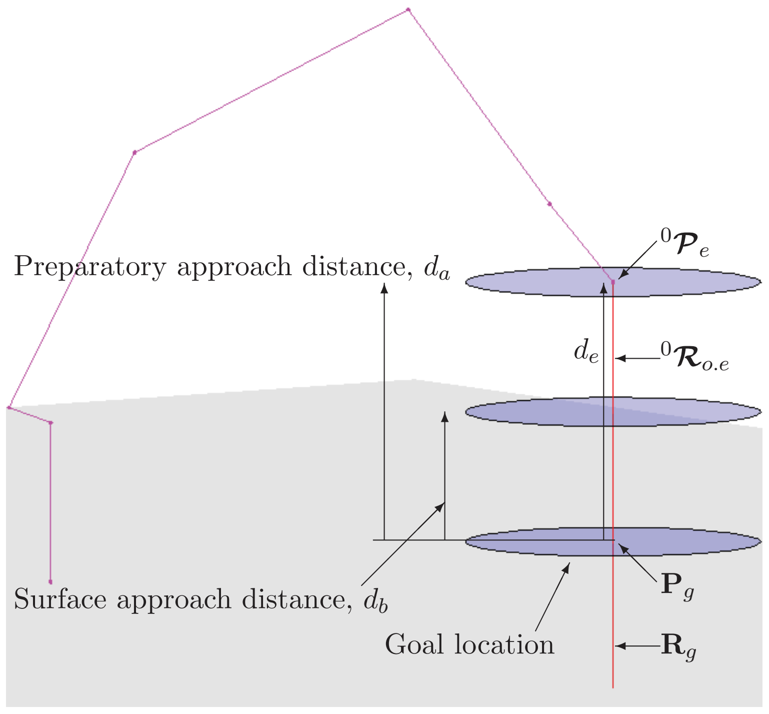

Preparatory approach distance, d_a
${}^0\boldsymbol{\mathcal{P}}_e$
d_e
${}^0\boldsymbol{\mathcal{R}}_{o.e}$
$\mathbf{P}_g$
Surface approach distance, d_b
Goal location
$\mathbf{R}_g$

FIGURE 4.17: Different stages of surface attachment. From the preparatory approach distance ($d_e = d_a$) the end-effector is guided through the surface approach distance ($d_e = d_b$) to the goal location ($d_e = 0$).

The second stage involves guiding the end-effector towards the surface goal position. The growth factor, K_g, (Equation 3.16), is increased to a maximum at the surface approach distance, d_b, based on the distance from the end-effector to the surface goal location, d_e. Between the surface approach distance and the surface goal position the growth factor remains constant at 1. During this second stage a number of considerations need to be made.

Firstly, the voxels directly surrounding the surface goal location should not generate any environmental repulsive forces. This allows the end-effector to contact the surface. Environmental repulsive forces from other voxels may still be generated.

Secondly, if the 3D-PF^2 algorithm were to move towards a single point, the environmental repulsive forces generated from voxels to the side of the surface goal location may prevent the goal location from being reached. In this case it may be prudent to instead use a goal region. Provided the end-effector position, ${}^0\boldsymbol{\mathcal{P}}_e$, lands within the goal region, the algorithm is successful and surface attachment can be performed. To facilitate this, rather than calculating the translational component of the attractive force, $\boldsymbol{f}_{att}$, in the global coordinate frame, it is calculated in the end-effector's coordinate frame of reference. The components of the translational attractive force now act along the axes of the end-effector's coordinate frame of reference. The translational force acting on the axis aligned with the goal normal, ${}^{0}\boldsymbol{\mathcal{R}}_{o.e}$, has its calculated force replaced with a constant force. This force causes the end-effector to move towards the surface goal location. The forces acting along the other axes will move the end-effector towards the goal normal, $\mathbf{R}_g$, positioned at the centre of the goal region. This allows constant motion towards a goal region to be achieved.

Finally, as there exists variance in the end-effector location it is difficult to determine when the surface has been reached without a sensor, such as a Hall effector sensor. With regards to the inchworm robot a surface detection sensor is not available. Instead the algorithm is considered at the goal once the end-effector position is at the surface goal location.

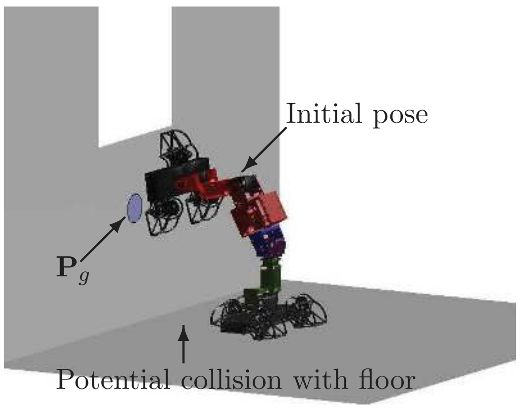
4.3 Simulations and Verification

Simulations were performed using MATLAB® to verify the 3D-PF^2 algorithm. Simulations were run using an Intel i7 3.4 GHz. The algorithm is tested using a 7DOF inchworm robot [1]. In all tests the inchworm robot is positioned on a surface at a start pose. A path is then generated which attempts to manoeuvre the inchworm robot safely to a goal location. Comparisons are made with the 3D-PF^2 algorithm by considering different allowed maximum hand position and structural uncertainties. When both the mean and variance for both uncertainties are zero, the 3D-PF^2 algorithm is similar to the 3D-F^2 algorithm as the model is deterministic. The uncertainty at a joint's coordinate frame of reference is defined by interpolating the uncertainty at the end-effector, $G_c(...)$, $F_c(...)$ and $E_c(...)$ (Equations 3.24 to 3.26), assuming a linear relationship. That is, the uncertainty at a joint is linearly dependent on the ratio between the joint's distance from the base over the maximum distance between the base and end-effector (Chapter 3). In all simulations the allowed maximum uncertainty was varied; all other variables remained constant. A maximum variance factor, ξ_{max}, and a minimum variance factor, ξ_{min}, of 1 and 0.75 was used respectively.

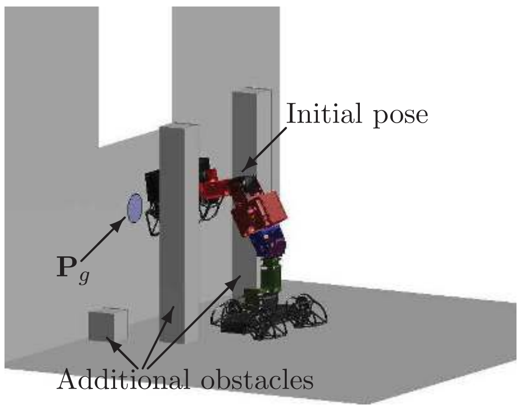
4.3.1 Verification of the Attachment Procedure in Simulated Environments

This simulation tested the capacity of the 3D-PF^2 algorithm with an attachment procedure and how the algorithm adjusts to possible collisions with different means and variances in the hand position uncertainty. A surface goal region with a 2cm radius, orientated parallel to the surface, is used to show the effects of the algorithm and show failure conditions. This surface goal region is shown as a disc. For each simulation both stages of the attachment procedure were completed. In the event of failure, the algorithm was allowed to continue until the end-effector reached the surface to provide additional insight into the algorithm. Reaching the surface does not mean that the surface goal region was successfully reached only that the end-effector touched a point of the surface.

Two cases were used to verify the attachment. In the first case the obstacle near the surface goal location is the floor (Figure 4.18a). In the second case obstacles are positioned near the robot body (Figure 4.18b). The results of these simulations are show in Tables 4.1 and 4.2 and Figures 4.19 to 4.22. In the graphs, the repulsive forces are shown as the sum of all repulsive forces generated at a given program cycle.

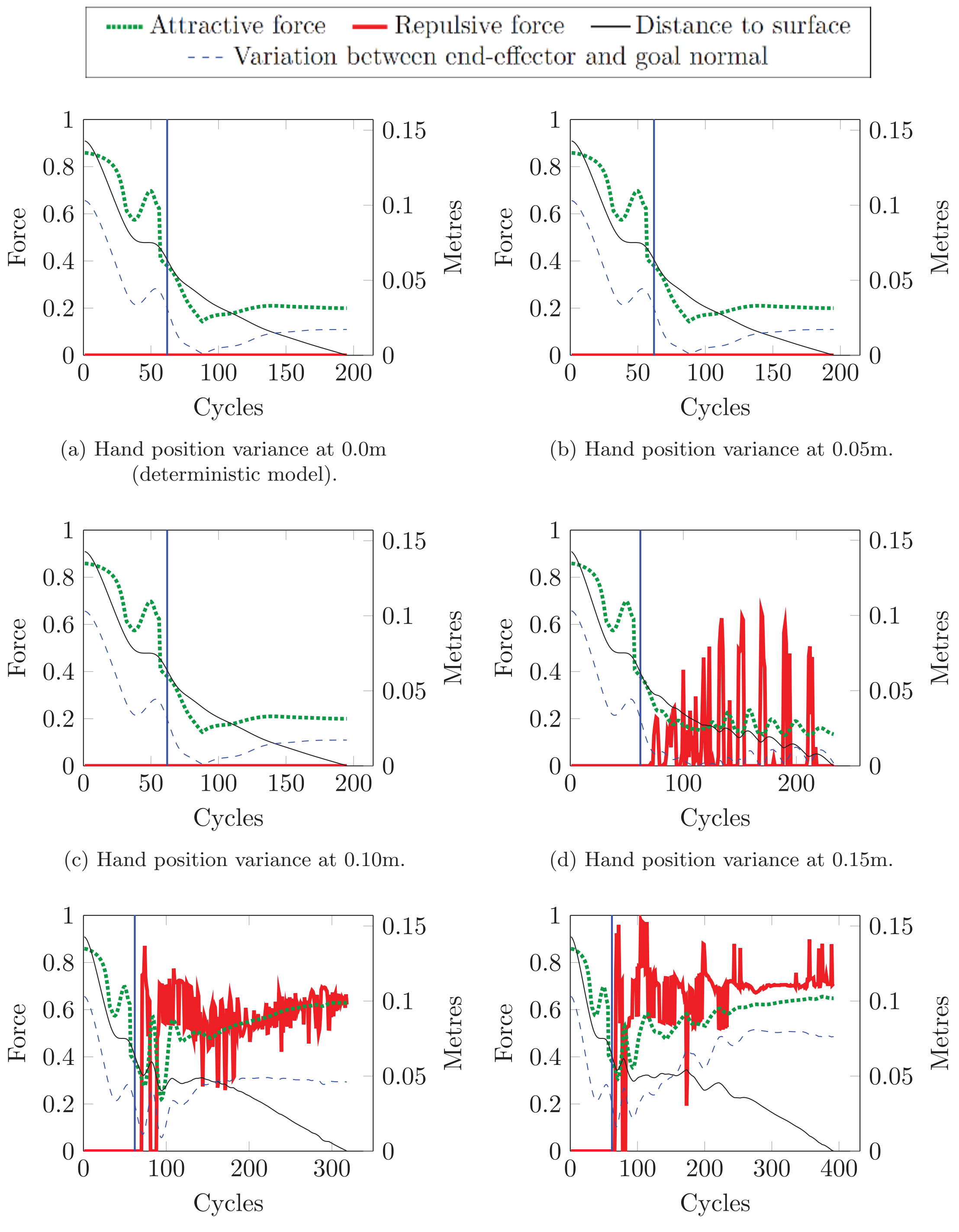


(a)



(b)

FIGURE 4.18: (a) Case 1: An obstacle near the surface goal location. As the variance increases, the size of the force fields also increase. This increases the chances of repulsive forces being generated from potential collision with the floor. (b) Case 2: Three additional obstacles are placed near the body of the robot. The deterministic representation of the inchworm robot is shown as is the goal location, $\mathbf{P}_g$. The goal orientation, $\mathbf{R}_g$, is not shown as it is pointing towards the surface. The goal region is shown as a circle centred at the goal location.



(a) Hand position variance at 0.0m (deterministic model).

(b) Hand position variance at 0.05m.

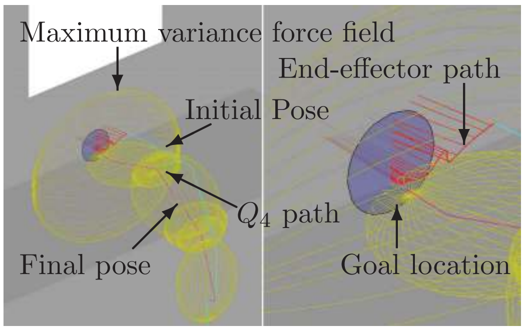
(c) Hand position variance at 0.10m.

(d) Hand position variance at 0.15m.

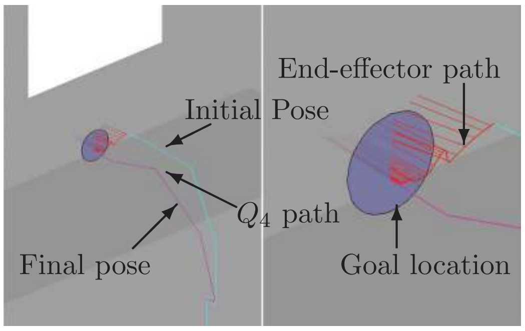
(e) Hand position variance at 0.20m.

(f) Hand position variance at 0.25m.

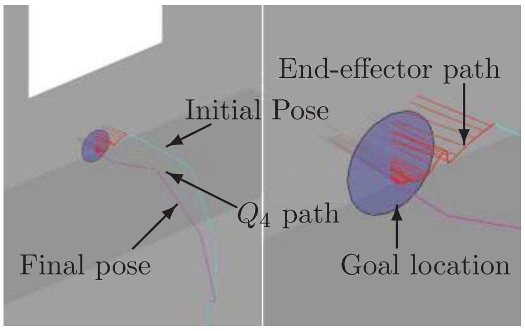
FIGURE 4.19: Case 1: Graphs of forces, the distance to the goal and the end-effector's distance to the goal normal at varying allowed maximum hand position variance values. The vertical blue lines indicate the cycle when the first stage begins.



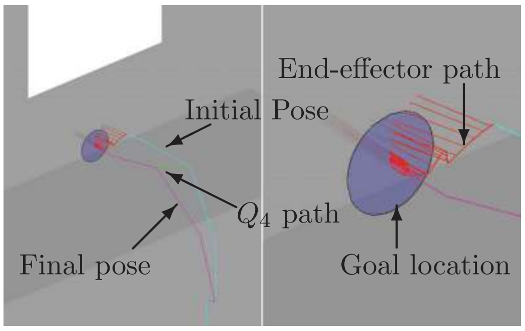
(a) Hand position variance at 0.00m (deterministic model)



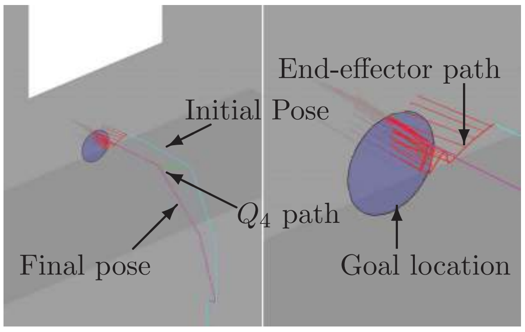
(b) Hand position variance at 0.05m



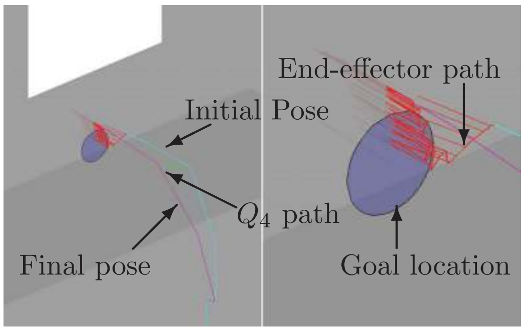
(c) Hand position variance at 0.10m



(d) Hand position variance at 0.15m

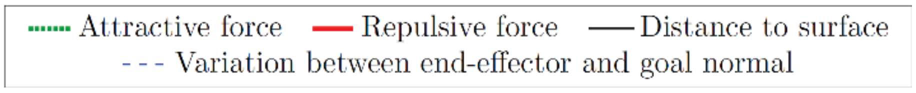


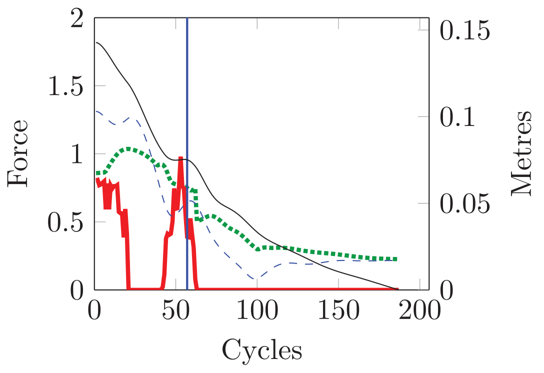
(e) Hand position variance at 0.20m



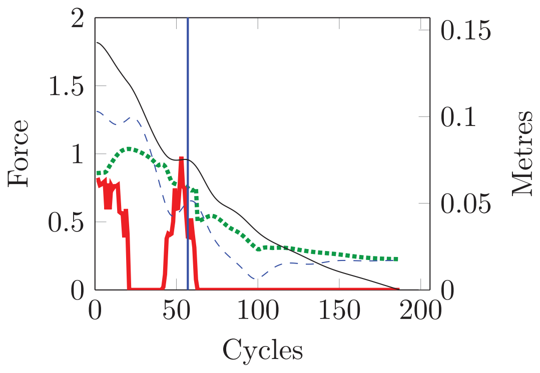
(f) Hand position variance at 0.25m

FIGURE 4.20: Attachment results: Case 1 at different hand position variance values. Each figure shows final pose with the end-effector position, ${}^{0}\mathcal{P}_{\mathbf{e}}$, and orientation, ${}^{0}\mathcal{R}_{\mathbf{e.o}}$, along the path in red. The initial pose is in cyan, final pose in magenta, and the position of the fourth joint along the path in green. Force fields are omitted from Figures 4.20b to 4.20f for clarity.

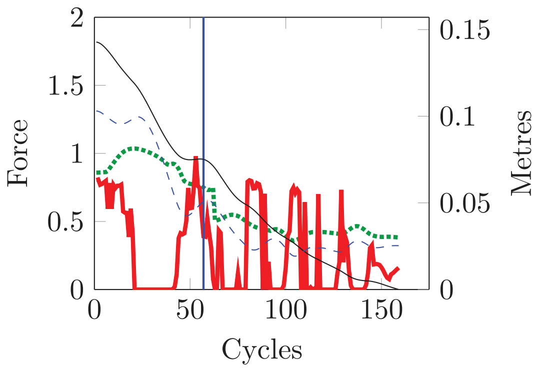




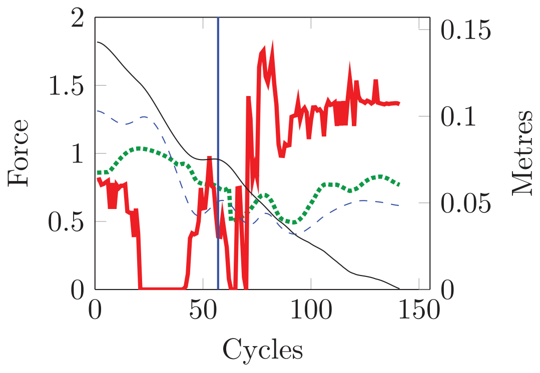
(a) Hand position variance at 0.0m (deterministic model).



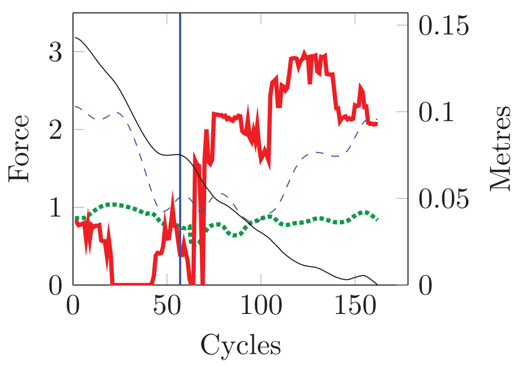
(b) Hand position variance at 0.05m.



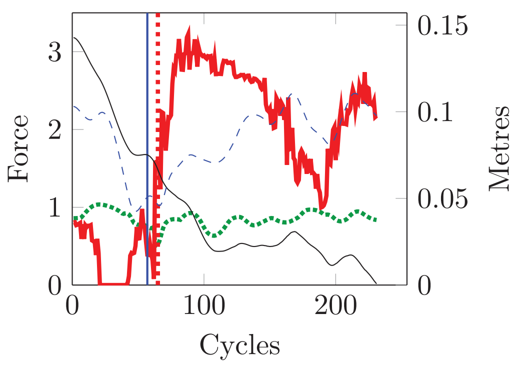
(c) Hand position variance at 0.10m.



(d) Hand position variance at 0.15m.

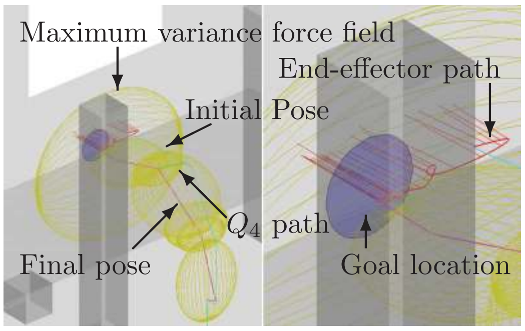


(e) Hand position variance at 0.20m.

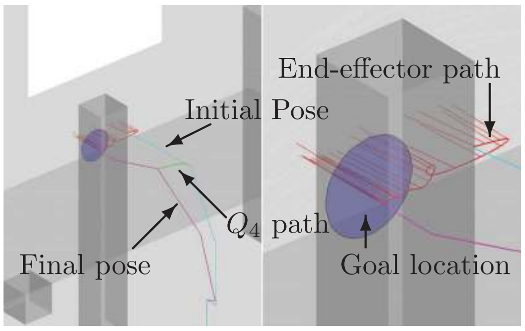


(f) Hand position variance at 0.25m.

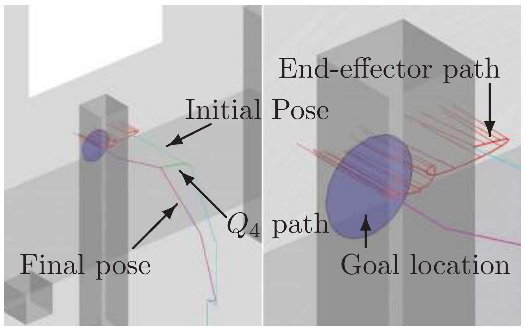
FIGURE 4.21: Case 2: Graphs of forces, the distance to the goal and the end-effector's distance to the goal normal at varying allowed maximum hand position variance values. The vertical blue lines indicate the cycle when the first stage begins. The vertical dotted red lines show when a failure condition occurred; the algorithm continued in these cases.



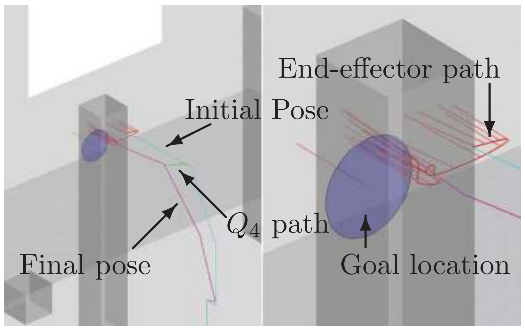
(a) Hand position variance at 0.00m (deterministic model)



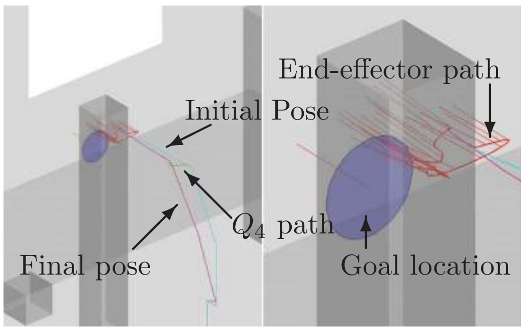
(b) Hand position variance at 0.05m



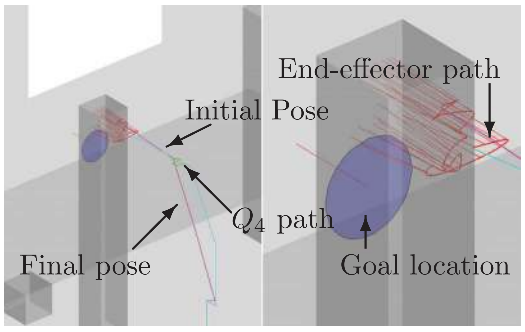
(c) Hand position variance at 0.10m



(d) Hand position variance at 0.15m



(e) Hand position variance at 0.20m



(f) Hand position variance at 0.25m

FIGURE 4.22: Attachment results: Case 2 at different hand position variance values. Each figure shows final pose with end-effector position, ${}^{0}\mathcal{P}_{\mathbf{e}}$, and orientation, ${}^{0}\mathcal{R}_{\mathbf{e.o}}$, along the path in red. The initial pose is in cyan, final pose in magenta, and the position of the fourth joint along the path in green. Force fields are omitted from Figures 4.22b to 4.22f for clarity.

Table 4.1: Case 1: Obstacle near surface goal location

Allowed maximum hand position variance (m)	0	0.05	0.10	0.15	0.20	0.25
Termination condition	S	S	S	S	S	FL
Final end-effector distance to goal (m)	0.017	0.017	0.017	0.001	0.042	0.074
Program cycles to goal	196	196	196	234	319	393

[1] S - Success
[2] FL - Failure due to end-effector moving outside landing region

Table 4.2: Case 2: Obstacle near surface goal location

Allowed maximum hand position variance (m)	0	0.05	0.10	0.15	0.20	0.25
Termination condition	S	S	S	S	FL	FF/FL
Final end-effector distance to goal (m)	0.011	0.011	0.022	0.076	0.149	0.158
Path length	188	188	160	142	163	232

[1] S - Success
[2] FL - Failure due to end-effector moving outside landing region
[2] FF - Failure due to obstacle entering minimum variance force field

Stage one of the attachment procedure finds a path from the initial pose to the preparatory approach distance. This path was developed in 62 and 64 program iterations for cases 1 and 2 respectively. This point is shown as a blue vertical line in Figures 4.19 and 4.21. Stage 2 then seeks to move the end-effector towards the surface goal location. In case 2 repulsive forces are generated from the initiation of the algorithm due to an obstacle (Figure 4.21a). As the obstacle was not within the minimum variance force field the algorithm continued.

The deterministic model is considered when the allowed maximum hand position variance is zero. In both cases, the 3D-PF^2 algorithm successfully found a path to the goal region for this model. Successful results were also found in both cases when the allowed maximum variance was less than 0.1m and 0.05m respectively.

In case 1, repulsive forces are generated when the force field surrounding the final link contacts the floor. The distance from the floor to the goal location is 0.4m. As the end-effector approaches the goal location, the hand position uncertainty increases which increases the size of the force fields and allows the algorithm to compensate for the obstruction. This can be seen in Figure 4.19d. As the force field increases, the repulsive forces similarly increase. Each "spike" in the repulsive force adjusts the path to prevent collisions while the attractive force continues toward the surface goal location. The interaction between the forces causes the end-effector to land within 0.001m of the goal location.

The success is repeated at an allowed maximum variance of 0.2m (Figure 4.19e); however, due to the size of the resulting force field, repulsive forces are constantly generated while the attractive force seeks to align the orientation component of the end-effector's rotational matrix with the goal normal. In normal operation this may have resulted in a local minimum; however, in stage 2 of the attachment procedure a constant attractive force is applied towards the goal location allowing a path to be successfully found. In this instance the final end-effector distance to the goal is 0.042m. This is higher than previous runs. Figure 4.20d shows the end-effector landing towards the edge of the landing region.

At a variance of 0.25m the 3D-PF^2 algorithm fails to find a path which successfully lands within the goal region. The generated repulsive forces are consistently higher than those generated at previous allowed maximum variances. This is due to larger force fields being penetrated further by the floor (Figure 4.19f). This causes the end-effector to land outside of the goal region (Figure 4.20f).

In case 2, three obstacles with fixed positions were placed within the environment. In four simulations, with values for the allowed maximum hand position variances of 0.15m and less, successful paths were found. The remaining two simulations in case 2 with values for the variance of 0.2m and greater failed as the end-effector landed outside of the goal region. While the failures in case 1 are also due to the end-effector landing outside of the goal region, the generated repulsive forces now act on the robot body rather than exclusively on the end-effector due to the additional obstacles. This results in the repulsive forces generated to be generally larger in case 2 (Figure 4.21) than case 1 (Figure 4.19) for the equivalent variances.

At a variance of 0.25m the 3D-PF^2 algorithm fails as the minimum variance force field is penetrated upon initiation of stage 2 of the surface attachment procedure after the hand position uncertainty is applied (Figure 4.21f and 4.22f). As a high value for the variance is applied, the size of the force fields increased quickly. This caused an obstacle to penetrate the minimum variance force field before the algorithm could respond to the generated repulsive force. While this did cause the algorithm to fail immediately the algorithm was allowed to continue to garner additional results. The algorithm then failed once more by failing to land successfully within the goal region.

Failure due to the algorithm not responding quickly enough to the generated repulsive force may be prevented provided the algorithm variables are tuned correctly. In these simulations only the allowed maximum hand position variance is varied while other variables remained constant. This may have prevented the algorithm from functioning optimally for given values of the uncertainty. A number of variables related to the repulsive forces could be modified to improve the performance and prevent this failure. A greater value for the repulsive force amplitude, K_f, will generate a higher repulsive force causing a greater reaction from the algorithm to prevent collisions. A smaller value for the minimum variance factor, ξ_{min}, will allow an obstacle to penetrate the maximum force field further before a failure occurs. While the difference between the values for the preparatory approach distance, d_a, and the surface approach distance, d_b, could be increased which would reduce the rate the hand position uncertainty increases over the distance. These variables should be tuned for a given robot provided constant allowed maximum uncertainties are applied.

4.3.2 Verification of the 3D-PF^2 Algorithm in Simulations

This section compares the 3D-PF^2 algorithm, at different allowed maximum structural means and variances, in a number of different environments (Figure 4.23 and 4.24). In this simulation the partition plate and manhole are considered cases 1 and 2 respectively. In each case the inchworm robot is positioned on one side of the obstruction with the goal location on the opposite side. The simulations are run with differing allowed maximum structural translations, rotations and variances. The goal location is considered to be reached if the end-effector's position is within 0.05m and it's orientation is within 5°.

Tables 4.3 and 4.4 show the results of the simulations summarised based on the allowed maximum variance. Tables 4.5 and 4.6 show the full results. From these tables it can be seen that the success of the simulation is highly dependent on the allowed maximum variance. Graphs of the forces and joint positions for case 1 are shown in Figures 4.25 and 4.26 and for case 2 are shown in Figures 4.33 and 4.34. In these graphs the repulsive forces are shown as the sum of all repulsive forces generated at a given program cycle.

Table 4.3: Case 1: Results of partition plate environment collected based on allowed maximum variance

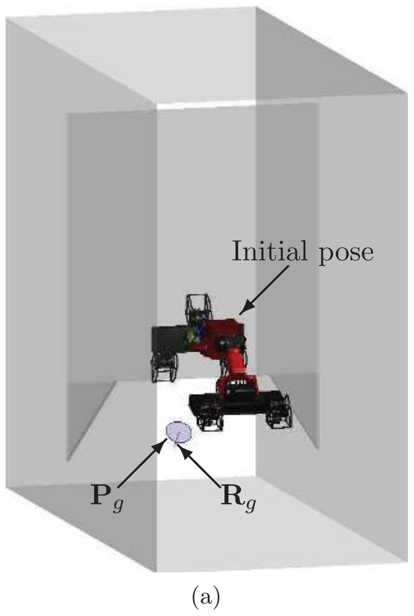
Allowed maximum variance (m)	0	0.025	0.025+
Success rate (%)	100	67	0
Average final distance to goal (m)	0.040	0.043	0
Average final angle to goal (°)	3.62	3.48	0
Average program cycles	367.17	400.5	0

Table 4.4: Case 2: Results of manhole environment collected based on allowed maximum variance

Allowed maximum variance (m)	0	0.025	0.05	0.075	0.1	0.125
Success rate (%)	100	100	50	33	0	17
Average distance to goal location (m)	0.033	0.036	0.018	0.032	0	0.043
Average final angle to goal (°)	3.21	3.22	3.10	4.23	0	4.87
Average program cycles	413.17	977.17	908.33	934.00	0	811.00

In both cases, the 3D-PF^2 algorithm successfully found a path to the goal location when the deterministic model was used. The deterministic model is considered when the allowed maximum translation, rotation and variance are zero. In case 1, minimal repulsive forces are generated through a potential collision with the floor and then with the partition plate itself (Figures 4.25a and 4.27). Case 2 has similar success with repulsive forces only generated with potential collision with the manhole (Figures 4.33a and 4.35). The goal location in both cases was then reached.

In both cases with an allowed maximum variance of zero, the 3D-PF^2 algorithm was able to successfully generate paths at all allowed maximum translations and rotations; however, as the allowed maximum variance increased the success rate decreased. For case 1, successful paths were only found when the allowed maximum variance was 0.025m or below. For case 2, the 3D-PF^2 algorithm successfully found solutions at values for the majority of allowed maximum variances (excluding at 0.075m); although, predictably, at higher values for the allowed maximum variances the success rate decreased. This was due to the size of the



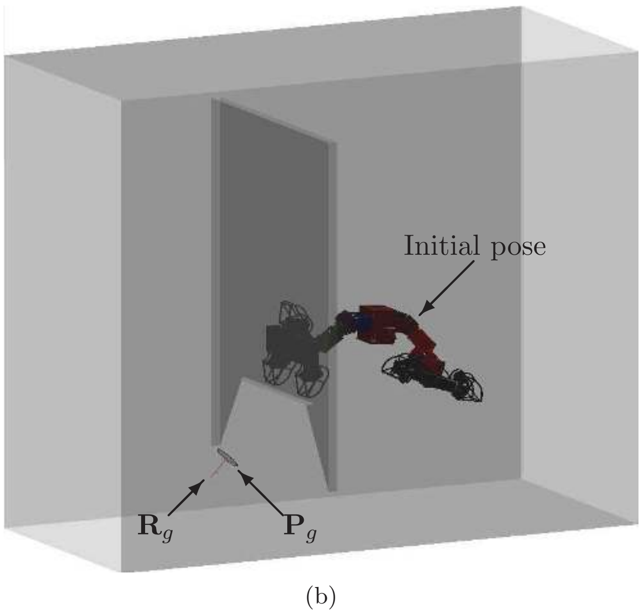
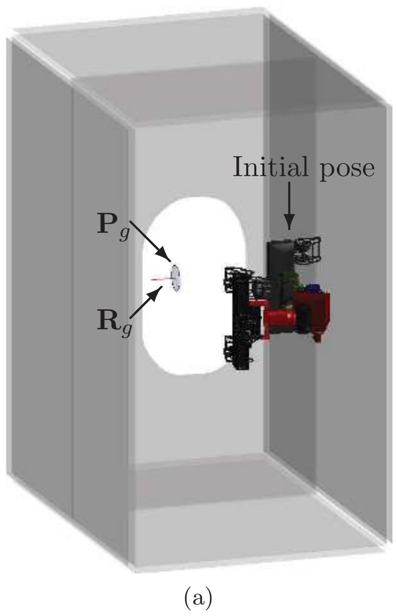


FIGURE 4.23: Case 1: Partition plate environment from the front (a) and from the side (b). The deterministic representation of the inchworm robot at the initial pose; the goal location position, $\mathbf{P}_g$, and orientation, $\mathbf{R}_g$ are shown. The deterministic representation considers all allowed maximum mean and variance values to be zero.



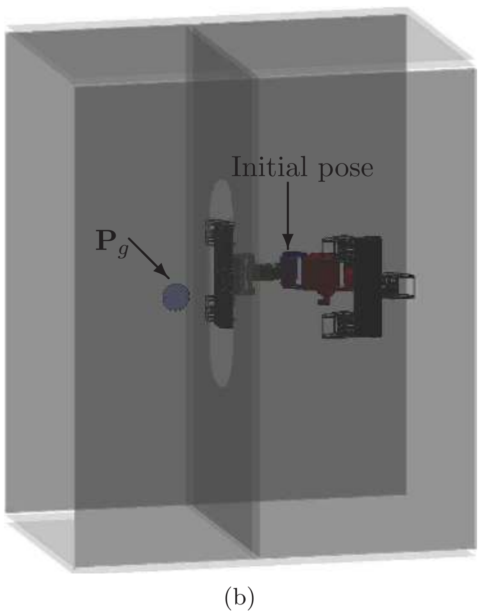


FIGURE 4.24: Case 2: Manhole environment from the front (a) and from the side (b). The deterministic representation of the inchworm robot at the initial pose; the goal location position, $\mathbf{P}_g$, and orientation, $\mathbf{R}_g$ are shown. The deterministic representation considers all allowed maximum mean and variance values to be zero.

Table 4.5: Case 1: Results of partition plate simulation with various allowed maximum translations, rotations and variances.

Allowed maximum translation (m)	0.00						0.025					
Allowed maximum rotation (°)	0						-5					
Allowed maximum variance (m)	**0**	0.025	0.05	0.075	0.1	0.125	0	0.025	0.05	**0.075**	0.1	0.125
Termination condition	**S**	S	FF	FF	FF	FF	S	S	FF	**FF**	FF	FF
Final distance to goal (m)	**0.024**	0.032	0.516	0.556	0.58	0.599	0.038	0.041	0.123	**0.583**	0.611	0.614
Final angle to goal (°)	**2.78**	3.5	34.72	28.71	25.59	22.2	2.66	3.25	36.39	**27.6**	23.02	25.38
Program cycles	**319**	329	46	38	35	33	305	333	379	**38**	34	38

Allowed maximum translation (m)	0.05						0.075					
Allowed maximum rotation (°)	-10						-15					
Allowed maximum variance (m)	0	0.025	**0.05**	0.075	0.1	0.125	0	**0.025**	0.05	0.075	**0.1**	0.125
Termination condition	S	S	**FF**	FF	FF	FF	S	**S**	FF	FF	**LM**	FF
Final distance to goal (m)	0.043	0.054	**0.022**	0.606	0.593	0.613	0.051	**0.045**	0.429	0.074	**0.658**	0.449
Final angle to goal (°)	4.14	4.37	**48.74**	26.57	31.6	33.36	4.22	**2.81**	3.79	43.52	**4.14**	61.59
Program cycles	330	366	**607**	484	111	117	338	**574**	381	402	**2090**	540

Allowed maximum translation (m)	0.10						0.125					
Allowed maximum rotation (°)	-20						-25					
Allowed maximum variance (m)	0	0.025	0.05	0.075	0.1	0.125	0	0.025	0.05	0.075	0.1	**0.125**
Termination condition	S	FF	FF	FF	FF	FF	S	FF	FF	FF	FF	**FF**
Final distance to goal (m)	0.041	0.241	0.215	0.278	0.584	0.432	0.043	0.726	0.569	0.616	0.589	**0.491**
Final angle to goal (°)	3.81	11.66	22.77	18.52	5.17	18.47	4.11	10.69	57.31	11.24	5.5	**49.35**
Program cycles	368	320	326	326	329	351	543	23	222	481	303	**176**

[1] S - Success

[2] FF - Termination due to obstacle entering the minimum variance force field

[3] LM - Termination due to local minima

[4] Simulation results of bold values shown in Figure 4.27 to 4.31

Table 4.6: Case 2: Results of manhole simulation with various allowed maximum translations, rotations and variances.

Allowed maximum translation (m)	0.00						0.025					
Allowed maximum rotation (°)	0						-5					
Allowed maximum variance (m)	**0**	0.025	0.05	0.075	0.1	0.125	0	0.025	**0.05**	0.075	0.1	0.125
Termination condition	**S**	S	FF	FF	LM	LM	S	S	**FF**	FF	FF	LM
Final distance to goal (m)	**0.042**	0.045	0.247	0.199	0.451	0.432	0.029	0.033	**0.271**	0.256	0.329	0.432
Final angle to goal (°)	**3.4**	1.73	49.25	55.22	0.17	2.34	3.04	4.34	**28.8**	24.45	27.83	2.39
Program cycles	**361**	1117	1345	1361	435	470	391	979	**471**	505	1826	611

Allowed maximum translation (m)	0.05						0.075					
Allowed maximum rotation (°)	-10						-15					
Allowed maximum variance (m)	0	0.025	0.05	0.075	0.1	**0.125**	0	0.025	0.05	**0.075**	0.1	0.125
Termination condition	S	S	S	S	FF	**S**	S	S	S	**LM**	FF	FF
Final distance to goal (m)	0.030	0.035	0.016	0.043	0.021	**0.043**	0.032	0.024	0.017	**0.473**	0.075	0.061
Final angle to goal (°)	3.17	4.05	1.29	3.7	38.97	**4.87**	2.76	4.99	3.41	**1.93**	38.68	31.12
Program cycles	408	858	1018	983	576	**811**	426	1102	813	**2526**	508	656

Allowed maximum translation (m)	0.10						0.125					
Allowed maximum rotation (°)	-20						-25					
Allowed maximum variance (m)	0	0.025	0.05	0.075	0.1	**0.125**	0	0.025	0.05	0.075	0.1	**0.125**
Termination condition	S	S	S	S	FF	**LM**	S	S	FF	FF	FF	**FF**
Final distance to goal (m)	0.033	0.038	0.021	0.02	0.113	**0.497**	0.033	0.04	0.298	0.21	0.048	**0.574**
Final angle to goal (°)	3.91	1.65	4.62	4.75	20.52	**28.85**	2.96	2.54	28.36	36.01	15.5	**12.89**
Program cycles	446	1003	894	885	683	**875**	447	804	784	492	689	**1894**

[1] S - Success
[2] FF - Termination due to obstacle entering the minimum variance force field
[3] LM - Termination due to local minima
[4] Simulation results of bold values shown in Figure 4.35 to 4.38

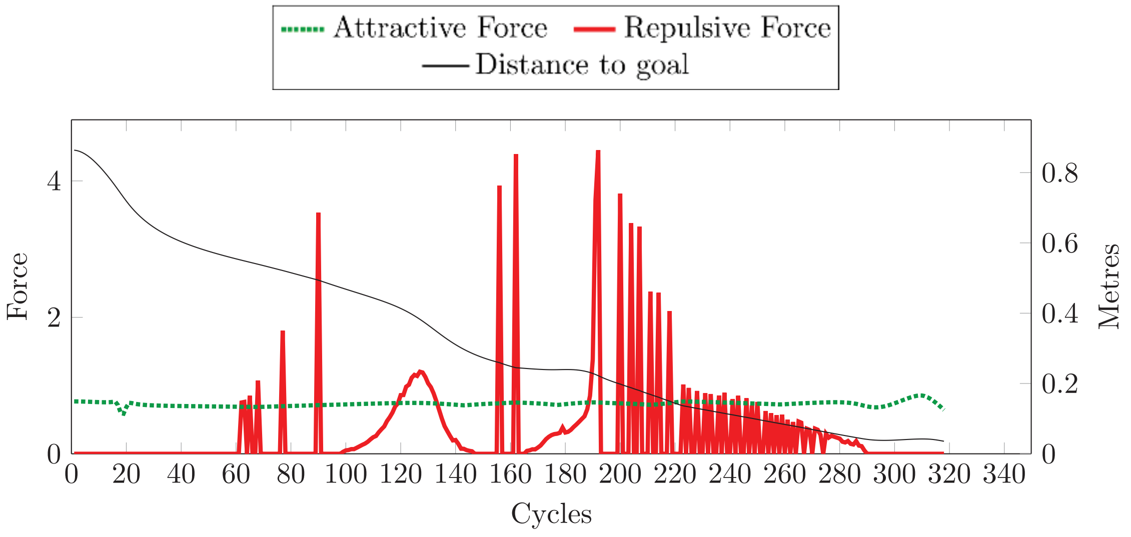
force fields increasing relative to the size of the allowed maximum variance. As a force field increases in size a larger volume was covered increasing the likelihood that repulsive forces were generated. If these repulsive forces acted in the opposite direction to the attractive force the chances of reaching the goal location would be reduced.

The effect of the allowed maximum variance is highlighted on the successfully generated path for case 1 at an allowed maximum translation and rotation of 0.075m and -15.0° and allowed maximum variance of 0.025m (Figures 4.25b and 4.28). While the allowed maximum variance is relatively small, the resulting repulsive forces are twice that of those generated in the simulation with the deterministic model (Figure 4.25a). These forces initially caused the generated path to be repelled further from the floor in comparison to the deterministic case. However, the force fields will reduce in size as the end-effector passes the gap below the partition plate. This is due to the maximum cantilever distance (Equation 3.6) reducing as the end-effector position moves closer to the base reducing the effect of the structural deformation. This is further evident in case 2 with the simulation at an allowed maximum translation and rotation of 0.05m and -10.0° respectively and an allowed maximum variance of 0.125m (Figure 4.33b and 4.36). As the end-effector moves towards the manhole the maximum cantilever distance again shortens reducing the size of the force fields. Without the reduction in size, the force field would be larger than the manhole itself preventing a path being found. While the maximum cantilever distance's effect on the structural uncertainty may improve the chances of success in some simulations, in others it may cause failure.

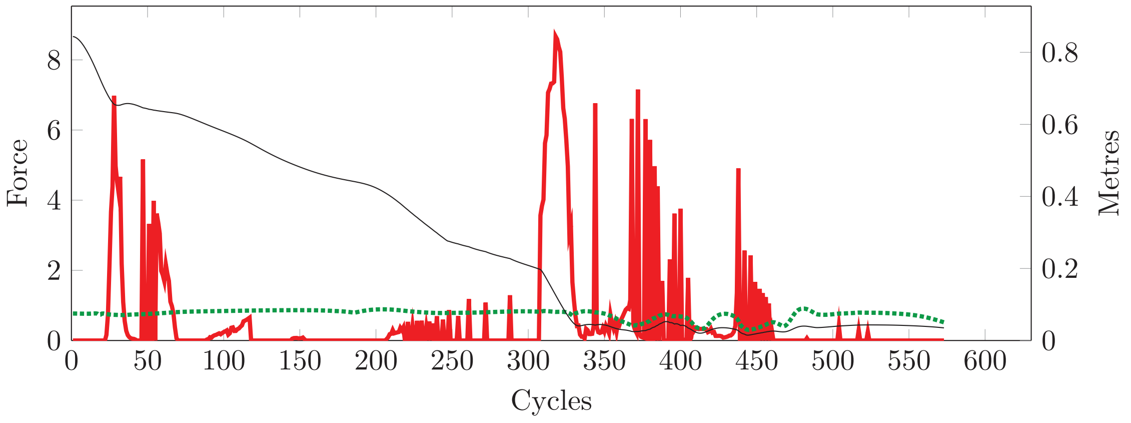
The main cause of failure is due to an obstacle entering the minimum variance force field, occurring in 69.4% and 36.1% of the simulations for cases 1 and 2 respectively. The values of 69.4% and 36.1% are based on the number of times the simulations failed due to obstacles entering the minimum variance force field ("FF" in the table) divided by the number of simulations. This occurred 25 and 13 times for cases one and two respectively. Based on a total of 36 simulations per case this results in 69.4% (25/36) for case one and 36.1% (13/36) for case two. This failure is due to one of two reasons. First, due to the maximum cantilever distance, force fields may increase in size faster than the 3D-PF^2 algorithm can react to potential obstacles generating repulsive forces. In both cases the maximum cantilever distance first decreases upon approach to the gap in the environment then subsequently increases after bypassing the gap as the distance between the end-effector and base decreases and increases respectively. The increase in the maximum cantilever distance causes the force fields to increase in size. Higher values for the allowed maximum variance, in comparison to lower values, will cause force fields to increase to a larger size over the same variation in the maximum cantilever distance. In some cases, if the size of the force field increases rapidly, the 3D-PF^2 algorithm may not be able to compensate for the generated repulsive forces. This is shown in case 1 with an allowed maximum translation and rotation at 0.05m and -15.0° respectively and an allowed maximum variance of 0.05m (Figure 4.25c and 4.29). Proper tuning may reduce the chances of this failure condition from occurring. By increasing the value of the error factor, E_r, for example, the distance before a repulsive force is generated between a force field and potential obstacle is increased allowing a longer time for the 3D-PF^2 algorithm to react. However increasing the error factor may cause additional repulsive forces to be generated which may reduce the chances of successfully finding a path, especially when bypassing a gap.

The second reason an obstacle may enter the minimum variance force field is due to the attractive and repulsive forces acting on the robot in a manner that reduces the chance of collision avoidance. As repulsive forces can act on the robot from multiple directions there may be situations where the repulsive forces either cancel each other out or reduce their effectiveness for avoiding collision. As the attractive force only considers the distance between the goal and the end-effector in its calculation, the resulting force may cause the robot to collide with an obstacle as the repulsive forces are rendered ineffective. This is shown in two simulations for case 1; first at an allowed maximum translation and rotation of 0.025m and -5.0° and variance of 0.075m (Figure 4.26a and 4.30) and second at an allowed maximum translation and rotation of 0.125m and -25.0° and variance of 0.125m (Figure 4.26c and 4.31). In both simulations, the self and environmental repulsive forces at the end-effector act in opposing directions effectively cancelling each other out. A similar situation occurs in case 2 in the simulation with an allowed maximum translation and rotation of 0.025m and -5.0° respectively and a variance of 0.05m (Figures 4.33c and 4.37). As the end-effector bypasses the manhole the repulsive forces generated from the top and bottom of the manhole effectively cancel each other out. This is repeated at an allowed maximum translation and rotation of 0.125m and -25.0° respectively and a variance of 0.125m (Figures 4.34a and 4.38). In these simulations the attractive force continues to act on the robot causing an obstacle to enter the minimum variance force field due to the ineffective repulsive forces.

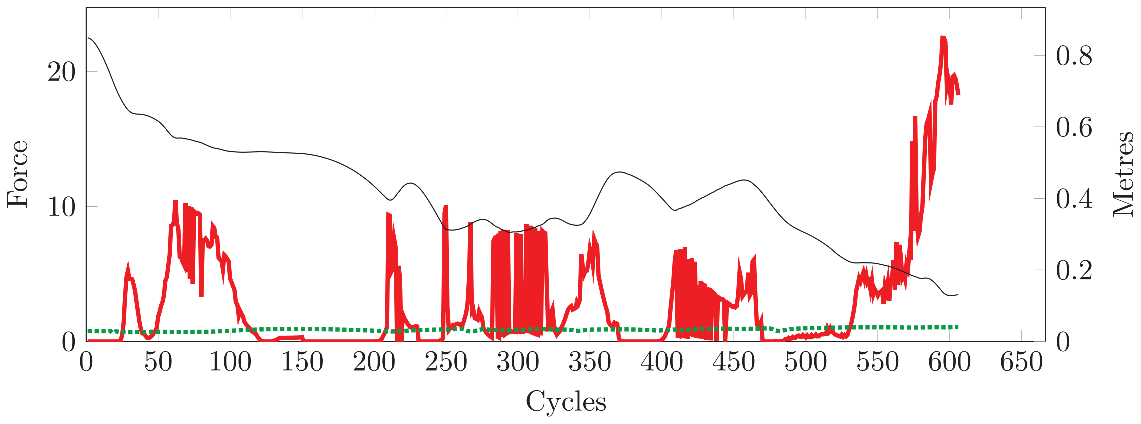
The secondary cause of failure in the simulations is due to the 3D-PF^2 algorithm entering a local minimum. This occurred in 2.7% and 13.8% of the simulations in cases 1 and 2 respectively. A local minimum is entered in case 1 in the simulation at an allowed maximum translation and rotation of 0.075m and -15.0° respectively and an allowed maximum variance of 0.1m (Figure 4.26b and 4.32). The local minimum occurs due to the path continuously repeating itself (Equation 4.69). The repulsive forces push the robot away from both the partition plate and the floor. The robot is moved into the same position again, due to the attractive force, and the motion is repeated. Local minima in case 2 are highlighted in two simulations. The first simulation is at a allowed maximum translation and rotation of 0.075m and -15.0° respectively and allowed maximum variance of 0.075m. The local minimum occurs due to insignificant joint motion over a significant period (Equation 4.67). In Figure 4.39 the final force field can be seen cradled within the manhole, stalling the motion. The associated graph is shown in Figure 4.34b. The second simulation is with an allowed maximum translation and rotation of 0.10m and -20.0° and allowed maximum variance of 0.125m. The local minimum occurs due to insignificant joint velocities over a set period (Equation 4.68). The path is shown in Figure 4.40 with the associated graph in Figure 4.34c. These local minima occur due to the sum of all forces acting on the robot being zero.



(a) Allowed maximum structural translation at 0.0m and rotation 0°, variance at 0.0m (deterministic model).

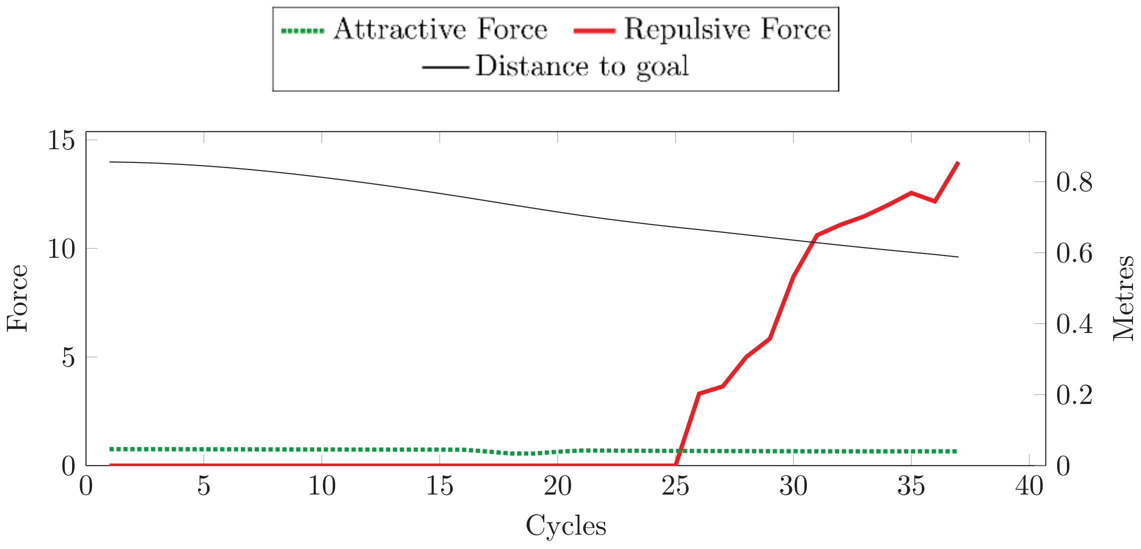


(b) Allowed maximum structural translation at 0.075m and -15.0°, Variance at 0.025m.

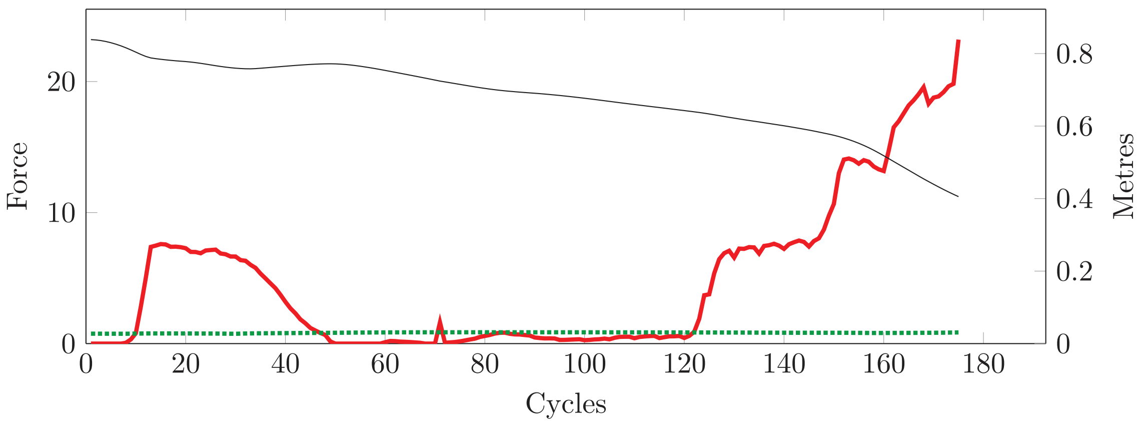


(c) Allowed maximum structural translation at 0.05m and -15.0°, Variance at 0.05m.

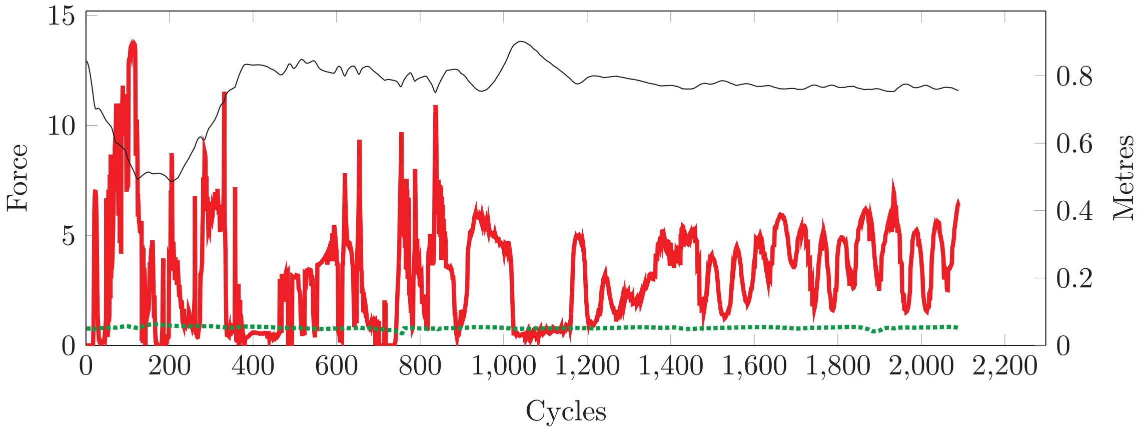
FIGURE 4.25: Case 1: Graphs of forces and the distance to the goal at various allowed maximum structural translations, rotations and variances for the partition plate simulation (part 1).



(a) Allowed maximum structural translation at 0.025m and -5.0°, Variance at 0.075m

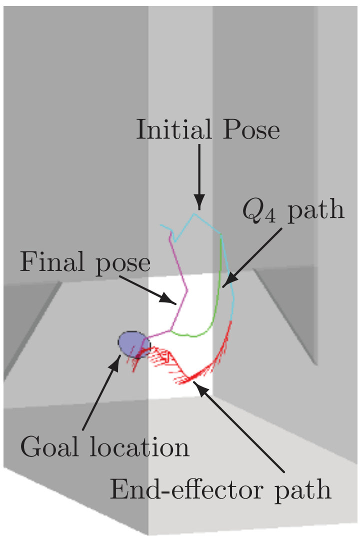


(b) Allowed maximum structural translation at 0.125m and -25.0°, Variance at 0.125m.

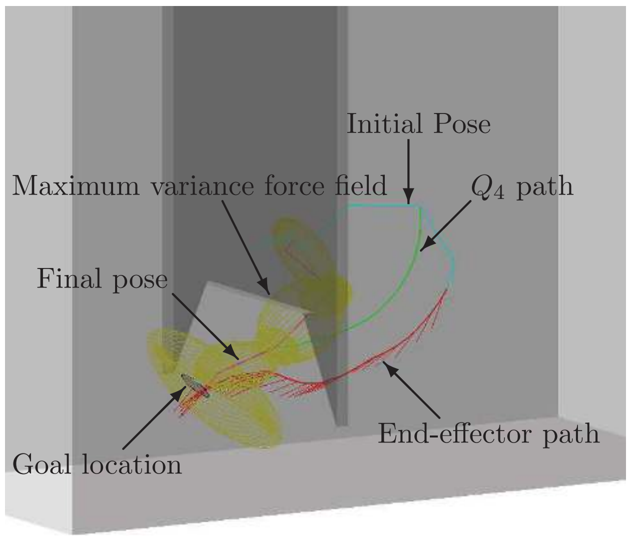


(c) Allowed maximum structural translation at 0.075m and -15.0°, Variance at 0.10m

FIGURE 4.26: Case 1: Graphs of forces and the distance to the goal at various allowed maximum structural translations, rotations and variances or the partition plate simulation (part 2).

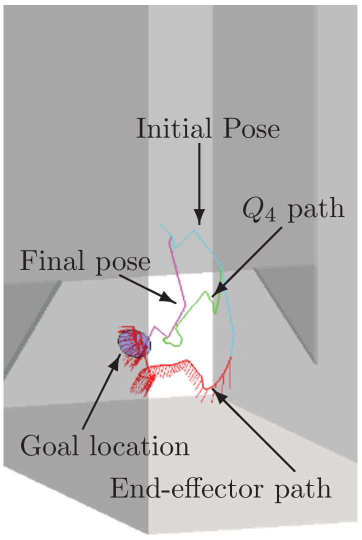


(a) Base side view

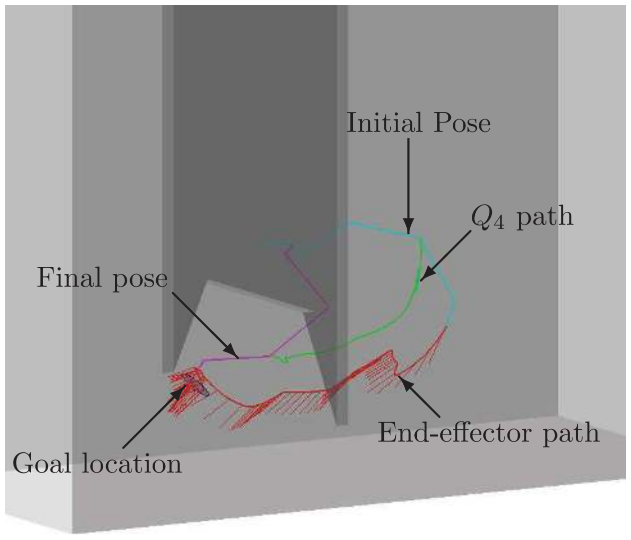


(b) Side view

FIGURE 4.27: Representation of the inchworm robot in the partition plate environment with the allowed maximum structural translation at 0.0m and rotation at 0.0° and variance at 0.0m (deterministic model). The partition plate is shown from the point of view of (a) the base side and (b) the side of the partition plate. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red.



(a) Base side view



(b) Side view

FIGURE 4.28: Representation of the inchworm robot in the partition plate environment with the allowed maximum structural translation at 0.075m and rotation at -15.0° and variance at 0.025m. The partition plate is shown from the point of view of (a) the base side and (b) the side of the partition plate. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.

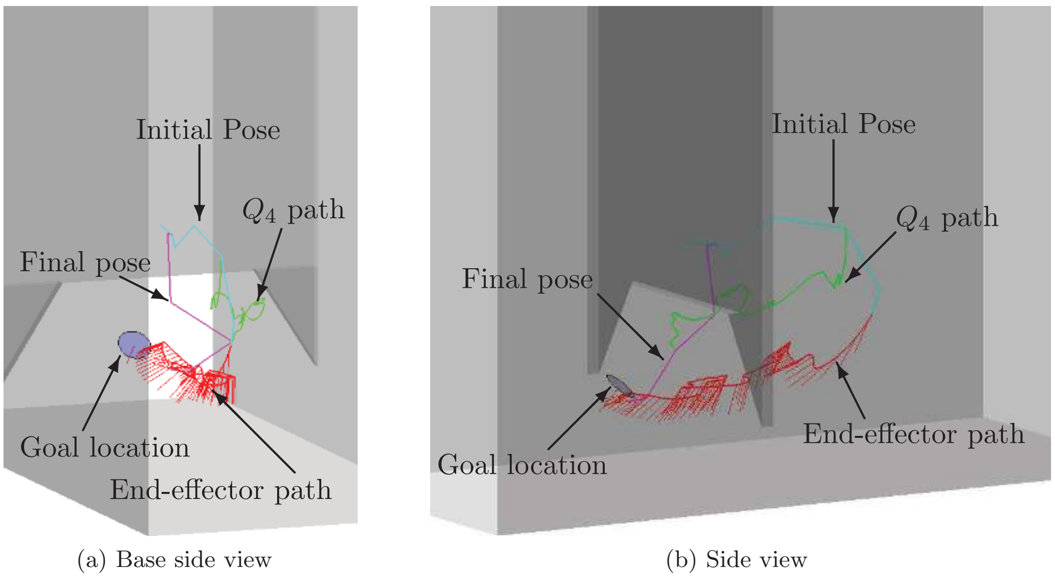


FIGURE 4.29: Representation of the inchworm robot in the partition plate environment with the allowed maximum structural translation at 0.05m and rotation at -15.0° and variance at 0.05m. The partition plate is shown from the point of view of (a) the base side and (b) the side of the partition plate. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.

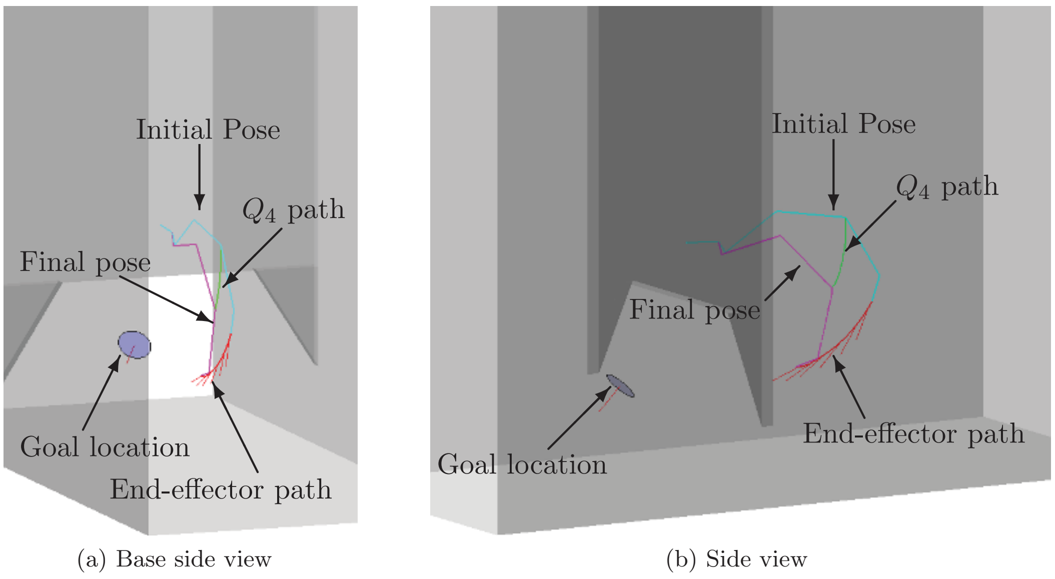
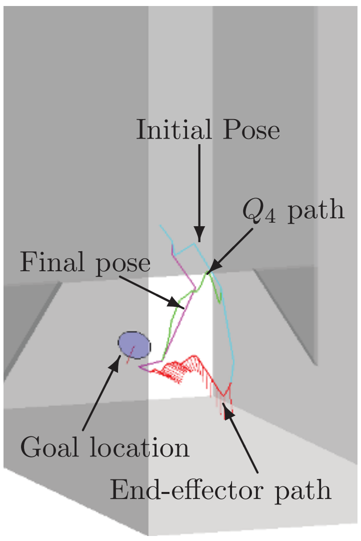
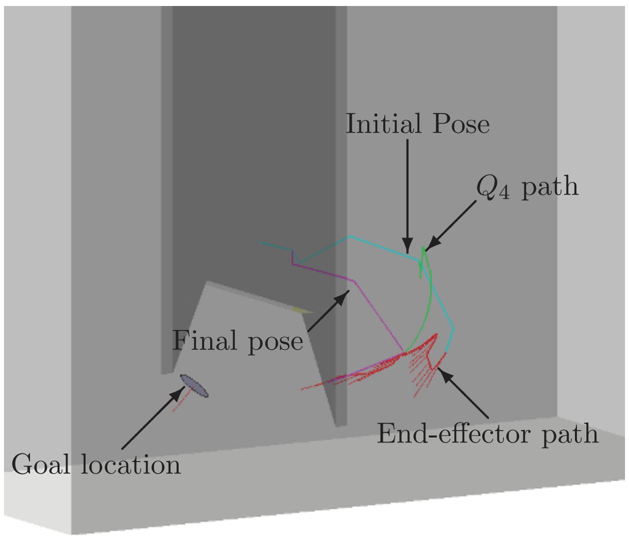


FIGURE 4.30: Representation of the inchworm robot in the partition plate environment with the allowed maximum structural translation at 0.025m and rotation at -5.0° and variance at 0.075m. The partition plate is shown from the point of view of (a) the base side and (b) the side of the partition plate. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.

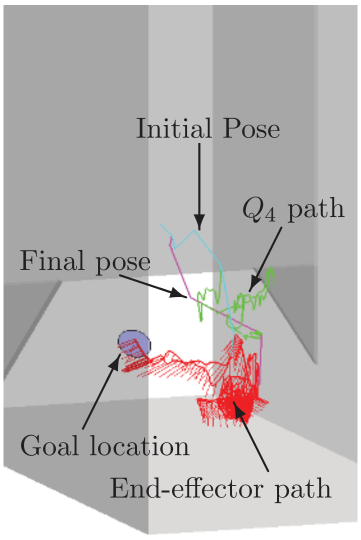


(a) Base side view

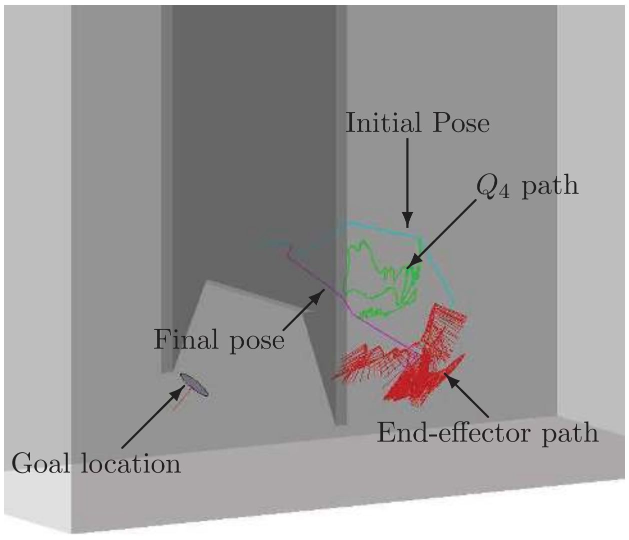


(b) Side view

FIGURE 4.31: Representation of the inchworm robot in the partition plate environment with the allowed maximum structural translation at 0.125m and rotation at -25.0° and variance at 0.125m. The partition plate is shown from the point of view of (a) the base side and (b) the side of the partition plate. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.

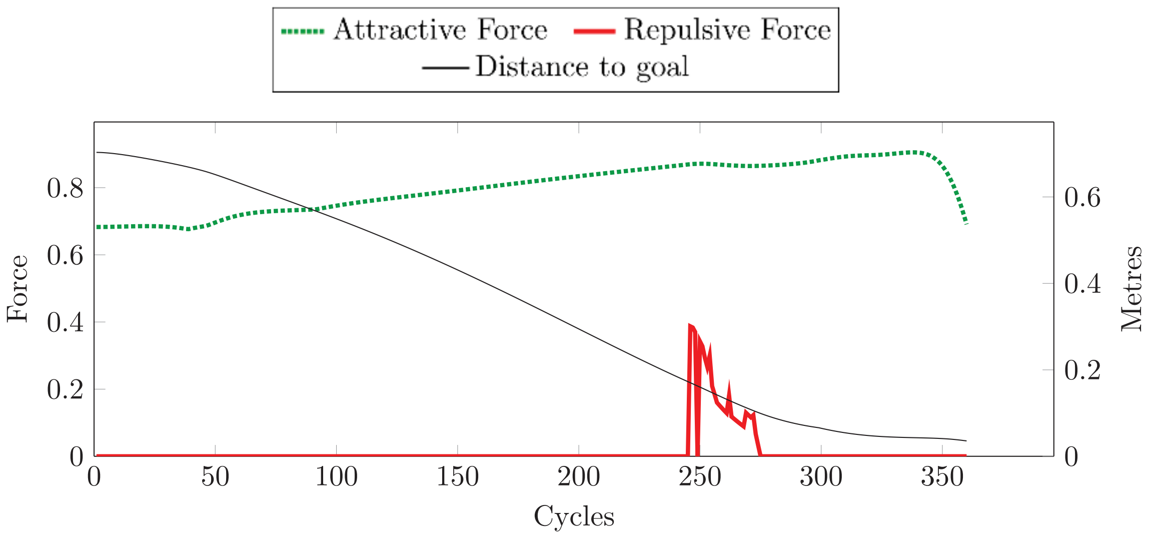


(a) Base side view

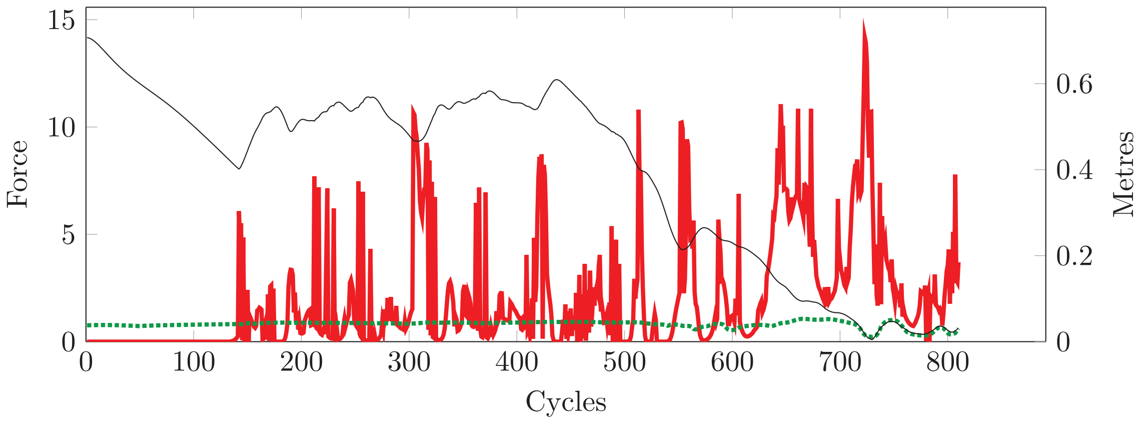


(b) Side view

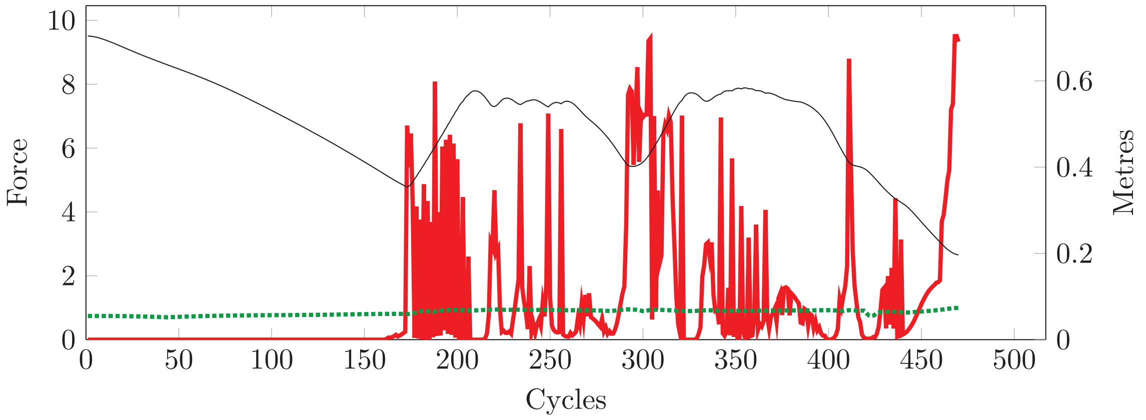
FIGURE 4.32: Representation of the inchworm robot in the partition plate environment with the allowed maximum structural translation at 0.075m and rotation at -15.0° and variance at 0.10m. The partition plate is shown from the point of view of (a) the base side and (b) the side of the partition plate. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.



(a) Allowed maximum structural translation at 0.0m, rotation at 0° and variance at 0.0m (deterministic model).

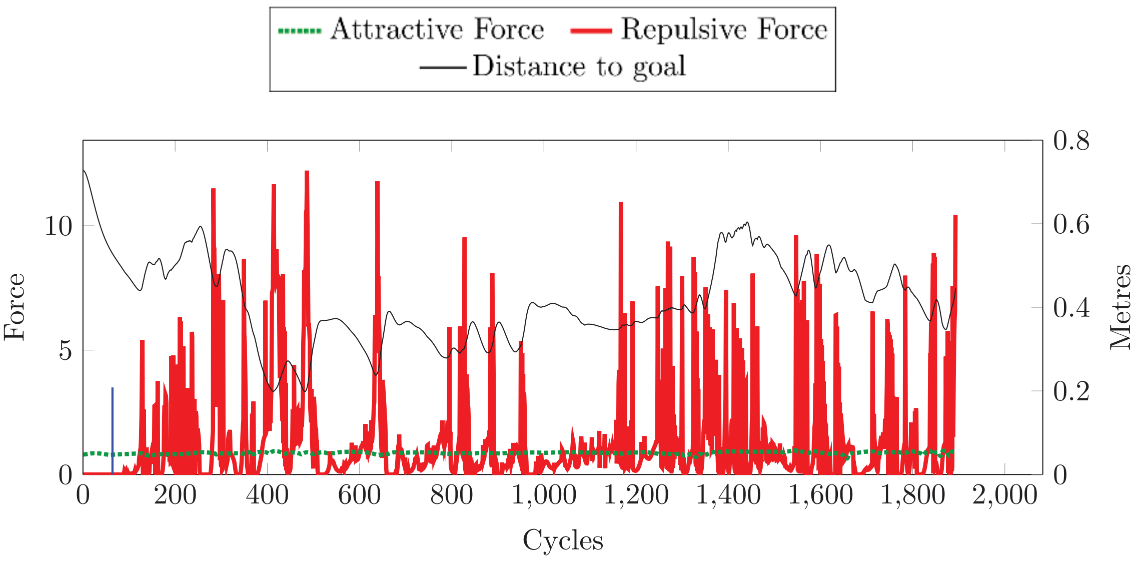


(b) Allowed maximum structural translation at 0.05m, rotation at -10.0° and variance at 0.125m.

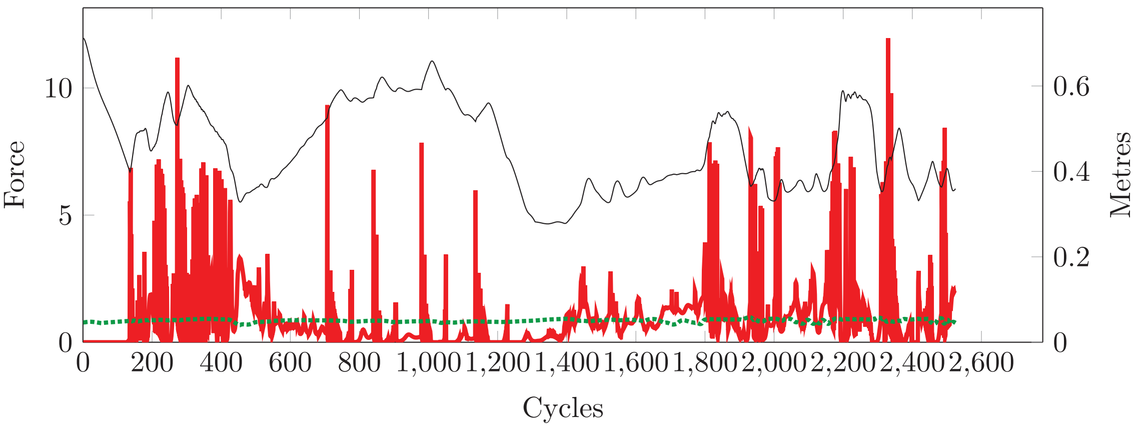


(c) Allowed maximum structural translation at 0.025m, rotation at -5.0° and variance at 0.05m

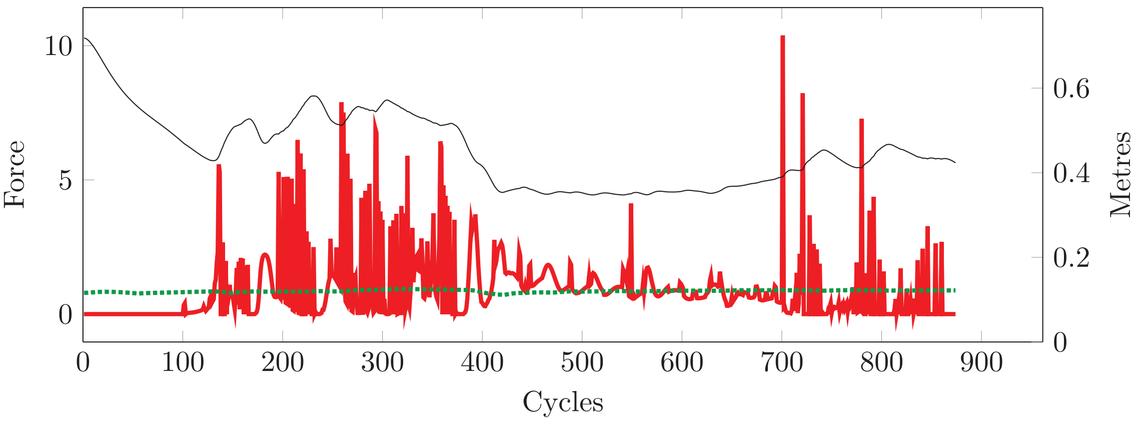
FIGURE 4.33: Case 2: Graphs of forces and the distance to the goal at various allowed maximum structural translation, rotation and variance values for the manhole simulation (part 1).



(a) Allowed maximum structural translation at 0.125m, rotation at -25.0° and variance at 0.125m.

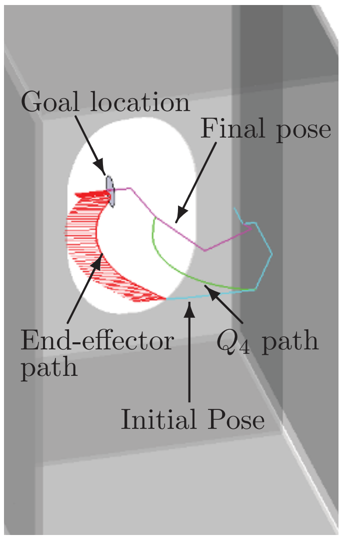


(b) Allowed maximum structural translation at 0.075m, rotation at -15.0° and variance at 0.075m.

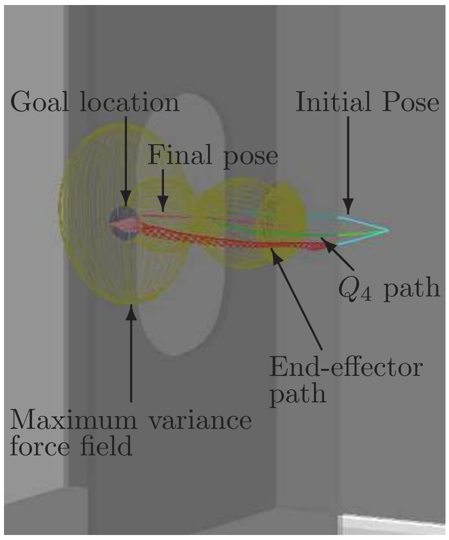


(c) Allowed maximum structural translation at 0.10m, rotation at -20.0° and variance at 0.125m

FIGURE 4.34: Case 2: Graphs of forces and the distance to the goal at various allowed maximum structural translation, rotation and variance values for the manhole simulation (part 2).

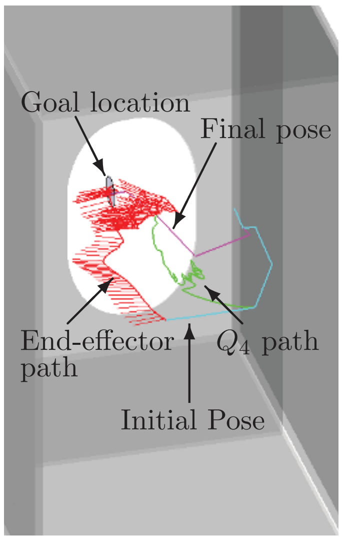


(a) Front view

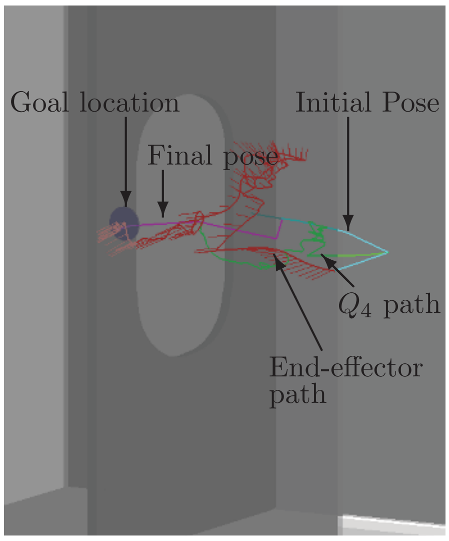


(b) Side view

FIGURE 4.35: Representation of the inchworm robot in the manhole environment with the allowed maximum structural translation at 0.0m, rotation at 0.0° and variance at 0.0m (deterministic model). The manhole is shown from the point of view of (a) the base side and (b) the side of the manhole. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red.

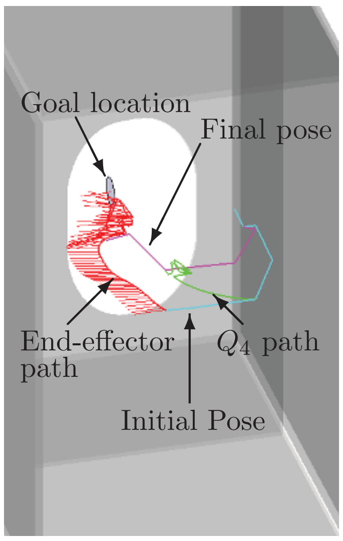


(a) Front view

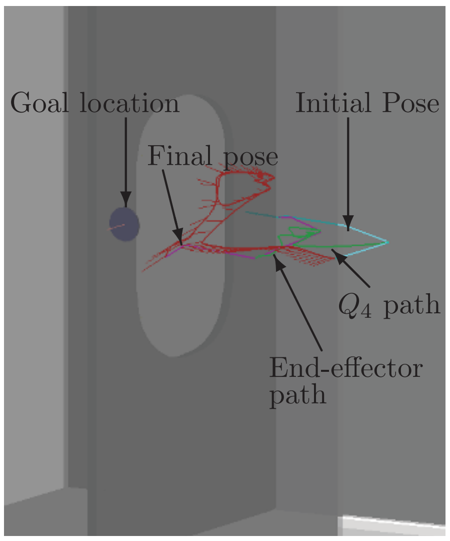


(b) Side view

FIGURE 4.36: Representation of the inchworm robot in the manhole environment with the allowed maximum structural translation at 0.05m, rotation at -10.0° and variance at 0.125m. The manhole is shown from the point of view of (a) the base side and (b) the side of the manhole. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.

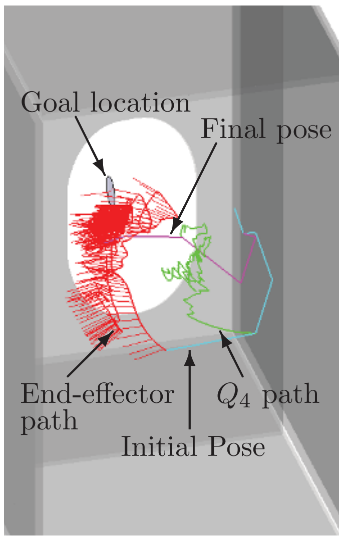


(a) Front view

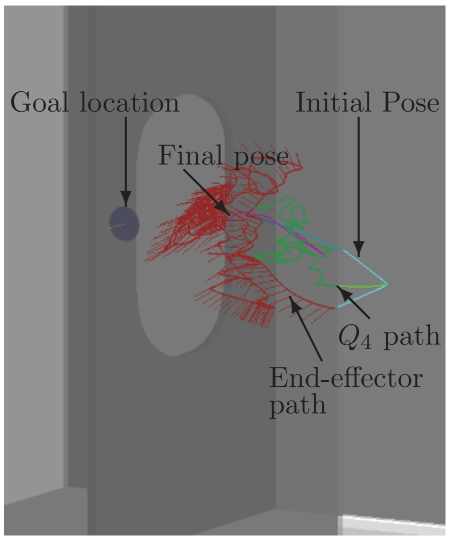


(b) Side view

FIGURE 4.37: Representation of the inchworm robot in the manhole environment with the allowed maximum structural translation at 0.025m, rotation at -5.0° and variance at 0.05m. The manhole is shown from the point of view of (a) the base side and (b) the side of the manhole. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.

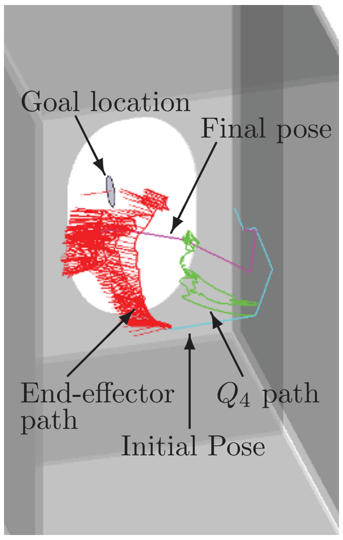


(a) Front view

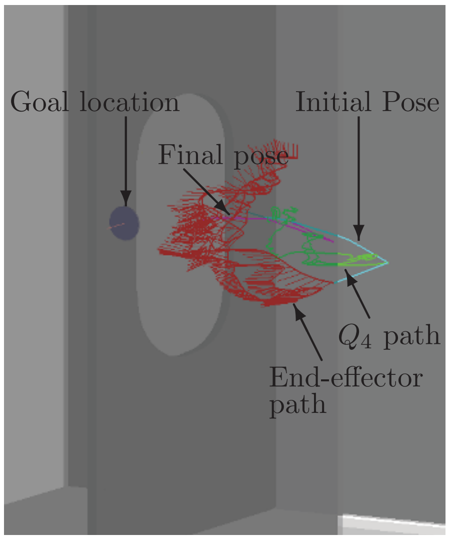


(b) Side view

FIGURE 4.38: Representation of the inchworm robot in the manhole environment with the allowed maximum structural translation at 0.125m, rotation at -25.0° and variance at 0.125m. The manhole is shown from the point of view of (a) the base side and (b) the side of the manhole. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.

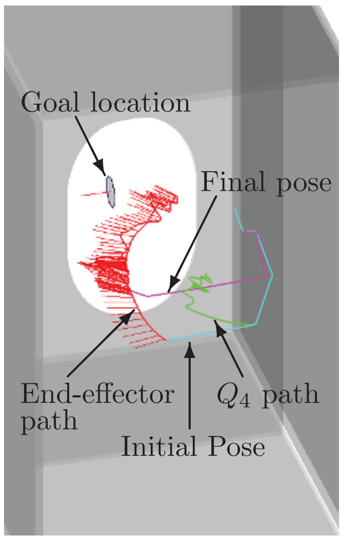


(a) Front view

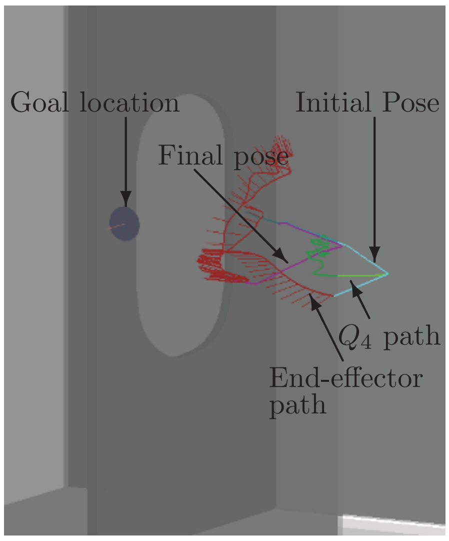


(b) Side view

FIGURE 4.39: Representation of the inchworm robot in the manhole environment with the allowed maximum structural translation at 0.075m, rotation at -15.0° and variance at 0.075m. The manhole is shown from the point of view of (a) the base side and (b) the side of the manhole. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.



(a) Front view



(b) Side view

FIGURE 4.40: Representation of the inchworm robot in the manhole environment with the allowed maximum structural translation at 0.10m, rotation at -20.0° and variance at 0.125m. The manhole is shown from the point of view of (a) the base side and (b) the side of the manhole. The initial pose is shown in cyan, the final pose in magenta, the path of the fourth joint in green and the end-effector path and orientation of the end-effector along the path in red. The force field is omitted for clarity.

4.3.3 Verification of the 3D-PF^2 Algorithm in Experiments

The 3D-PF^2 algorithm has been used for motion planning of a 7DOF inchworm robot in real application scenarios. The algorithm was run using the allowed maximum structural and hand position uncertainties found on the 7DOF inchworm robot (Table 4.7). Three environments are considered: a tunnel environment with inter-spaced obstacles, a partition plate environment and a manhole environment. Paths were developed offline using the 3D-PF^2 algorithm, then the paths were implemented on the 7DOF inchworm robot in the three environments.

4.3.4 Experimental Verification of the 3D-PF^2 Algorithm for Attachment in a Tunnel Environment

This experiment is used to verify the surface attachment. The 3D-PF^2 algorithm was used to develop a path for the 7DOF inchworm robot to attach to a surface. The environment has a number of obstacles surrounding the inchworm robot (Figure 4.41). The path is developed from an initial pose to the preparatory approach distance (Stage 1 of the attachment procedure) then to the goal location (Stage 2 of the attachment procedure).

Figures 4.42a shows the distance to the surface and variation between the current end-effector normal and the goal normal based on the generated path. Figure 4.42b shows the distance to the surface and variation between the current end-effector normal to the goal normal measured from joint feedback of the inchworm robot. Both Figures show the attractive and repulsive forces over the generated path. The difference in the distance to the goal location between the two figures can be accounted for by the path smoothing performed by the inchworm robot controller.

Table 4.7: Table of allowed maximum structural and hand position uncertainty values used during experiments.

Allowed maximum structural translation (m)	0.065
Allowed maximum structural rotation (°)	8.5
Allowed maximum structural translation variance (m)	0.04
Allowed maximum hand position variance (m)	0.05

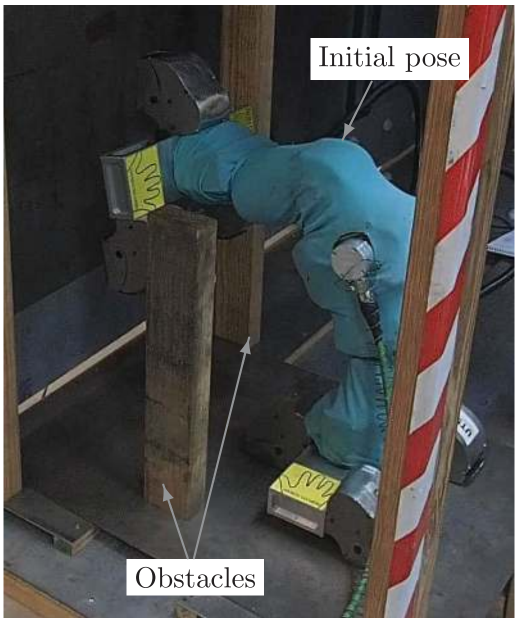


FIGURE 4.41: The tunnel environment used to verify the surface attachment.

The preparatory approach distance was reached with no generated repulsive forces. Upon approach to the goal location during Stage 2 of the attachment procedure, repulsive forces are generated due to the force field surrounding the end-effector interacting with the obstacles closest to the wall and the floor (Figure 4.43e). As the end-effector moves towards the goal location the force field size increases and, in turn, generates a higher repulsive force which pushes the end-effector away from the obstacles (Figure 4.43f). The end-effector then moves further away from the surface which reduces the size of the force field. While this does cause oscillations in Figures 4.42a and 4.42b a successful path is still found to the surface goal location (Figures 4.43g to 4.43i). This is due to two factors. First, the rotational component of the attractive force causes the end-effector's orientation to align with the goal normal; and secondly, the variation to the translational component of the attractive force in stage 2 of the attachment procedure whereby the attractive force is applied to the end-effector in the direction of the goal normal ensures that surface contact is made.

The repulsive forces in Figures 4.42a and 4.42b are higher than the generated attractive forces for two reasons. First they ensure that any individually generated repulsive force may overcome the attractive force as seen by the smaller peaks. Secondly, as multiple repulsive forces may occur, the sums of these forces are reflected through the higher peaks

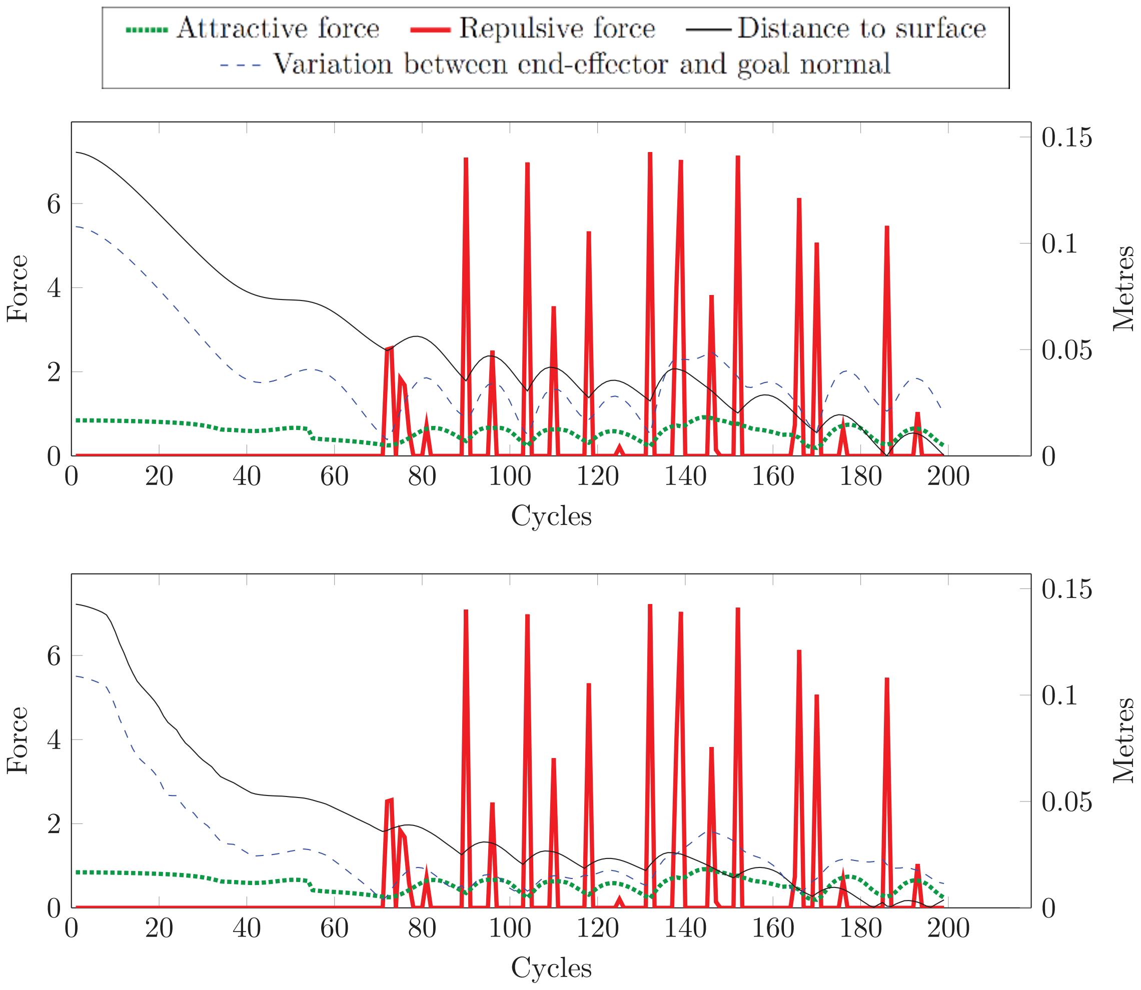
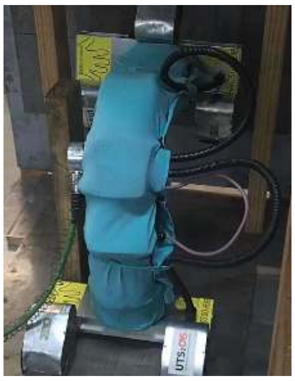


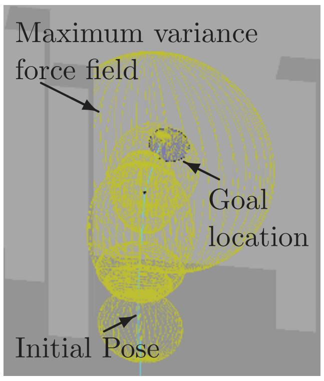
FIGURE 4.42: 4.42a Graph of the generated forces, the distance to the goal and the end-effector's distance to the goal normal generated for surface attachment. 4.42b Graph of the generated forces and the distance to the goal and the end-effector's distance to the goal normal from feedback from the joint angles on the inchworm robot. Here the allowed maximum structural translation is 0.065m, rotation is 8.5° and variance is 0.04m while the allowed maximum hand position variance is 0.05m.

on the graphs. These peaks may oscillate due to the attractive force causing an obstacle to re-enter the force fields to generate similar repulsive forces after previously being repelled.

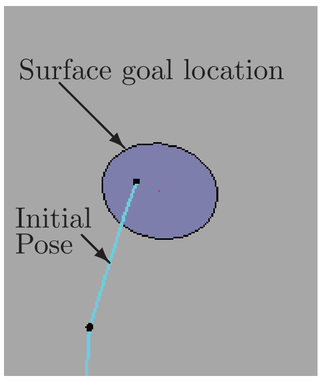
4.3.5 Experimental Verification of the 3D-PF^2 Algorithm in the Partition Plate Environment

This experiment is used to verify the 3D-PF^2 algorithm in the partition plate environment (Figure 4.45a and 4.45b). The 3D-PF^2 algorithm develops a path for the inchworm robot

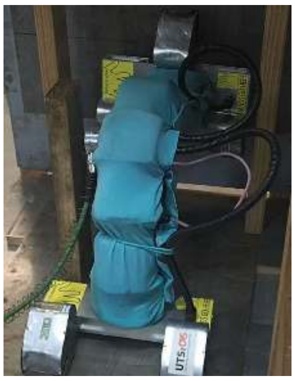

(a) Initial pose.

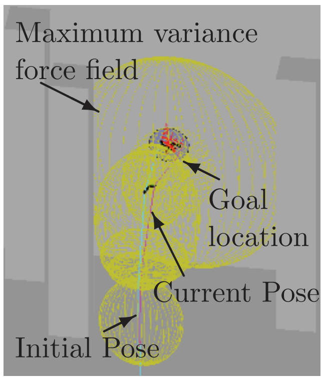


(b) Initial pose with force fields.

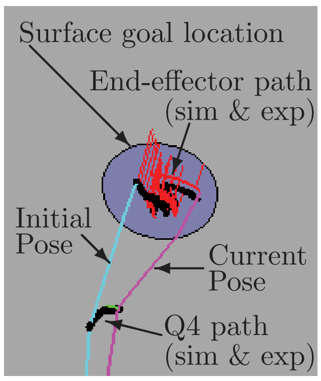


(c) Skeleton of the robot at the initial pose.

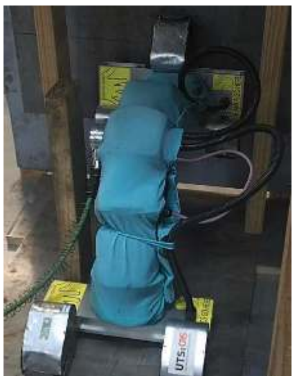

(d) Pose at the furthest distance from the goal normal.

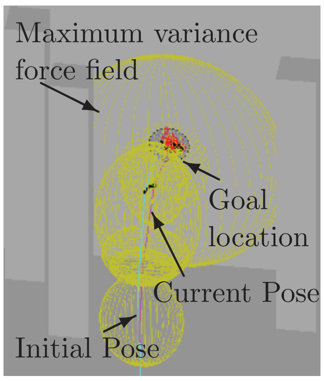


(e) Pose at the furthest distance from the goal normal with force fields.

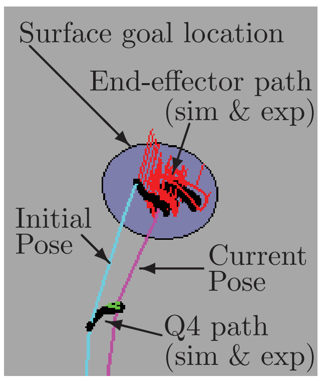


(f) Path followed to the furthest distance.


(g) Final pose.



(h) Final pose with force fields.



(i) Path followed to the final pose.

FIGURE 4.43: The inchworm robot at (a) to (c) the initial pose, (d) to (f) the pose furthest from the goal normal and (g) to (i) the final pose. The yellow ellipsoids are the maximum variance force fields, the cyan line is the initial robot pose, the green line is the position of the fourth joint over the generated path, the red lines are the position and orientation of the end-effector over the generated path, and the magenta lines show the given pose. The black dots show the output of the fourth joint and end-effector from the experiment over the respective paths.

to move from one side of the partition plate to the goal location on the opposite side.

Figures 4.44a shows the distance to the goal based on the path generated by the algorithm while Figure 4.44b shows the distance to the goal measured from the joint feedback of the inchworm robot. Both Figures show the attractive and repulsive forces along the path. The difference in the distance to the goal location between the two figures can be accounted for by the trajectory smoothing performed by the inchworm robot controller.

The path generated is shown in Figure 4.45. From the initial pose (Figure 4.45a and 4.45b),

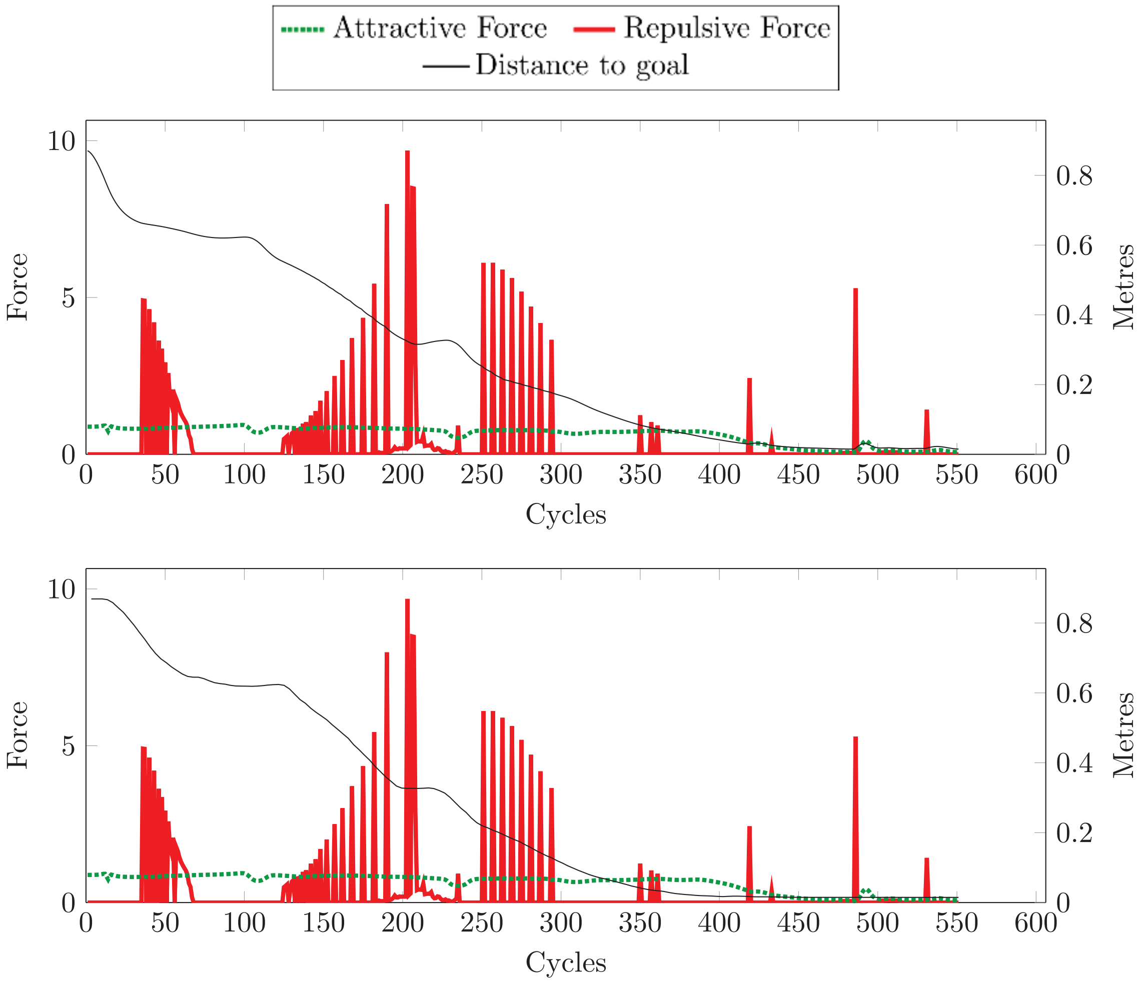


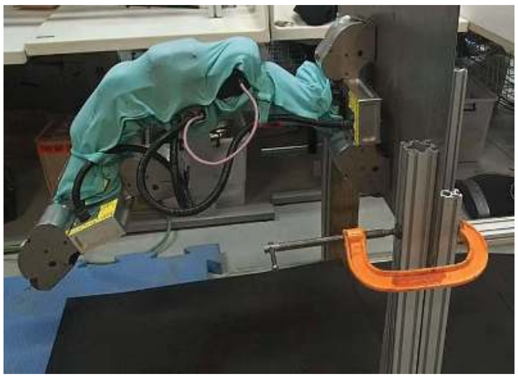
FIGURE 4.44: 4.44a Graph of the generated forces and the distance to the goal for a path through the derived partition plate environment. 4.44b Graph of the generated forces and the distance to the goal from feedback from the joint angles on the inchworm robot through the partition plate environment. Here the allowed maximum structural translation is 0.065m, rotation is 8.5° and variance is 0.04m while the allowed maximum hand position variance is 0.05m.

repulsive forces are generated as the final link moves closer to the floor (Figure 4.45c and 4.45d). The 3D-PF^2 algorithm continues until the partition plate enters the force field surrounding the final link. While this does repel the robot, it causes additional repulsive forces due to potential collision with the floor. The combination of these repulsive forces and the attractive force allows a path to be successfully found to the goal location (Figure 4.45e and 4.45f). The final pose and path generated from either side of the partition plate are shown in Figure 4.46.

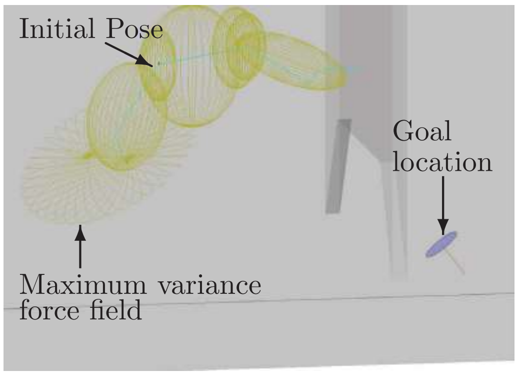
The repulsive forces in Figures 4.44a and 4.44b are higher than the generated attractive forces for the same two reasons as explained previously; to ensure that individually generated repulsive forces are higher than the attractive force and multiple repulsive forces are additive at the same program cycle. However, in these graphs it can be seen that the oscillating peaks of the repulsive forces increase and decrease over a number of program cycles. This is due to the interaction between attractive and repulsive forces. As the repulsive forces are pushed away from the floor, in the first set of oscillations, the attractive force causes the robot to move back towards the floor, due to the dynamics of the robot, which again generates repulsive forces. As this continues the floor does not penetrate into the force field as far each time, slowly allowing a pose to be reached which freely allows the attractive force to move the robot away from the floor. The second set of oscillations are again due to the interaction of the forces although, instead, here the repulsive forces increase while trying to bypass the gap below the partition plate. Initial contact with the partition plate causes a small repulsive force. However, as the attractive force continually causes the robot to move back towards the partition plate, each time at a slightly different pose, higher repulsive forces are generated. These repulsive forces, in combination with the attractive force, cause the robot to eventually move into a position to bypass the gap.

4.3.6 Experimental Verification of the 3D-PF^2 Algorithm in the Manhole Environment

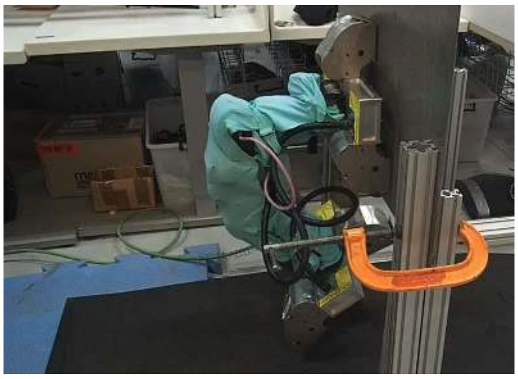
This experiment is used to verify the 3D-PF^2 algorithm in the manhole environment (Figure 4.48a and 4.48b). With the robot positioned on one side of the manhole, the



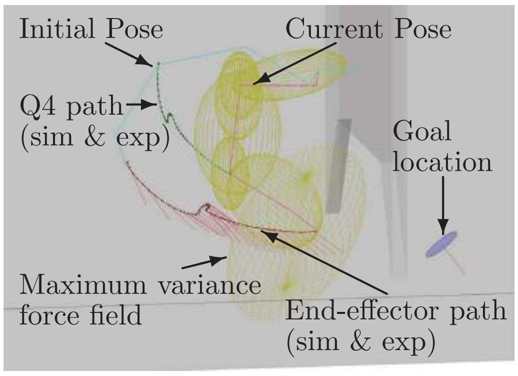
(a) Inchworm robot at the initial pose in the partition plate environment.



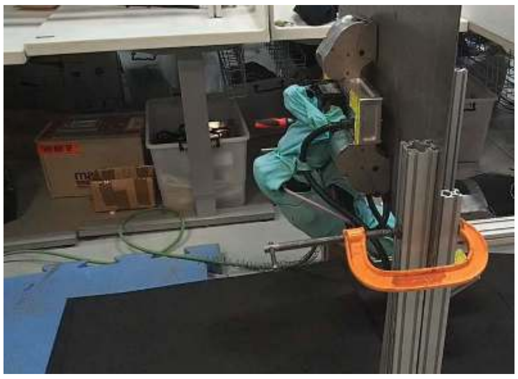
(b) Representation of the inchworm robot at the initial pose in the derived partition plate environment.



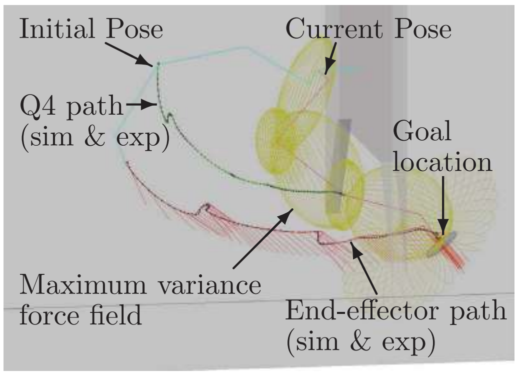
(c) Inchworm robot in a pose prior to repulsive force pushing the end-effector away from the partition plate.



(d) Pose prior to repulsive force pushing the end-effector away from the partition plate.

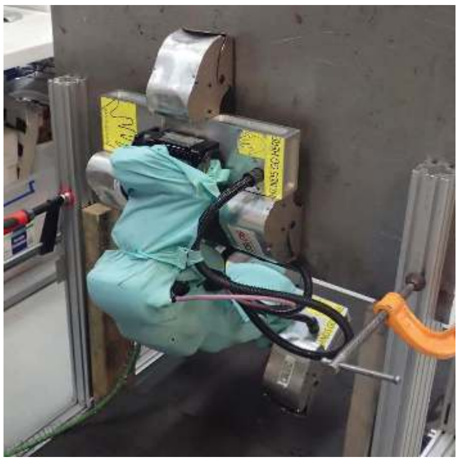


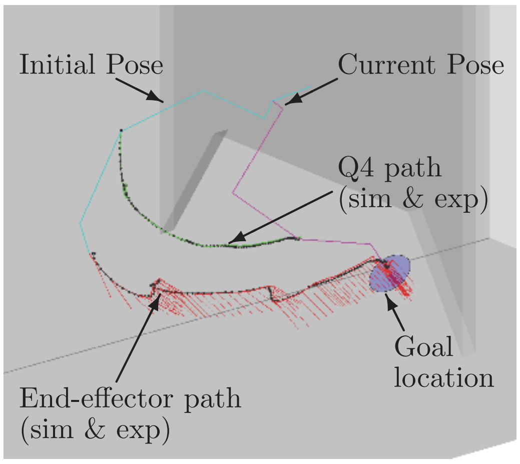
(e) Final inchworm robot pose in the partition plate environment.



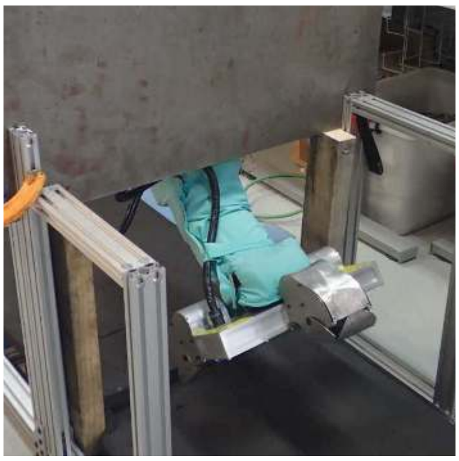
(f) Final pose in the derived partition plate environment.

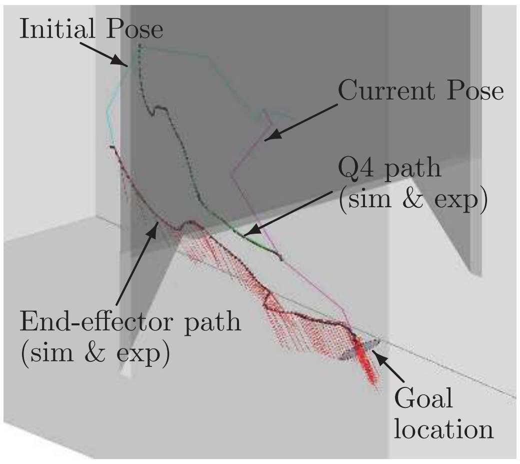
FIGURE 4.45: The inchworm robot (a) and (b) at the initial pose, (c) and (d) the pose prior to repulsive forces pushing the robot away from the partition plate and (e) and (f) the final pose. The yellow ellipsoids are the low maximum variance force fields, the cyan line is the initial robot pose, the blue lines are the position of the joints over the generated path, the red lines are the position and orientation of the end-effector over the generated path, and the magenta lines show the given pose. The black dots show the output of the fourth joint and end-effector from the experiment over the respective paths.


(a) View from the base side of the robot in the partition plate.



(b) Path through the derived partition plate viewed from the base side of the robot.


(c) View from the goal side of the partition plate.



(d) Path through the derived partition plate viewed from the goal side of the environment.

Figure 4.46: The inchworm robot in the final pose on either side of the partition plate. The cyan line is the initial robot pose, the blue lines are the position of the joints over the generated path, the red lines are the position and orientation of the end-effector over the generated path, and the magenta lines show the final pose. The black dots show the output of the fourth joint and end-effector from the experiment over the respective paths.

3D-PF^2 algorithm develops a path for the robot to move from one side to the opposite side of the manhole.

Figure 4.47a shows the distance to the goal along the generated path while Figure 4.47b shows the distance to the goal measured from the joint feedback of the inchworm robot. Both Figures show the attractive and repulsive forces along the path. The difference in the distance to the goal location between the two figures can be accounted for by the trajectory smoothing performed by the inchworm robot controller.

The path generated is shown in Figure 4.48. From the initial pose (Figure 4.48a and 4.48b), a path is generated through the manhole (Figure 4.48c and 4.48d). Repulsive forces are generated due to the proximity of the force fields to the side of the manhole. These repulsive forces act in a similar direction to the attractive force which assists in finding a successful path through the manhole itself. The 3D-PF^2 algorithm then successfully finds

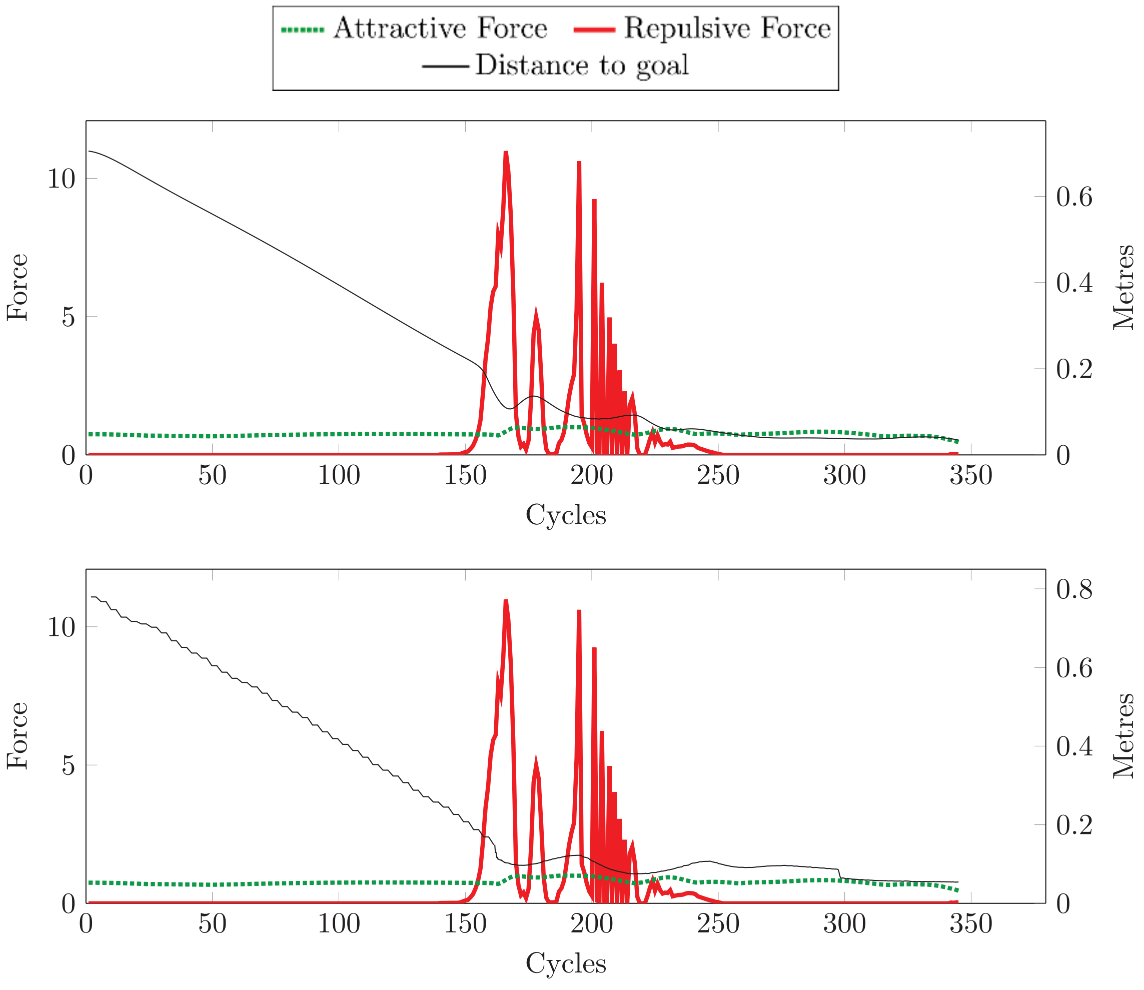
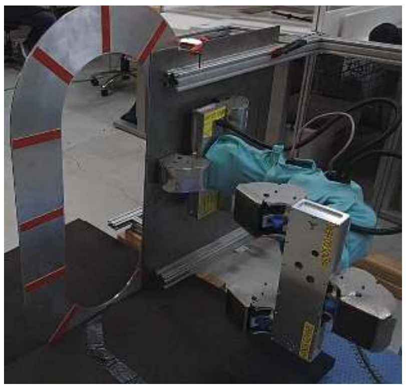
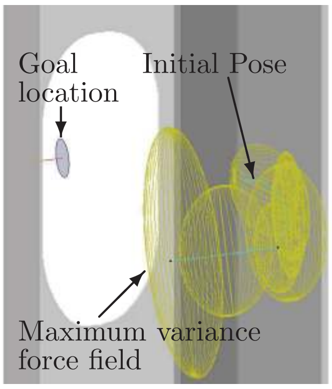
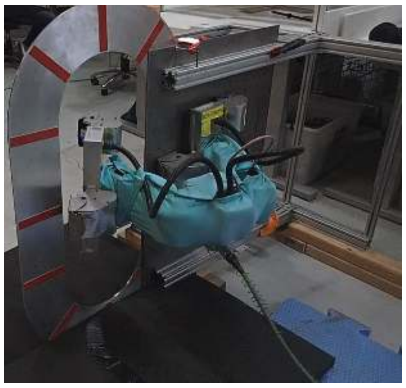


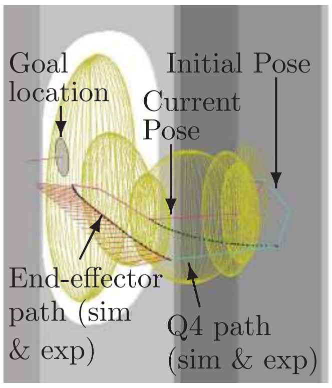
FIGURE 4.47: 4.47a Graph of the generated forces and the distance to the goal for a path through the derived manhole environment. 4.47b Graph of the generated forces and the distance to the goal from feedback from the joint angles on the inchworm robot through the manhole environment. Here the allowed maximum structural translation is 0.065m, rotation is 8.5° and variance is 0.04m while the allowed maximum hand position variance is 0.05m.


(a) Inchworm robot at the initial pose in the manhole environment.

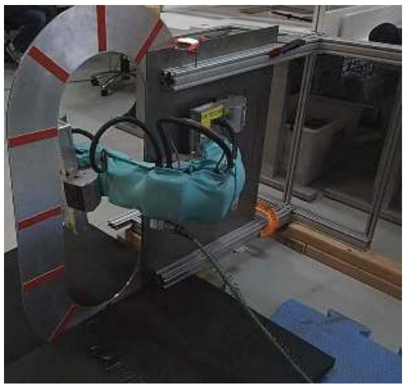


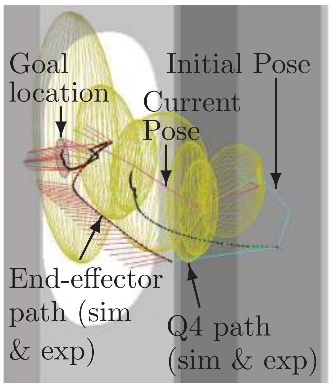
(b) Representation of the inchworm robot at the initial pose in the derived manhole environment.


(c) Inchworm robot prior to entering through the manhole.



(d) Representation of the inchworm robot prior to entering through the derived manhole.


(e) Final inchworm robot pose in the manhole environment.

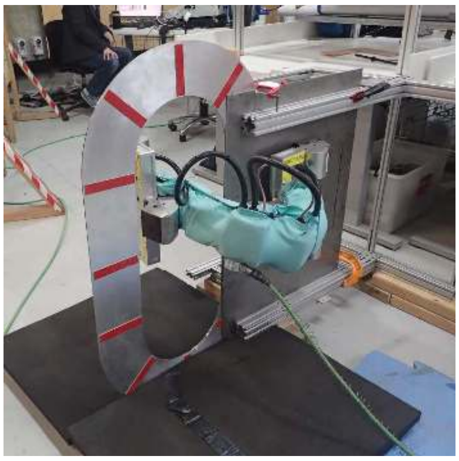


(f) Representation of the inchworm robot in the final pose in the derived manhole environment.

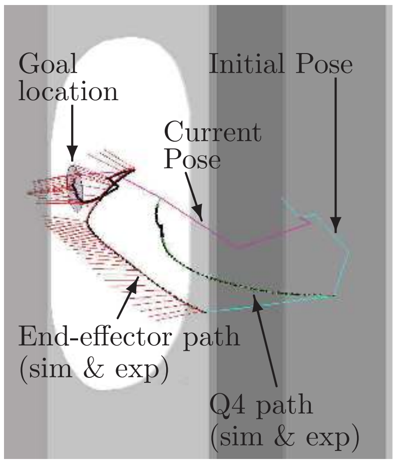
FIGURE 4.48: The inchworm robot (a) and (b) at the initial pose, (c) and (d) the pose prior to entering through the manhole and (e) and (f) the final pose. The yellow ellipsoids are the low maximum variance force fields, the cyan line is the initial robot pose, the blue lines are the position of the joints over the generated path, the red lines are the position and orientation of the end-effector over the generated path, and the magenta lines show the given pose. The black dots show the output of the fourth joint and end-effector from the experiment over the respective paths.

a path to the goal location (Figure 4.48e and 4.48f). The final pose and path generated from either side of the manhole is shown in Figure 4.49.

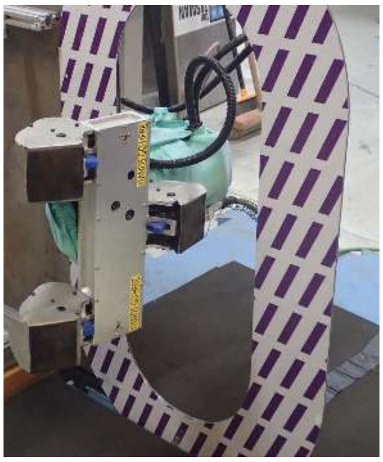
The repulsive forces in Figures 4.47a and 4.47b are higher than the generated attractive forces, again, to ensure that individually generated repulsive forces are higher than the attractive force and multiple repulsive forces are additive at the same program cycle. The



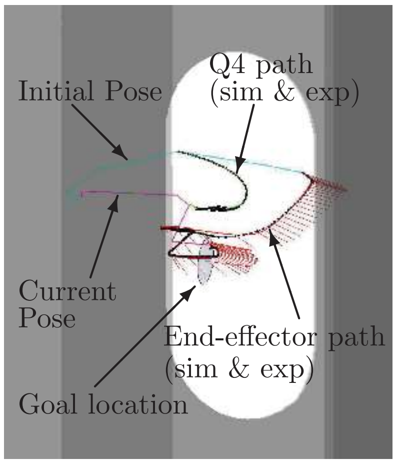
(a) View of the base side of the inchworm robot in the final pose in the manhole.



(b) Path through the manhole viewed from the base side of the robot.



(c) View from the goal side of the manhole.



(d) Path through the manhole viewed from the goal side of the environment.

FIGURE 4.49: The inchworm robot in the final pose on either side of the manhole. The cyan line is the initial robot pose, the blue lines are the position of the joints over the generated path, the red lines are the position and orientation of the end-effector over the generated path, and the magenta lines show the final pose. The black dots show the output of the fourth joint and end-effector from the experiment over the respective paths.

first set of repulsive forces occur due to the interaction between the force field surrounding the end-effector and the manhole as a path is found through the gap. The total repulsive force increases as the force field surrounding the end-effector enters further into the manhole. This causes higher generated repulsive forces as the walls of the manhole penetrate further into the force field. The second set of repulsive forces occur after the end-effector has bypassed the manhole and the force fields surrounding the central links of the robot interact with the wall of the manhole closest to the robot base.

4.4 Discussion

In the simulations the failure rate is increased when the allowed maximum variance is increased. In many cases the force field may have been able to pass the environmental obstacle if it was orientated correctly prior to motion. For example, in the partition plate environment if the end-effector was orientated towards the floor prior to motion an increase in the variance would increase the radius of the force field surrounding the end-effector along its minor axis. As the resulting force field would be smaller than the partition plate's gap, the chances of successful navigation would increase. If the end-effector was instead orientated towards the side wall of the partition plate environment, the resulting force field would not allow the robot to bypass the partition plate. Without careful selection of the start pose and goal location, the 3D-PF^2 algorithm can not actively decide to move the end-effector to a specific orientation to ensure obstacles are bypassed. The combination of repulsive and attractive forces, in some cases, may cause the end-effector to move to the desired orientation although this is not guaranteed. In other cases the forces may actually cause a well aligned end-effector to change its orientation preventing a successful path being found. Prior to commencing motion it is difficult to predict the outcome.

It was previously mentioned that a Gaussian distribution was the most likely representation for the uncertainties. However, in this thesis, allowed maximum values are used to represent the uncertainties and to develop the force fields. In comparison to potentially using Gaussian distributions, allowed maximum values would produce larger force fields. Larger force fields would allow for more conservative, safer navigation; however, successful paths are less likely to be found due to their increased size.

The same parameters are used in all simulations with only the uncertainty being varied. These parameters highly influence the path generated by the 3D-PF^2 algorithm. The error factor, E_r, greatly influences the repulsive forces. If the value is high, repulsive forces may prevent the robot from passing through gaps in the environment; if it is too low, the 3D-PF^2 algorithm may not be able to react fast enough to an obstacle. Hence a higher value for the error factor may result in a higher chance to avoid collisions but may also reduce the chances for a successful path being found. The attractive and repulsive force amplitudes, $K_{att.T|R}$ and K_f, dictate the maximum forces and should be considered relative to each other. If the attractive force amplitudes are low relative to the repulsive forces, the 3D-PF^2 algorithm may not be able to overcome the repulsive forces preventing further motion. Conversely, if the repulsive forces are low relative to the attractive force, there is a greater chance that the attractive force will overpower the repulsive forces and cause a collision. The difference between the values of the minimum and maximum variance factors, ξ_{min} and ξ_{max}, affects the distance an obstacle can penetrate the maximum force field before the minimum force field is penetrated. The greater the difference between these two values the higher the chances that collisions will be avoided, with higher values for the maximum variance factor further increasing the chances. However, the higher the maximum variance factor the less likely successful paths would be found due to the larger sized force fields. If the value of the minimum variance factor is too high the minimum force field may be unnecessarily large, increasing the chances that it would be penetrated and the algorithm terminating in a failure. The minimum variance factor should be valued based on the uncertainty. At a value of 0% the resulting force field will only encompass the deterministic model, while at 100% all values represented by the variance in the model will be encompassed. Values less than 100% may be necessary to ensure successful paths are found, especially through relatively small gaps. The exact value for these parameters would depend on the type of robot, the environment and the confidence in the system uncertainty.

Another method that can help with the selection of the parameter values is by analysing the environment prior to motion commencing. The 3D-PF^2 algorithm would need to then be given a series of waypoints along a planned path. This will be explored in the next chapter.

4.5 Conclusions

This chapter described a 3D-PF^2 algorithm for smooth collision-free motion planning for a multi-link serial robot with structural and hand position uncertainties. The uncertainty is taken into account in the 3D-PF^2 algorithm through the force fields surrounding each link. When the uncertainty increases, the force fields increase.

The 3D-PF^2 algorithm was tested through simulations and experiments. The algorithm allows smooth collision-free motion planning in 3D environments. However it is limited to short time-horizon paths as it requires goal positions to be supplied. This reduces it's capacity to navigate complex environments. Additionally the 3D-PF^2 algorithm lacks the ability to recover from failure conditions. A longer-time horizon planner is needed to develop a path through the environment. This will be explored in the next chapter.

Chapter 5

Line of Sight Tree Algorithm

In Chapter 4 a 3D-PF^2 algorithm was presented for developing short-time horizon paths for a multi-link serial robot with uncertainty in the joint coordinate frame of reference. It was stated that a longer-time horizon planner is required; a Line of Sight Tree (LoST) algorithm was developed for this purpose.

The LoST algorithm provides waypoints as goal locations for the 3D-PF^2 algorithm which plans paths for short time-horizon motion planning and collision avoidance. Waypoints are found in a manner loosely based on the way a person views a scene whereby their gaze tends towards important regions such as the edges of objects. The LoST algorithm uses line of sight (LoS) to incrementally build a Cartesian-space tree of waypoints from a start node through the environment to a goal node.

This chapter describes the LoST algorithm [23] both as a standalone Cartesian space path planner and integrated with the 3D-PF^2 algorithm. With the 3D-PF^2 algorithm, the end-effector's start position and the goal location are cast as the start and goal nodes respectively. The LoST algorithm is capable of functioning with known and unknown environments. While the LoST algorithm was developed for use with the 3D-PF^2 algorithm, it can be used as a longer-time horizon planner for any point-to-point planner.

The outline of the chapter is as follows: the architecture of the LoST algorithm is explained as a standalone path planner; modifications to the architecture are described in conjunction with the 3D-F^2 and 3D-PF^2 algorithms; and the results of simulations and experiments shown in comparison to an RRT algorithm.

5.1 Line of Sight Tree Algorithm Architecture

The LoST algorithm draws inspiration from how humans perceive a scene whereby their gaze tends to fixate on important and information-rich regions, such as contrasting colours or the edges of objects [203]. These regions subsequently direct their gaze to other peripheral regions. The LoST algorithm similarly uses this concept to find the goal node. The algorithm "looks" from the current position of the robot for the goal node. If the goal node is not found, the LoST algorithm finds obstacle edges, within line of sight (LoS) of its current position, to place intermediate nodes beyond. The LoS is the unimpeded view between two points within the environment.

The basic flow of the LoST algorithm is shown in Figure 5.1. The LoST algorithm develops a path in Cartesian space to a goal node based on line of sight from nodes. From the scan position, the line of sight surrounding a node establishes the visible region and is determined by omnidirectionally ray casting around it. Each ray returns the distance to the first obstacle it hits. From the scan both common regions and intermediate nodes are

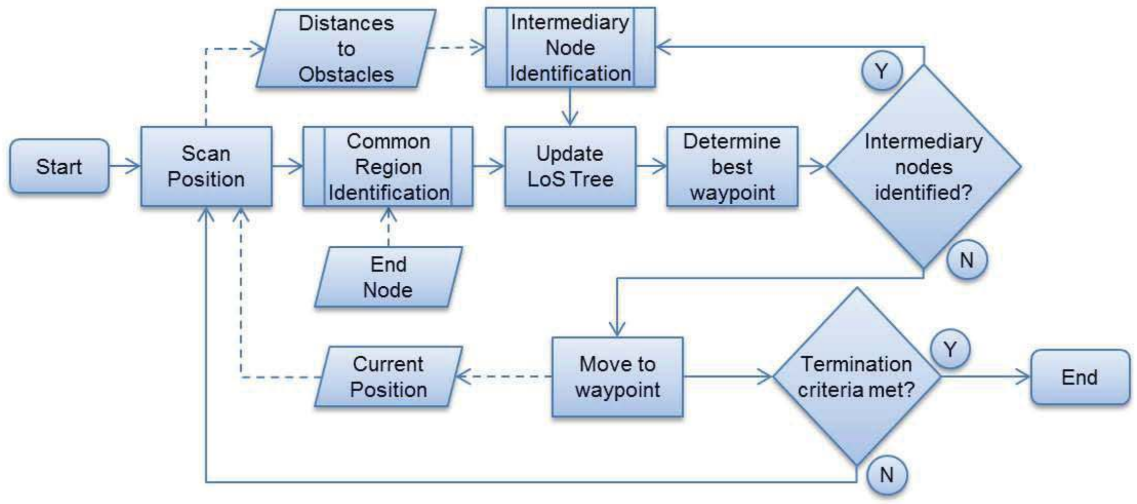


Figure 5.1: Basic LoST Algorithm flowchart.

determined. Common region identification is performed to determined common regions; areas between which a node and the goal node share a common visible region. The centre of this region, the common region centre, is used to connect to the goal node. These regions are populated into the LoS tree. If no common regions are found, intermediary node identification is performed to find intermediate nodes. These nodes are also populated into the LoS tree. The nodes in the LoS tree are weighted and the next best waypoint to search from is determined. At this stage the LoST algorithm may determine that intermediary node identification is required to be performed again. Otherwise the robot moves to the next best waypoint and, if the goal location has not been reached, the process repeats. The visible region of the goal node can only be established in known environments, otherwise the goal node is searched for directly instead of the common region. Intermediate nodes are positioned beyond obstacles based on, and with respect to, the LoS from a node. Intermediate nodes open new regions to be searched, increasing the chances of finding a common region.

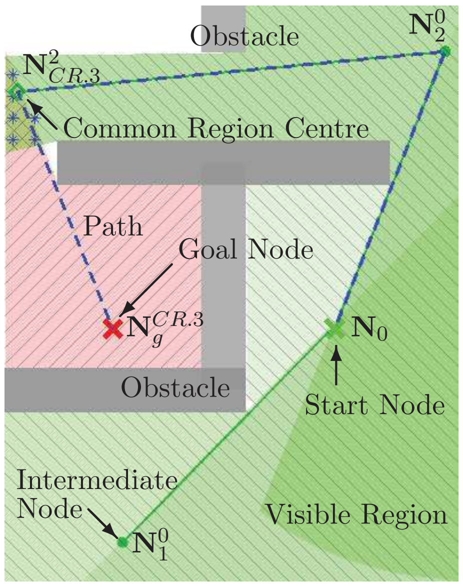
A LoS tree is built to represent the connectivity between the start node, common region centres, intermediate nodes and the goal node. By using LoS an obstacle free path is created between sequential elements of the LoS tree with paths through the environment found by following branches of the LoS tree. The final path is determined by backtracking through the LoS tree from the goal node to the common region centre then to each preceding node back to the start node.

The following notation is used to describe the LoST algorithm. An intermediate node, $\mathbf{N}_u^v$, describes a possible robot position in Cartesian space with the superscript, v, referring to the index of the node it was generated from and the subscript, u, referring to its index within the LoS tree. A common region centre, $\mathbf{N}_{CR.u}^v$, is a point in Cartesian space with the subscript suffix, u, denoting its index and superscript, v, refers to the index of the node it was found from. The current node, $\mathbf{N}_u$, describes a robot position in Cartesian space, $\{x, y\}$ in 2D and $\{x, y, z\}$ in 3D, with the subscript, u, again referring to the node's index within the LoS tree. The superscript notation for the current node is not shown to differentiate itself from its intermediate nodes and common regions within this thesis. The start node, $\mathbf{N}_0$, and goal node, $\mathbf{N}_g$, describes the initial and desired final robot positions.

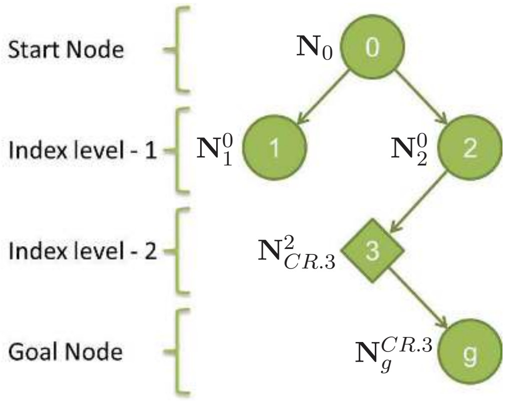
Common regions and intermediate nodes generated from the start node have an index level of 1. The index level of a common region or intermediate node is incremented by 1 based on the index level of the node they were generated from. For example, a node with an index level of 3 will generate common region centres with an index level of 4. To prevent the LoST algorithm from searching indefinitely, a maximum index level is used. At the maximum index level only common regions are generated as searches from intermediate nodes are no longer performed.

The LoST algorithm determines visible regions by ray casting omnidirectionally from a node. The visible region refers to the visibility of the area surrounding a point in the environment. Ray casting provides a quick and computationally efficient method to determine the distance from the node to environmental obstacles [204]. As ray casting does not need previous environmental knowledge, the LoST algorithm can subsequently be suitable for online applications in unknown environments. While ray casting is used in this algorithm, this method is interchangeable with any sensors or methods which determine the visible regions within a set field of view such as ultrasonic sensors or computer vision.

Figure 5.2a shows the LoST algorithm (Algorithm 3) finding a path through an environment with the resulting LoS tree shown in Figure 5.2b. For this example a maximum index level of 2 is used to demonstrate the algorithm's behaviour. From both the start and goal nodes, $\mathbf{N}_0$ and $\mathbf{N}_g$, the visible regions are determined and checked to find common regions (Section 5.1.1). The start node's visible region is represented as the green shaded area, while the goal node's is red. As no common regions are found between $\mathbf{N}_0$ and $\mathbf{N}_g$, intermediate node identification is performed from the current node, $\mathbf{N}_0$, (Section 5.1.2) resulting in two intermediate nodes, $\mathbf{N}_1^0$ and $\mathbf{N}_2^0$, both of which have an index level of 1 (Figure 5.2b). Both intermediate nodes are then weighted (Section 5.1.4) to determine their index within the LoS tree. The LoST algorithm then determines which intermediate node is selected to continue the search process by using a depth first search on the LoS tree. In this instance the depth first search selected node 1, $\mathbf{N}_1^0$. The process then continues with common region identification between $\mathbf{N}_1$ and $\mathbf{N}_g$ which also fails. Intermediate nodes are not determined as subsequent common region centres will have an index level above the maximum index level. As no further searching is possible from node 1, $\mathbf{N}_1$, its weighting is set to zero and the next best node is determined – which is node 2, $\mathbf{N}_2$.



(a) LoST algorithm in a simple 2D environment



(b) Related LoS tree for the 2D environment

FIGURE 5.2: (a) The start and goal nodes are represented as green and red crosses. From the start node, intermediate nodes, shown as green dots, are searched for until a common region is found, shown as blue asterisks. The centre is determined, the green diamond, and is used to connect to the goal node. The final path is shown as a blue dashed line. The green backgrounds with diagonal lines running from the top left to bottom right are the start and intermediate node's visible regions. The goal node's visible region is represented by the red background with diagonal lines running in the opposite direction. (b) The LoS tree for the shown simple environment. Circles represent nodes which are indexed according to their sorted weighted value at each index level. Diamonds represent common regions and are numerated similarly.

As a common region is found between $\mathbf{N}_2$ and $\mathbf{N}_g$ the common region centre, $\mathbf{N}_{CR.3}^2$, is determined and used to connect to the goal node. After backtracking through the LoS tree, the final path is from $\mathbf{N}_0 \rightarrow \mathbf{N}_2^0 \rightarrow \mathbf{N}_{CR.3}^2 \rightarrow \mathbf{N}_g^{CR.3}$.

The search algorithm used to determine the next best waypoint affects the path taken by the robot. A depth first search algorithm ensures a branch of the LoS tree is fully explored; which prevents excessive backtracking. Instead an A* algorithm may be used for the shortest path. In this research a depth first search algorithm is used.

5.1.1 Common Regions

The visible region of a node, $\mathbf{N}_u$, refers to the visibility of the area surrounding the node within the environment. When the visible region of a node overlaps with that of the goal

Algorithm 3: Line of Sight Tree Algorithm

Input: $StartNode \longleftarrow$ Start node, $GoalNode \longleftarrow$ Goal node
Output: $Path \longleftarrow$ Final path

```
$CurrentNode \leftarrow StartNode$                                  ▷ Set the first node to search from
while $Path = \emptyset$ do                                          ▷ Keep searching until a path is found
    $CommonRegions \leftarrow FindCommonRegions(CurrentNode, GoalNode)$    ▷ Determine common regions (Alg. 4)
    if $CommonRegions = \emptyset$ then                              ▷ If no common regions are found
        $IntermediateNodes \leftarrow DetermineNextNodes(CurrentNode)$   ▷ Determine the intermediate nodes (Alg. 5)
        if $IntermediateNodes \neq \emptyset$ then                   ▷ If intermediate nodes are found
            $SetNodeWeights(IntermediateNodes)$                      ▷ Weight the nodes (Section 5.1.4)
            $LoSTree \leftarrow Connectivity(CurrentNode, IntermediateNodes)$   ▷ Update the LoS tree
        end if
        $CurrentNode \leftarrow DetermineNextNode(LoSTree)$          ▷ Use the depth first search to find the next best node
        if $CurrentNode = StartNode$ then                            ▷ If the next index is the start node
            return $Path \leftarrow \emptyset$                       ▷ The algorithm fails
        end if
    else
        $SetNodeWeights(CommonRegions)$                              ▷ Weight the common region centres (Section 5.1.4)
        $Path \leftarrow backtrack(GoalNode, LoSTree, StartNode)$    ▷ Backtrack to find the path
    end if
end while
return $Path$                                                        ▷ Successful
```

node, $\mathbf{N}_g$, they share a common region (Algorithm 4). The centre of the common region is then used to connect the path between the node and the goal node. The visible region is established by ray casting omnidirectionally (circularly in 2D, spherically in 3D), to a maximum ray cast distance, with the area covered by the rays designating a node's visible region (Figure 5.3). While a longer distance will increase a node's visible region, it is prudent to instead set the maximum ray cast distance to the maximum range of a sensor.

To determine common regions, the visible regions from both the current node and the goal node are found and discretised into voxels [205] (shown as asterisks in Figure 5.3). An appropriate voxel resolution should be set according to the implementation. However, for maximum visibility it should be ensured that, at the maximum ray cast distance, the

Algorithm 4: Common Region Identification Algorithm

Input: $CurrentNode \longleftarrow$ Current node, $GoalNode \longleftarrow$ Goal node
Output: $CommonRegionCentres \longleftarrow$ Common region centres

```
$CurrentVoxels \leftarrow Discretise(RayCast(CurrentVisibleRegions))$   ▷ Find the visible voxels from the current node
$GoalVoxels \leftarrow Discretise(RayCast(GoalVisibleRegions))$   ▷ Find the visible voxels from the goal node
$CommonVoxels \leftarrow CompareVoxels(CurrentVoxels, GoalVoxels)$   ▷ Find any common voxels
if $CommonVoxels \neq \emptyset$ then   ▷ If there are common voxels
    for $currentVoxel = 1$ to $CommonVoxels$ do   ▷ For each common voxel
        $CRInRange \leftarrow FindCRInRange(currentVoxel, CRs)$   ▷ Find common regions in range of the current voxel
        if $CRInRange \neq \emptyset$ then   ▷ If there are no common regions within range
            $NewCommonRegion(currentVoxel)$   ▷ Create a new common region
        else
            $Merge(currentVoxel, CRInRange)$   ▷ Otherwise merge the current voxel with in range common regions
        end if
    end for
    $CommonRegionCentres \leftarrow FindCentres(CRs)$   ▷ Find the common region centres for each common region
else
    $CommonRegionCentres \leftarrow \emptyset$   ▷ Otherwise there are no common regions
end if
return $CommonRegionCentres$   ▷ Return the common regions
```

distance between ray end points does not exceed the distance between voxels. Any voxels which are visible from both the current node and goal node are common voxels. If no common voxels exist, no common region centres are returned. Otherwise common voxels are grouped together to form common regions. This is necessary as an obstacle may split a node's visible region causing multiple common regions. Each common voxel is checked to determine if it is within a minimum Euclidean distance of voxels within existing common regions. If it is, all common regions within the distance are merged with the voxel into a single common region. If it is not, a new common region is created from the voxel.

Once all voxels have been evaluated, the common region centres, $\mathbf{N}^{v}_{CR.u}$, of each common region are calculated by finding its geometric centre. All common region centres are then weighted, as described in Section 5.1.4, to determine its index within the LoS tree. This is

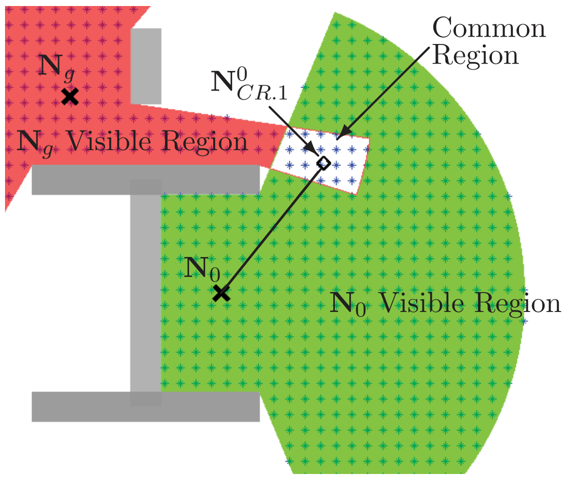


FIGURE 5.3: The start node, $\mathbf{N}_0$, and the goal node, $\mathbf{N}_g$, visible regions are shown as green and red regions respectively. The asterisks are the voxels seen from each node. The overlapping area is the common region, shown in white, with the common region centre, $\mathbf{N}^0_{CR.1}$, shown as a diamond.

to differentiate between common region centres when multiple common regions have been found.

In unknown environments the visible region surrounding the goal node cannot be determined. Therefore the described implementation cannot be directly used. Instead the goal node should be searched for directly. To account for this situation in the existing framework the voxels closest to the goal node need be determined. The common region identification algorithm would then search for these voxels instead of all possible voxels within the goal node's visible region. This assumes that the area directly surrounding the goal node is free of obstacles and a path can be established from the closest voxels to the goal node.

5.1.2 Intermediate Node Identification Algorithm

Intermediate nodes are points from where the LoST algorithm searches for common regions. Intermediate nodes are positioned beyond obstacles with respect to, but within LoS of, the current node, $\mathbf{N}_u$. This increases the chances of exploring new regions and successfully finding a common region. Also by maintaining LoS between the current node and their intermediate nodes an obstacle-free path through the environment is ensured.

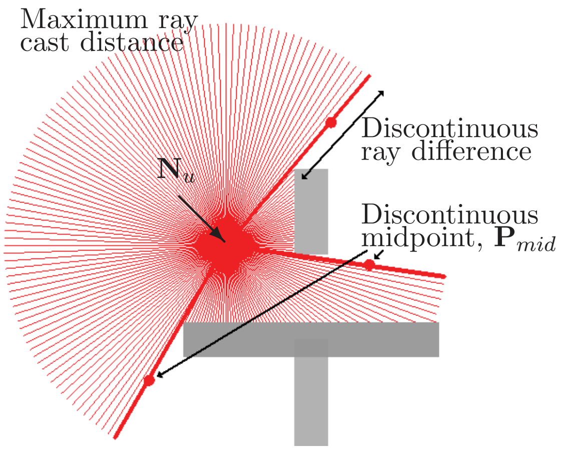
Algorithm 5 is used to determine intermediate nodes for the current node. First, ray casting is performed omnidirectionally (circularly in 2D or spherically in 3D) from the current node to determine the distances to surrounding obstacles. Rays only extend up to a maximum ray cast distance. Figure 5.4 shows 2D ray casting performed at 1° intervals around a node. The difference between the distance travelled by two adjacent rays is the discontinuous ray difference. If the discontinuous ray difference is longer than a threshold, the two rays are considered to be discontinuous; implying that an obstruction has occurred between them. A discontinuous midpoint, $\mathbf{P}_{mid}$, is then positioned along the longer ray at a distance from the current node equal to the average length of the two rays, l_{min} and l_{max} (Equation 5.1). Here $\mathbf{P}_{max}$ is the end point of the longer ray. Placing the discontinuous midpoint along the longer ray ensures it extends beyond the obstruction.

$$\mathbf{P}_{mid} = \mathbf{N}_u + \frac{l_{min} + l_{max}}{2}(\mathbf{P}_{max} - \mathbf{N}_u) \tag{5.1}$$

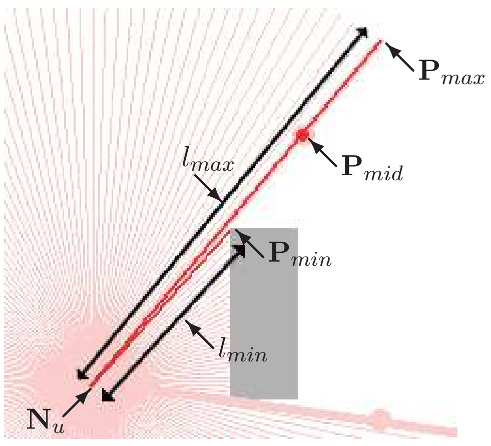
Algorithm 5: Intermediate Node Identification Algorithm

```
Input: $CurrentNode \longleftarrow$ Current node
Output: $IntermediateNodes \longleftarrow$ Intermediate nodes

$RayLengths \leftarrow RayCast(CurrentNode)$   ▷ Determine all ray lengths
$DiscontinuousPairs \leftarrow FindDiscontinuousPairs(RayLengths)$   ▷ Determine the discontinuous pairs
if $DiscontinuousPairs \neq \emptyset$ then   ▷ If there are discontinuous pairs
    $DiscontinousMidpoints \leftarrow FindDiscontinuousMidPoints(DiscontinuousPairs)$   ▷ Find the discontinuous midpoints
    if $3DEnvironment$ then   ▷ If it is a 3D environment
        $IntermediateNodes \leftarrow RANSAC(DiscontinousMidpoints)$   ▷ Use RANSAC to find the intermediate nodes
    else
        $IntermediateNodes \leftarrow DiscontinousMidpoints$   ▷ Use the midpoints as intermediate nodes
    end if
else   ▷ Otherwise if there are no discontinuous pairs
    $RayEnds \leftarrow FindRayEndPoints(RayLengths)$   ▷ Find the end point of each ray
    $DetermineEndWeights(RayEnds)$   ▷ Weight the ray ends (Section 5.1.4)
    $IntermediateNodes \leftarrow Highest(RayEnds)$   ▷ Use the highest weighted ray end as an intermediate node
end if
```


Maximum ray cast distance
Discontinuous ray difference
$\mathbf{N}_u$
Discontinuous midpoint, $\mathbf{P}_{mid}$

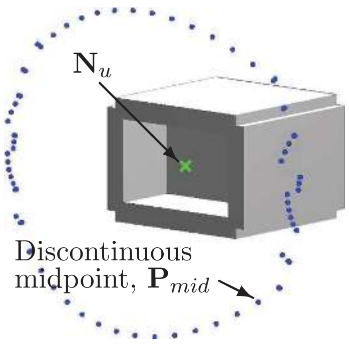
(a)



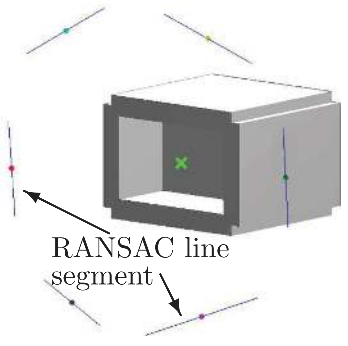
(b)

Figure 5.4: (a) Ray casting performed at 1° intervals around a central node in 2D with the longer of discontinuous rays shown as thickened red lines. The centre of the difference between discontinuous rays is taken from the longer to find the discontinuous midpoints. These are shown as red dots along the length of the longer of the discontinuous rays. (b) shows a discontinuous pair with the position of the current node, $\mathbf{N}_u$, the discontinuous midpoint, $\mathbf{P}_{mid}$, the longer ray end point, $\mathbf{P}_{max}$, and length, l_{min} and the shorter ray end point, $\mathbf{P}_{min}$, and length, l_{min}.

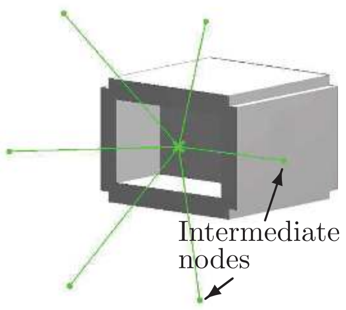
In 2D environments the discontinuous midpoints are directly used as the intermediate nodes. However, in 3D environments there will be a large number of discontinuous midpoints along obstacle edges (Figure 5.5a). If these discontinuous midpoints are used directly as intermediate nodes, common region checks will result in similar solutions and unnecessary computation. Instead resultant midpoints should be found which would encompass the visible regions from the majority of the discontinuous midpoints. As no



(a) Multiple discontinuous midpoints



(b) Line segment midpoints

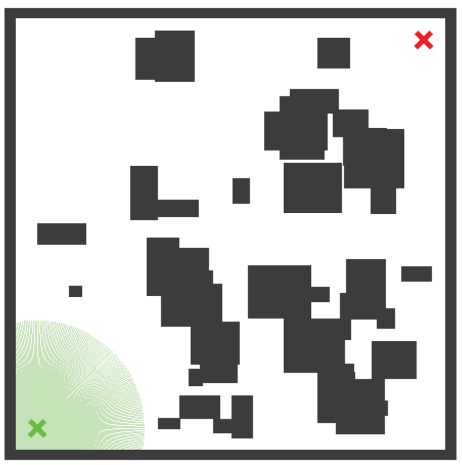


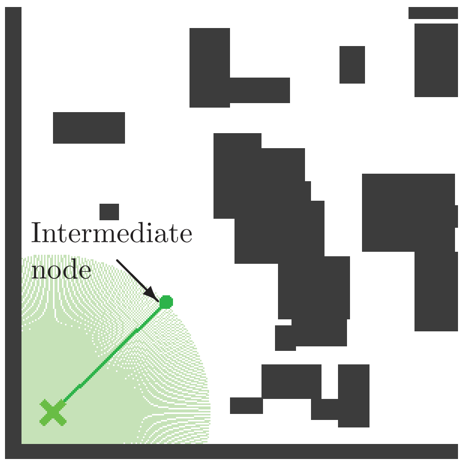
(c) Resultant intermediate nodes

Figure 5.5: (a) In 3D environments, LoST algorithm will find a high number of discontinuous midpoints. (b) RANASC is used to find line segment and, subsequently, their midpoints. (c) These midpoints are used as intermediate nodes.

considerations are made of the environment beyond the current node's visible region, it is assumed that any cluster of discontinuous midpoints sharing a common obstacle edge will have similar visible regions; with the visible region of a discontinuous midpoint at the centre of that cluster having the highest shared visibility with other discontinuous midpoint visible regions within that cluster. Resultant nodes should thus be positioned in the centre of clusters of discontinuous midpoints which share a common obstacle edge. To determine which midpoints share a common edge a Random Sample Consensus (RANSAC) algorithm is applied to all discontinuous midpoints to determine line segments (Figure 5.5b). The geometric centre of these line segments are determined and used as intermediate nodes for the 3D environment (Figure 5.5c).

It is possible that in some situations the maximum ray cast distance is insufficient to detect any obstacles resulting in no new intermediate nodes being found. This can prematurely end either a single branch of the LoS tree or the LoST algorithm altogether (Figure 5.6a). This can be addressed by determining the end point for each ray. These points are then weighted (Section 5.1.4) with the highest weighted point used as an intermediate node (Figure 5.6b).


(a) No obstacles within range



(b) Resultant intermediate node

FIGURE 5.6: (a) The maximum ray cast distance may be too short, preventing rays from reaching obstacles. (b) In this case the highest weighted ray end point should be used as an intermediate node instead.

5.1.3 Safe Robot Workspace

To prevent collisions, the size of the obstacles may be increased to account for the robot workspace. With the LoST algorithm, by instead using the ray cast data, an adjusted longer discontinuous ray can be found which will help to avoid collisions.

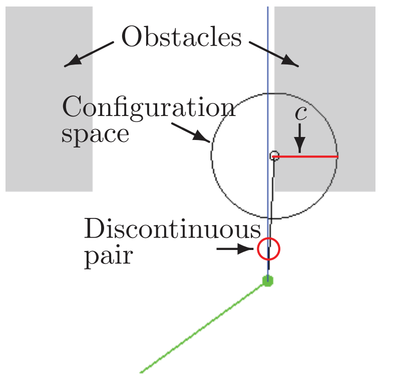
Determining the adjusted longer discontinuous ray is done in two stages. The first stage adjusts the ray to avoid collisions with the obstacle which caused the discontinuous pair (Figure 5.7a). Here c is the radius of the workspace surrounding the robot. Using the equation for the angle of a chord (Equation 5.2), the minimum angle required to bypass the obstacle, θ_{min}, is obtained. Here l_{min} is the length of the shorter discontinuous pair. The number of rays in the minimum angle, Z_{off}, is then determined by dividing the minimum angle, θ_{min}, and the angle between rays, θ_{res}, and rounding up the result (Equation 5.3). An offset ray is found by counting from discontinuous ray the number of rays in the minimum angle (Figure 5.7b).

$$\theta_{min} = 2\arcsin(\frac{c}{2l_{min}}) \tag{5.2}$$

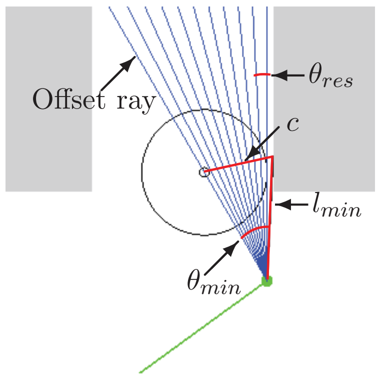
$$Z_{off} = \left\lceil \frac{\theta_{min}}{\theta_{res}} \right\rceil \tag{5.3}$$

The second stage further offsets the ray to account for collisions prior to the discontinuous pair (Figure 5.7c). A collision occurs if a ray ends within the workspace along the offset ray. Only rays prior to the discontinuous pair and up to 90° from the offset ray need be considered; any rays beyond this are not in the direction of motion and will not collide. By using trigonometry, the closest ray is determined and compared to the radius of the workspace, c (Equation 5.4). Here $\mathbf{L}$ are the ray distances and $\boldsymbol{\theta}$ are their angles from the offset ray. The offset ray is then further adjusted based on the angle, θ_{off}, found in Equation 5.4 (Figure 5.7d).

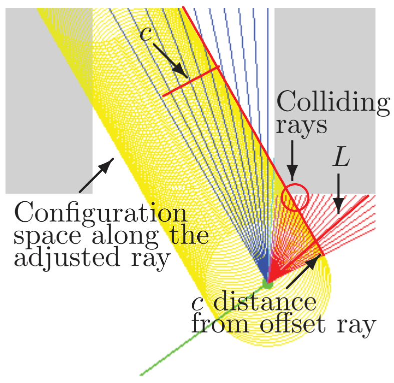
$$\theta_{off} = \max_{\mathbf{L}\sin\boldsymbol{\theta}\leq c} \boldsymbol{\theta} \tag{5.4}$$



(a) Workspace around the shorter discontinuous pair

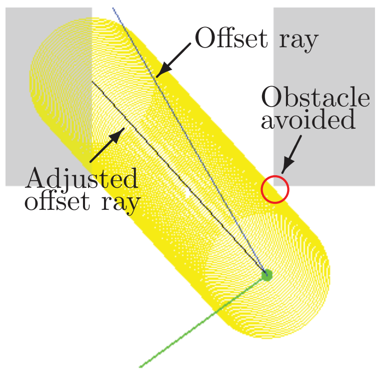


(b) Offset ray used to avoid obstacle causing discontinuous pair

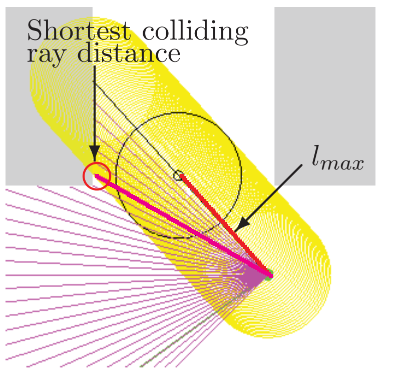


c
Colliding rays
L
Configuration space along the adjusted ray
c distance from offset ray

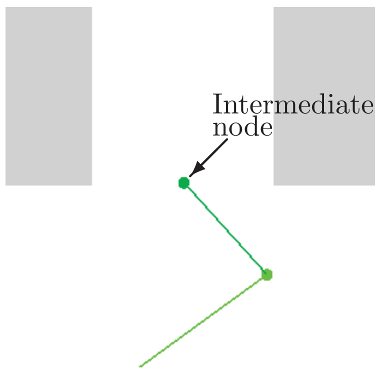
(c) Workspace of the robot along the offset ray



(d) Workspace along the adjusted offset ray



(e) Rays checked for lowest collision within the adjusted offset workspace



(f) Resultant intermediate point

FIGURE 5.7: (a) Workspace around the shorter discontinuous pair should be considered. (b) From the shorter discontinuous pair, the offset representing the number of rays required to avoid the obstacle is determined. (c) The rays colliding with the workspace along the offset ray are determined. (d) The adjusted offset ray avoids the colliding obstacles. (e) The shortest colliding ray within the workspace along the adjusted offset ray is determined. (f) The final discontinuous midpoint based on the shortest colliding ray, minus the workspace, and the original shortest discontinuous ray. Here the LoS tree and intermediate nodes are shown as green lines and dots.

Once the adjusted offset ray has been finalised, the maximum distance along it without colliding with obstacles must be determined (Figure 5.7e). Again using trigonometry, the shortest ray which collides with the workspace along the adjusted offset ray is determined (Equation 5.5). In 2D the rays up to 90° from the adjusted offset ray are checked. In 3D this also includes all rays up to 90° from the adjusted offset ray in the directions perpendicular to the discontinuous pair. Here δ is the ray's offset angle in the perpendicular direction. The distance of the shortest ray, minus the radius of the workspace, dictates the maximum distance along the adjusted offset ray a robot can travel without colliding, l_{max}. This distance is compared to the shortest discontinuous ray length, again with the radius of the workspace subtracted (Equation 5.6). If it is longer, an intermediate node is found (Equation 5.1) with the end of the longer discontinuous ray, $\mathbf{P}_{max}$, substituted for the end of the adjusted offset ray and the distance of the longer ray, l_{max}, substituted for the distance found in Equation 5.5 (Figure 5.7f); otherwise no intermediate node is found.

$$l_{max} = \begin{cases} \min\limits_{\mathbf{L}\sin\boldsymbol{\theta}\leq c} \mathbf{L} - c & \text{in 2D} \\ \\ \min\limits_{\mathbf{L}\sin\boldsymbol{\theta}\cos\boldsymbol{\delta}\leq c} \mathbf{L} - c & \text{in 3D} \end{cases} \tag{5.5}$$

$$l_{max} > l_{min} - c \tag{5.6}$$

5.1.4 Weighting Criteria

Weighting criteria are applied to all intermediate nodes, $\mathbf{N}_u^v$, of the current node, $\mathbf{N}_u$, to improve the chances of selecting intermediate nodes which will converge upon a solution sooner. It also determines which common region centres, $\mathbf{N}_{CR.u}^v$, will be used when multiple common region centres are found. In both cases, higher weightings are preferred. While weighting criteria are necessary, the specific criterion used is dependent on the application and is at the discretion of the user. Weighting factors, α_m, can be used to bias certain criteria over others, prioritising nodes based on any environmental factors or to cater the LoST's algorithm to a specific application. The following criteria are all based on the distance from common region centres or intermediate nodes to other nodes within the LoS tree.

The first criterion is based on the Euclidean distance to the goal node, $\mathbf{N}_g$ (Equation 5.7). The closer intermediate nodes are to the goal node, the more likely subsequent common region checks will be successful as it is less likely there will be an obstruction within a shorter distance. While the closer common region centres are to the goal node, the shorter the final path. This is the only criterion for common region centres. The next criterion is based on the Euclidean distance from the start node, $\mathbf{N}_0$ (Equation 5.8). Longer distances improve the chances of intermediate nodes expanding to new regions by searching further away from the start node. The final criterion is based on the Euclidean distance from the current node, $\mathbf{N}_u$ (Equation 5.9). The further away an intermediate node is from the current node, the less likely that any new visible region check from the intermediate node will overlap with any regions previously seen, increasing the likelihood of common regions, and subsequently the goal node, being found.

$$d_{1.u} = \begin{cases} \|\mathbf{N}^v_{CR.u} - \mathbf{N}_g\| & \text{with common region centres} \\ \|\mathbf{N}^v_u - \mathbf{N}_g\| & \text{with intermediate nodes} \end{cases} \tag{5.7}$$

$$d_{2.u} = \begin{cases} 0 & \text{with common region centres} \\ \|\mathbf{N}^v_u - \mathbf{N}_0\| & \text{with intermediate nodes} \end{cases} \tag{5.8}$$

$$d_{3.u} = \begin{cases} 0 & \text{with common region centres} \\ \|\mathbf{N}^v_u - \mathbf{N}_u\| & \text{with intermediate nodes} \end{cases} \tag{5.9}$$

Once each set of distances, $d_{m.u}$, have been found, the values are normalised (Equation 5.10). Where $\min(d_m)$ and $\max(d_m)$ are the minimum and maximum values respectively of each set of distances, $d_{m.u}$, with m used to identify the weighting criterion.

$$K_{m.u} = \frac{d_{m.u} - \min(d_m)}{\max(d_m) - \min(d_m)} \tag{5.10}$$

The final weighting is the sum of individual normalised weighting coefficients for each common region (Equation 5.11), and intermediate node (Equation 5.12) multiplied by weighting factors, α. With regards to the goal node criterion (Equations 5.7), shorter distances are more favourable. The sum of the weighting factors is 1 (Equation 5.13) to ensure that the final weightings are between 0 and 1. The values of the α coefficients are determined based on the application. In this thesis, as it was decided that each criterion be given equal importance, all values of α are equal.

$$\mathbf{N}^v_{CR.u}.w = \alpha_1(1 - K_{1.u}) + \alpha_2 K_{2.u} + \alpha_3 K_{3.u} \tag{5.11}$$

$$\mathbf{N}^v_{u}.w = \alpha_1(1 - K_{1.u}) + \alpha_2 K_{2.u} + \alpha_3 K_{3.u} \tag{5.12}$$

$$\sum_{m=1}^{3} \alpha_m = 1 \tag{5.13}$$

5.1.5 Line of Sight Tree Algorithm Termination Criteria

There are two criteria for terminating the LoST algorithm: 1) a common region centre is found or 2) the depth first search identifies the start node as the next node to search from. If a common region centre is found, the LoST algorithm is successful. The final path is determined by backtracking from the goal node to the highest weighted common region centre then, via each subsequent intermediate node, back to the start node.

The second criterion occurs when all nodes of the LoS tree have a zero weighting resulting in the start node being selected as the next node to search from. When this occurs the LoST algorithm has failed. An intermediate node receives a zero weighting when further searches will cause the current index level to exceed the maximum index level. The maximum index level is used to prevent a branch from being searched indefinitely. The current node's weighting will also be set to zero if all its intermediate nodes have zero weightings. When the LoST algorithm fails either there is no solution or a higher maximum index level must be set.

5.2 Line of Sight Tree Algorithm with the 3D-F^2 Algorithm

An approach is established which integrates the LoST algorithm with the 3D-F^2 algorithm. The LoST algorithm can be used as a longer-time horizon path planner for the 3D-F^2 algorithm which develops short time-horizon paths.

The LoST algorithm develops obstacle free paths, i.e. waypoints, which cannot be directly used for smooth motion of a multi-link serial robot. The 3D-F^2 algorithm can generate smooth, collision-free motion, but it may become trapped in local minima stalling further motion [147]. As waypoints found by the LoST algorithm are found through LoS, they are unimpeded by obstacles. Similarly, the attractive force generated by the 3D-F^2 algorithm will be unimpeded as no obstacles will be present between two adjacent intermediate nodes. This reduces the chances of generating repulsive forces directly opposing the motion and entering into a local minimum.

5.2.1 Implementation of the Combined LoST and 3D-F^2 Approach

The basic flow of the combined LoST and 3D-F^2 approach is shown in Figure 5.8 and Algorithm 6. From an initial robot pose, the end-effector position, ${}^0\mathbf{P}_e$, is used as the start node, $\mathbf{N}_0$. The LoST algorithm is used to incrementally build a LoS tree from which waypoints are supplied to the 3D-F^2 algorithm. These waypoints can either be common region centres or intermediate nodes. The 3D-F^2 algorithm develops a smooth, collision-free path towards the selected waypoint and, if successfully reached, the LoST algorithm is again initiated. If a waypoint cannot be reached, its weighting in the LoS tree is set to zero and the robot backtracks to the waypoint the motion was initiated from and a new waypoint found. This process continues until the end-effector position, ${}^0\mathbf{P}_e$, reaches the goal node, $\mathbf{N}_g$ or the start node is selected as the next waypoint.

5.2.2 Updating the Line of Sight Tree Algorithm in Real-Time

The LoST algorithm ray casts omnidirectionally from the current node to determine both intermediate nodes and common regions. In real environments, depth sensors may be used

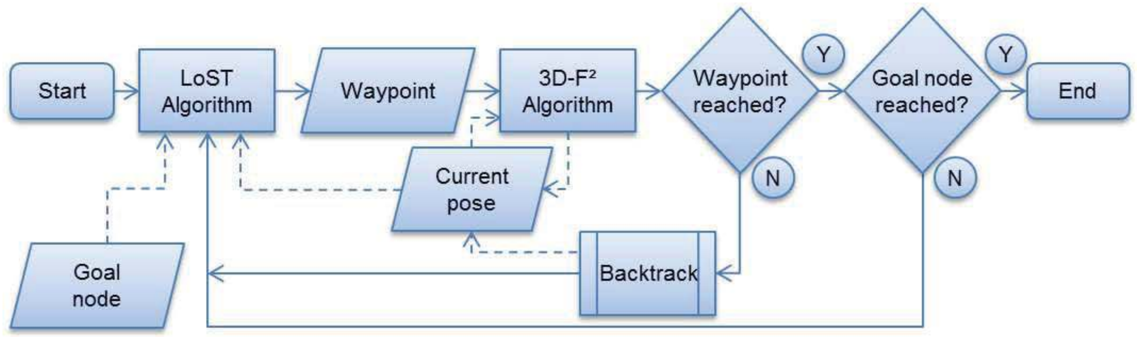


FIGURE 5.8: Basic flow of the combined LoST and 3D-F^2 approach

to determine the required information, although omnidirectional ray casting may not be practically applicable due to the limited field of view of the sensor. Multiple scans may be required to fully capture the necessary distance information surrounding a node.

As the 3D-F^2 algorithm moves the multi-link serial robot's end-effector to a waypoint generated by the LoST algorithm, further ray cast operations should occur from this waypoint. Optimally, the sensor origin and robot end-effector should be aligned and, while scanning the surrounding environment, the sensor origin position should remain stationary with only its orientation altered. However, sensors are generally offset from the end-effector and often reorientating will also modify the position. These issues prevent the sensor from ray casting directly from the waypoint. It is possible to find inverse kinematic solutions for each required orientation; however, this can be computationally expensive.

Algorithm 6: The Basic Approach Algorithm

```
Input: $Goal \longleftarrow$ Goal node, $StartPose \longleftarrow$ Start pose
Output: $Path \longleftarrow$ Final path

  $Pose \leftarrow StartPose$
  while $EndEffector \neq Goal$ do                      ▷ End-effector is not at the goal node
      while $EndEffector \neq Waypoint$ do                ▷ End-effector is not at a waypoint
          $Waypoint \leftarrow LoST(Goal)$           ▷ Use the LoST algorithm to find a waypoint
          $Pose \leftarrow 3D - F^2(Waypoint)$             ▷ Find a path towards the waypoint
          if $EndEffector \neq Waypoint$ then           ▷ If the waypoint was not reached
              $Pose \leftarrow Backtrack$           ▷ Backtrack to the preceding reached pose
          end if
      end while
  end while
  return $Path \leftarrow BacktrackCurrentPose, StartPose$                        ▷ Successful
```

Instead a set of poses may be generated which will allow the sensor to completely scan the surrounding environment while minimising the variation in the sensor position. While there may be some overlap with scans, this provides a quick, computationally efficient solution in practical applications.

As the robot moves through each pose, the sensor scans the environment. The readings are applied to the common region identification algorithm (Algorithm 4) to determine a node's visible region and discretised voxels, and the intermediate node identification algorithm (Algorithm 5) to determine intermediate nodes.

As multiple scans may overlap, the same discretised voxel may be seen multiple times while determining common regions. To account for this, as each scan is taken only new voxels are added to those already found. Similarly, as obstacles may be scanned multiple times due to overlapping scans, multiple intermediate nodes may be found which are within close proximity to each other. This can be minimised by averaging all intermediate nodes within a distance threshold into a new resultant intermediate node.

5.2.3 Additional Weighting Criteria for a Multi-Link Serial Robot

As the LoST algorithm does not take into account a multi-link serial robot's kinematics, the robot may not be able to reach the selected waypoint - which can be either a common region centre or an intermediate node. To increase the chances of finding a successful trajectory, the kinematics need to be incorporated into the weighting criteria for selection of waypoints that are more readily accessible by the multi-link serial robot, i.e. within its workspace. Two additional weighting criteria are proposed in conjunction with those described in Section 5.1.4 are applied.

The first additional criterion is based on the Euclidean distance to the robot's base position, $\mathbf{P}_0$ (Equation 5.14). Generally, for a given multi-link serial robot's workspace, the closer intermediate nodes or common region centres are to the robot's base the harder they are to reach due to joint hard limits and possible self-collisions. However, intermediate nodes or common region centres that are too far from the base also may not be reachable. The second additional criterion is then based on the Euclidean distance to the boundary of the robot's workspace. The number of valid poses to reach an intermediate node or common region centre decreases the closer either are to the boundary of the workspace. As such selecting an intermediate node or common region centre further away from the boundary would be more likely to find a valid pose. The workspace boundary can be difficult to calculate due to possible joint orientations and limits. This calculation is simplified by first determining the distance from an intermediate node or common region to the base position, $\mathbf{P}_0$. This is then subtracted from the length of the robot, d_w (Equation 5.15).

$$d_{4.u} = \begin{cases} \|\mathbf{N}^v_{CR.u} - \mathbf{P}_0\| & \text{with common region centres} \\ \|\mathbf{N}^v_u - \mathbf{P}_0\| & \text{with intermediate nodes} \end{cases} \tag{5.14}$$

$$d_{5.u} = \begin{cases} d_w - \|\mathbf{N}^v_{CR.u} - \mathbf{P}_0\| & \text{with common region centres} \\ d_w - \|\mathbf{N}^v_u - \mathbf{P}_0\| & \text{with intermediate nodes} \end{cases} \tag{5.15}$$

Each set of distances, $d_{m.u}$, are then normalised (Equation 5.10). The final weighting is the sum of the normalised weighting coefficients for each common region (Equation 5.16), and intermediate node (Equation 5.17), multiplied by weighting factors, α. The weightings factors should be modified so their sum equals 1 (Equation 5.18) ensuring the final weightings are between 0 and 1. The value of the α coefficients are determined based on the application. In this thesis, as it was decided that each criterion be given equal importance, all values of α are equal.

$$\mathbf{N}^v_{CR.u}.w = \alpha_1(1 - K_{1.u}) + \alpha_2 K_{2.u} + \alpha_3 K_{3.u} + \alpha_4 K_{4.u} + \alpha_5 K_{5.u} \tag{5.16}$$

$$\mathbf{N}^v_u.w = \alpha_1(1 - K_{1.u}) + \alpha_2 K_{2.u} + \alpha_3 K_{3.u} + \alpha_4 K_{4.u} + \alpha_5 K_{5.u} \tag{5.17}$$

$$\sum_{m=1}^{5} \alpha_m = 1 \tag{5.18}$$

5.3 Line of Sight Tree Algorithm with the 3D-PF^2 Algorithm

The LoST algorithm is integrated with the 3D-PF^2 algorithm using the kinematic model with uncertainty (Figure 5.9). The basic flow remains relatively unchanged to the flow described in Section 5.2 except for one minor addendum; the use of the kinematic model with uncertainty. The LoST algorithm uses the end-effector position, while the 3D-PF^2 algorithm both uses and updates the pose (Chapter 4), as described by this model. The uncertainty in the model is updated as described in Chapter 3.

5.3.1 Multi-Link Serial Robot Weighting Criteria with Uncertainty

Weighting criteria is used to determine the next best waypoint. An additional criterion is specified for use with the kinematic model of the multi-link serial robot with uncertainty. This criterion is in addition to the criteria specified in Section 5.2.3.

The additional criterion involves weighting based on the predicted variance at the waypoint. As discussed in Section 3.2.1, the structural uncertainty is more pronounced based on the distance of a joint from the base location along the global xy-plane. Thus selecting waypoints closer to the base location will result in lower variance and smaller force fields. The criterion is then based on the Euclidean distance, along the global xy-plane, from the

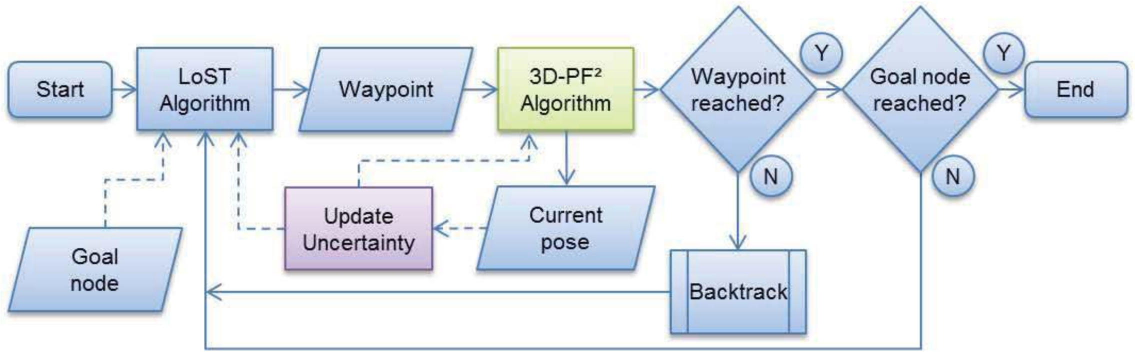


FIGURE 5.9: Basic flow of the combined LoST and 3D-PF^2 approach. The green object is modified based on the algorithm described in Section 5.2.1, the purple object has been added and blue objects are unchanged.

waypoint to the base location (Equation 5.19). This criterion is the same for both common region centres and intermediate nodes.

$$d_{6.u} = \begin{cases} \|\mathbf{N}^v_{CR.u} - \mathbf{P}_0\|_{\psi_{G_{xy}}} & \text{with common region centres} \\ \|\mathbf{N}^v_u - \mathbf{P}_0\|_{\psi_{G_{xy}}} & \text{with intermediate nodes} \end{cases} \tag{5.19}$$

This distance is normalised in the same manner as the other criteria distances (Equation 5.10) and used to find the final weighting for each common region (Equation 5.20) and intermediate node (Equation 5.21). The sum of the weighting factors is modified to include the new criterion (Equation 5.22). The value of the α coefficients are determined based on the application. In this thesis, as it was decided that each criterion be given equal importance, all values of α are equal.

$$\mathbf{N}^v_{CR.u}.w = \alpha_1(1 - K_{1.u}) + \alpha_2 K_{2.u} + \alpha_3 K_{3.u} + \alpha_4 K_{4.u} + \alpha_5 K_{5.u} + \alpha_6(1 - K_{6.u}) \tag{5.20}$$

$$\mathbf{N}^v_u.w = \alpha_1(1 - K_{1.u}) + \alpha_2 K_{2.u} + \alpha_3 K_{3.u} + \alpha_4 K_{4.u} + \alpha_5 K_{5.u} + \alpha_6(1 - K_{6.u}) \tag{5.21}$$

$$\sum_{m=1}^{6} \alpha_m = 1 \tag{5.22}$$

5.3.2 Updating the Line of Sight Algorithm with Uncertainty in the Kinematic Model

A secondary effect of the kinematic model with uncertainty is using raycasting to update the LoST algorithm as the position of the camera is affected. For omnidirectional raycasting the inchworm robot uses a set of pregenerated poses to move the camera (Section 5.2.2). These pregenerated poses are no longer applicable as collision-free poses are not guaranteed.

Instead the 3D-PF^2 algorithm can be used to find the necessary poses to scan from. As the pregenerated poses are used to position and orientate the end-effector mounted camera to the necessary location, the 3D-PF^2 algorithm can be used to manoeuvre the robot to the equivalent end-effector location. This allows collision avoidance to be considered.

A concern may arise from the variance in the end-effector caused by uncertainty affecting the camera position. This has minimal impact on the algorithm. As raycasting is based on the individual scan data, intermediate points are determined directly from the ray distances; the relative position and orientation of the camera in the global coordinate frame of reference does not matter.

5.4 Results

Simulations were performed using MATLAB® and were designed to verify the LoST algorithm as a standalone algorithm and with the 3D-F^2 and 3D-PF^2 algorithms. An RRT algorithm [157] was used as a comparison with the LoST algorithm. The 7DOF inchworm robot [1] was used as a multi-link serial robot.

The RRT algorithm utilises multiple trees to improve convergence towards a solution. This method has increased success in finding paths through narrow passageways, and with fewer collision checks and in fewer search iterations than standard RRT algorithms [157]. The RRT algorithm has been allowed to propagate from trees at both the start and goal nodes to increase the chances of converging upon a successful solution. To prevent the RRT algorithm from searching indefinitely, the maximum number of search iterations was set to 2000.

When the LoST algorithm is specified with a subscript, this subscript refers to the maximum node level, for example, LoST$_3$ refers to a LoST algorithm run with a maximum index level of 3. In all experiments, the maximum ray cast distance was set to ensure that the goal node could not be reached from the start node. The final path length is the distance a simulated point robot travels along the found path, while the total distance travelled is the distance a simulated robot moves between each node while searching for the goal node using the LoST algorithm. The final path length gives an indication of the distance required to reach the goal node if the environment was known and searched prior to moving, while the total distance travelled indicates the distance travelled if the environment is unknown. All values are averaged over the number of times the algorithms are run. When an algorithm failed to find a solution, its data set is not included in the average.

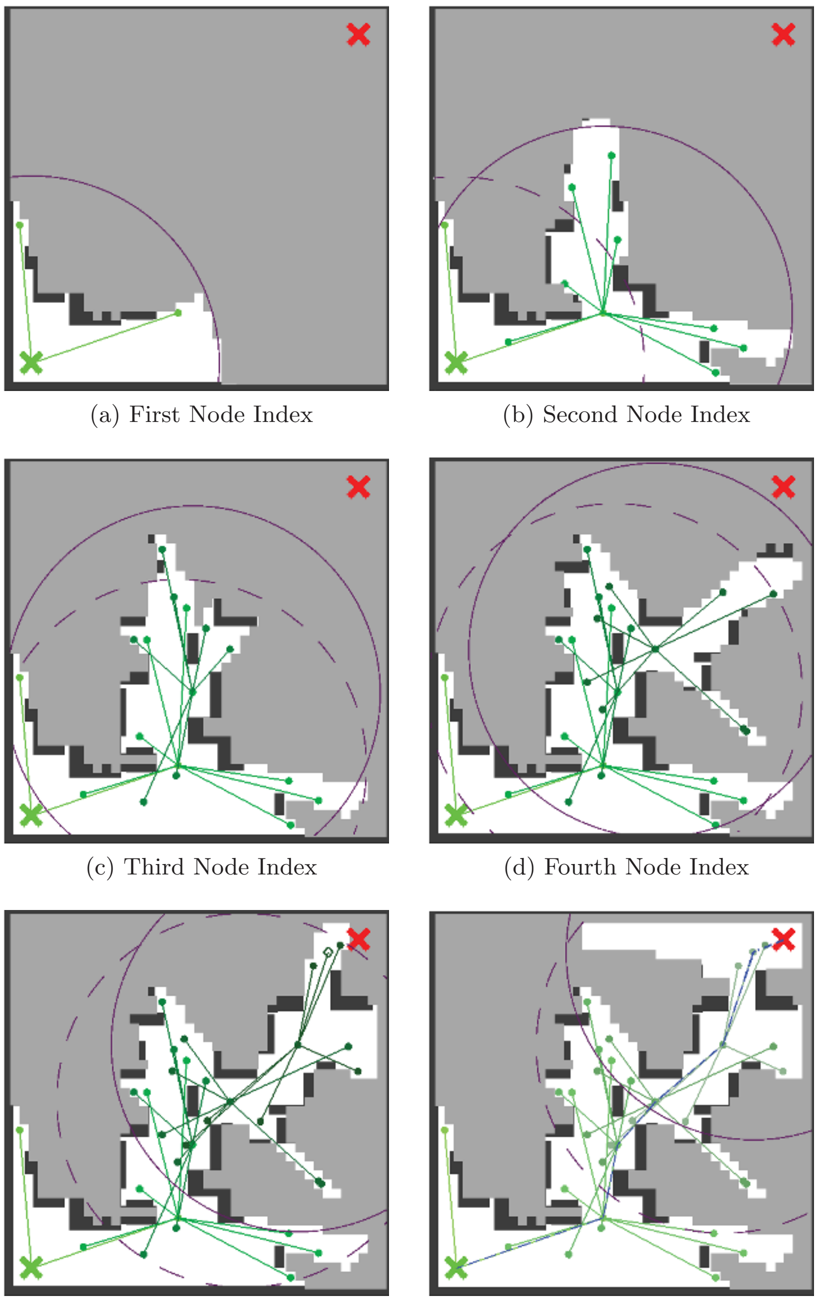
5.4.1 Simulation in Unknown 2D Environments

The LoST algorithm was first tested in 100 unknown environments with 40 randomly sized rectangular obstacles positioned randomly throughout (Figure 5.10). The start and goal nodes are known. The unknown space is represented in the figure by a grey overlayed grid. As the LoST algorithm searches, the visible regions from each node reveals the environment beneath the overlayed grid (Figures 5.10a to 5.10e). The LoST algorithm's maximum index level is set to 5. The RRT algorithm is used for comparison and was run 3 times on each generated environment. This is to prevent biased results due to its probabilistic nature. The RRT algorithm is also assumed to know the environment.

Table 5.1 shows the results for both the LoST and RRT algorithms. Both algorithms had a 100% completion rate. Results show that the LoST_5 algorithm found a path with an average of 50.5% less nodes in the final path, despite functioning in an unknown environment. This is intrinsic to the LoST algorithm as ray casting does not rely on any previous knowledge of the environment. The difference between the known and unknown environments is the termination criteria. In known environments, the LoST algorithm searches for common regions. Instead in unknown environments, the current node searches for a voxel directly surrounding the goal node, which is subsequently used to connect the goal node to the LoS tree. The final path is still revealed by backtracking through the LoS

TABLE 5.1: Unknown environment with 40 random obstacles - averaged over 100 runs

Type	LoST_5	RRT
Completion rate	100%	100%
Average number of nodes in final path	3.98	7.87
Average total number of nodes found	304.96	574.43


(a) First Node Index (b) Second Node Index

(c) Third Node Index (d) Fourth Node Index

(e) Fifth Node Index (f) Final Path

FIGURE 5.10: An example randomly generated unknown 2D environment with 40 random obstacles. Unknown space, which is represented as grey regions, is uncovered based on the visible region of a node. The green and red crosses are the start and goal node respectively with the LoS tree and intermediate nodes shown as green lines and dots. The final path is shown as a blue dashed line. At each index the maximum ray cast distance is shown as a solid magenta circle, with a broken circle denoting the previous maximum ray cast distances.

tree. In unknown environments this effectively increases the maximum number of nodes required to successfully find a path by one in comparison to known environment.

5.4.2 Simulation with Known 2D and 3D Environments

The LoST algorithm is tested in known environments. The environments are generated by creating a number of randomly sized rectangular obstacles in 2D, or rectangular prisms in 3D, positioned randomly within the area. Example environments are shown in Figure 5.11a and 5.12a. Obstacles were not generated near the robot's predetermined start and goal locations.

For both 2D and 3D environments, the LoST and RRT algorithms were run on six sets of 100 randomly generated environments. Beginning with 20 obstacles, each subsequent set incrementally generated an additional 20 obstacles with the last set containing 120 obstacles. For each set of environments, the LoST algorithm was run from a maximum index level of 1 (LoST$_1$) to a maximum index levels 5 (LoST$_5$). Additionally, in 3D

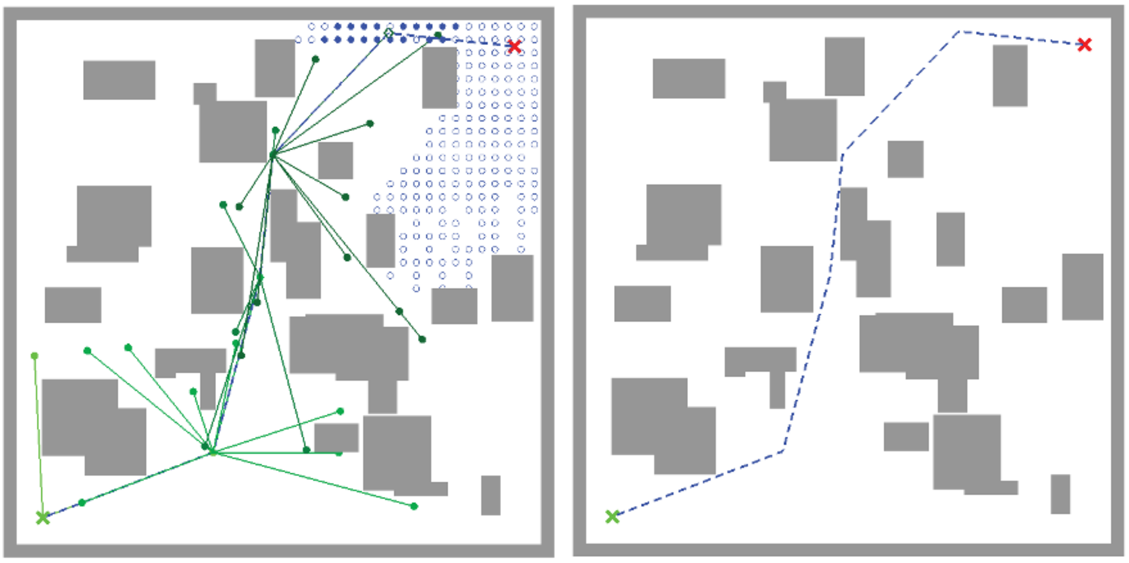
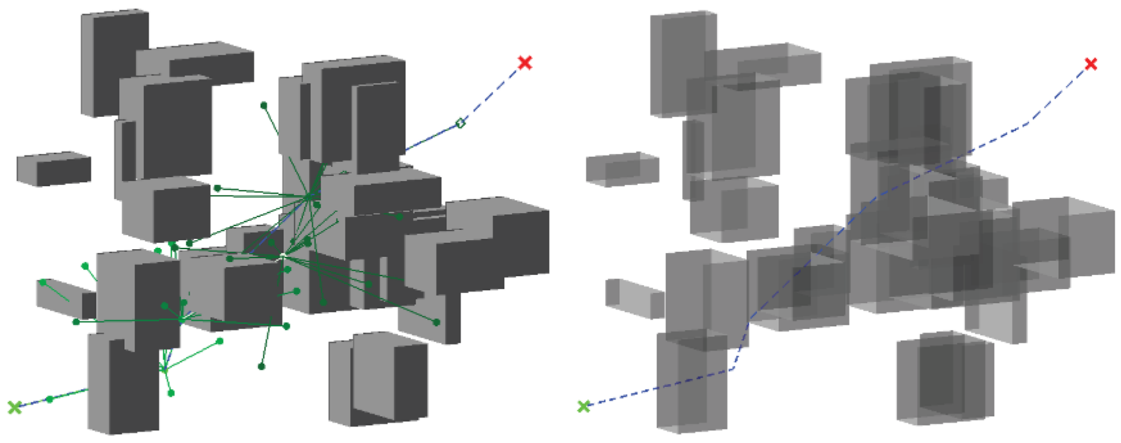

(a) Complete LoS tree (b) Final full path

FIGURE 5.11: An example 2D environment with (a) the LoS tree and common regions and (b) the final LoST path. The start node is represented as a green cross, goal node a red cross, green dots and their connecting lines are intermediate nodes, the empty blue circles represent the visible region from the goal node, the filled blue circles are common regions and the blue dotted line is the final path found through the environment.



(a) Complete LoS tree (b) Final full path

Figure 5.12: (a) An example 3D environment with the LoS tree and (b) the final path found. The start node is represented as a green cross, goal node a red cross, green dots and their connecting lines are intermediate node and the blue dotted line is the final path found through the environment

environments the LoST algorithm and the RRT algorithms were called three times each environment to account for their probabilistic natures with all successful results recorded. The results for the 2D environments are shown in Tables 5.2 to 5.7 with the results for the 3D environments shown in Tables 5.8 to 5.13.

The maximum index level of the LoST algorithm influences the tested results. The maximum index level specifies how far along a branch of a LoS tree is searched before terminating. At higher levels branches are searched further, increasing the chances of finding a common region and a solution. Although while higher maximum index levels do yield higher completion rates, the LoST algorithm may exhaustively search regions within the environment which will not yield a successful solution, such as areas with obstacles in close proximity which form barriers. In this case a lower maximum index level would be beneficial so as to prevent a large number of collision checks and excessive total distance travelled. However if the maximum index level is too low the search may end prematurely before a path is found. An appropriate maximum index level should then be selected which gives a high success rate and maintains a reduced number of total nodes found and total distance travelled. In the 3D environments (Tables 5.8 to 5.13) it can be seen that at certain maximum index levels with high completion rates, the LoST algorithm finds solutions with a reduced total number of nodes found and total distance travelled compared

Table 5.2: 2D environments with 20 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	44.00%	94.00%	97.00%	97.00%	97.00%	93.92%
Average number of nodes in final path	1.00	1.70	1.99	2.10	2.20	4.80
Average total number of nodes found	28.27	29.50	35.11	34.61	33.52	102.99
Average collision checks	1423.64	1508.94	1825.98	1803.71	1762.89	1316.97
Average final path length	2.70m	2.93m	3.04m	3.08m	3.12m	3.43m
Average total distance travelled	4.35m	4.06m	5.11m	4.56m	4.37m	-

Table 5.3: 2D environments with 40 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	5.00%	50.00%	84.00%	91.00%	93.00%	81.17%
Average number of nodes in final path	1.00	1.98	2.65	3.26	3.31	8.49
Average total number of nodes found	34.20	82.90	118.39	317.53	810.70	465.26
Average collision checks	1512.00	3988.80	5922.86	15436.48	39374.19	2403.79
Average final path length	2.67m	3.13m	3.25m	3.50m	3.55m	3.69m
Average total distance travelled	4.77m	17.27m	23.90m	69.18m	176.72m	-

Table 5.4: 2D environments with 60 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	0.00%	13.00%	45.00%	67.00%	75.00%	51.33%
Average number of nodes in final path	-	2.00	2.78	3.76	4.36	10.77
Average total number of nodes found	-	71.62	362.69	1262.01	3263.13	952.99
Average collision checks	-	3793.85	19440.00	68555.82	169419.20	3866.96
Average final path length	-	3.18m	3.37m	3.64m	3.82m	3.81m
Average total distance travelled	-	15.46m	85.58m	298.29m	731.58m	-

Table 5.5: 2D environments with 80 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	0.00%	2.00%	5.00%	36.00%	41.00%	15.92%
Average number of nodes in final path	-	2.00	3.00	3.86	4.71	10.56
Average total number of nodes found	-	36.00	212.60	1035.72	2432.98	1091.35
Average collision checks	-	2340.00	14040.00	67030.00	150989.27	4282.06
Average final path length	-	3.17m	3.25m	3.58m	3.88m	3.79m
Average total distance travelled	-	7.89m	57.09m	287.40m	605.93m	-

Table 5.6: 2D environments with 100 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	0.00%	0.00%	1.00%	8.00%	25.00%	3.75%
Average number of nodes in final path	-	-	3.00	3.75	4.68	9.84
Average total number of nodes found	-	-	115.00	513.13	4607.60	1098.82
Average collision checks	-	-	6480.00	35370.00	280411.20	4304.45
Average final path length	-	-	3.49m	3.51m	3.72m	3.78m
Average total distance travelled	-	-	26.46m	156.40m	1153.36m	-

Table 5.7: 2D environments with 120 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	0.00%	0.00%	1.00%	3.00%	10.00%	1.58%
Average number of nodes in final path	-	-	3.00	4.00	4.90	9.53
Average total number of nodes found	-	-	98.00	1103.33	1620.00	1146.90
Average collision checks	-	-	6480.00	76800.00	108756.00	4448.71
Average final path length	-	-	3.36m	3.40m	3.61m	3.75m
Average total distance travelled	-	-	23.41m	320.51m	419.78m	-

TABLE 5.8: 3D environments with 20 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	0.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Average number of nodes in final path	-	2.00	2.18	2.22	2.21	2.13
Average total number of nodes found	-	80.53	61.63	60.54	61.32	3.11
Average collision checks	-	7352.64	6153.80	6376.32	6912.00	1017.34
Average final path length	-	2.79m	2.83m	2.88m	2.87m	3.14m
Average total distance travelled	-	7.06m	4.59m	4.49m	4.83m	-

TABLE 5.9: 3D environments with 40 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	0.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Average number of nodes in final path	-	2.00	2.22	2.24	2.23	3.15
Average total number of nodes found	-	165.53	81.95	85.01	90.67	7.05
Average collision checks	-	10874.94	6596.64	7758.72	10640.16	1029.15
Average final path length	-	2.77m	2.84m	2.86m	2.83m	3.38m
Average total distance travelled	-	12.40m	4.93m	5.45m	6.52m	-

TABLE 5.10: 3D environments with 60 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST_1	LoST_2	LoST_3	LoST_4	LoST_5	RRT
Completion rate	0.00%	94.33%	99.00%	100.00%	100.00%	98.50%
Average number of nodes in final path	-	2.00	2.36	2.33	2.39	3.86
Average total number of nodes found	-	245.58	100.73	94.40	101.99	11.71
Average collision checks	-	14778.06	7780.36	7655.04	11491.20	1043.13
Average final path length	-	2.79m	2.88m	2.88m	2.90m	3.55m
Average total distance travelled	-	18.18m	6.16m	5.50m	7.08m	-

Table 5.11: 3D environments with 80 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST$_1$	LoST$_2$	LoST$_3$	LoST$_4$	LoST$_5$	RRT
Completion rate	0.00%	93.00%	99.00%	99.00%	100.00%	98.00%
Average number of nodes in final path	-	2.00	2.50	2.63	2.63	4.39
Average total number of nodes found	-	380.09	165.11	111.56	108.16	19.27
Average collision checks	-	22213.89	11144.73	10573.09	12765.60	1065.82
Average final path length	-	2.80m	2.92m	2.98m	2.97m	3.63m
Average total distance travelled	-	28.89m	10.47m	7.24m	8.23m	-

Table 5.12: 3D environments with 100 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST$_1$	LoST$_2$	LoST$_3$	LoST$_4$	LoST$_5$	RRT
Completion rate	0.00%	82.33%	99.00%	99.00%	100.00%	81.00%
Average number of nodes in final path	-	2.00	2.67	2.85	2.86	5.20
Average total number of nodes found	-	1036.88	274.94	117.13	117.75	37.71
Average collision checks	-	58239.64	19005.88	10763.94	13193.28	1086.34
Average final path length	-	2.78m	3.00m	3.05m	3.02m	3.83m
Average total distance travelled	-	81.11m	18.65m	7.75m	8.31m	-

Table 5.13: 3D environments with 120 random obstacles - averaged over 100 runs. LoST algorithm run at various maximum index levels.

Type	LoST$_1$	LoST$_2$	LoST$_3$	LoST$_4$	LoST$_5$	RRT
Completion rate	0.00%	64.33%	95.33%	96.00%	97.00%	74.70%
Average number of nodes in final path	-	2.00	2.81	3.05	3.11	5.45
Average total number of nodes found	-	1308.98	437.51	213.66	128.10	46.61
Average collision checks	-	77750.59	29162.26	17153.28	16774.56	1113.91
Average final path length	-	2.77m	3.03m	3.10m	3.11m	3.77m
Average total distance travelled	-	107.99m	31.10m	14.67m	10.53m	-

to others; with 20 and 40 obstacles this occurs at maximum index level 3; with 60, 80 and 100 obstacles at a maximum index level 4; and with 120 obstacles at maximum index level 5. In 2D environments (Tables 5.2 to 5.7) this is less evident as completion rates are not as considerably high.

In the majority of cases, the LoST algorithm develops paths with fewer nodes than the RRT algorithm with the variation between the two algorithms more obvious with an increasing number of obstacles. In 2D environments, at a minimum the RRT algorithm requires on average more than twice the number of nodes in the final path compared to the LoST algorithm. While in 3D environments the variation between the two algorithms is lower as the relative amount of free space is increased. The RRT algorithm does find paths with fewer nodes in the 20 obstacle case, in all other cases the LoST algorithm's paths have fewer. As the concentration of obstacles increases, both algorithms require more nodes to find the final path. By reducing the amount of free space, finding interconnecting nodes becomes more difficult for the RRT algorithm which increases the number of nodes required to find a path. The LoST algorithm has a decreased chance of finding common regions as a node's visible region is more likely to be hindered by additional obstacles which similarly increases the number of nodes required. However, significantly more nodes are found as more obstacle edges are now visible. While this may be necessary to navigate through complex environments, it does increases computational times. This allows the LoST algorithm to intrinsically direct paths around obstacles allowing it to converge upon a solution with fewer nodes in the final path as opposed to the RRT algorithm.

In comparison to lower maximum index levels, when a LoST algorithm is run with higher maximum index levels the number of nodes in the final path will be higher. Similar LoS trees will be generated on the same environment with LoST algorithms using higher maximum index levels mimicking paths followed by that of LoST algorithms with lower levels. Any common region found by the lower level LoST algorithm will also be found by the higher and terminate at similar times. However, the higher will also potentially find common regions by searching further along LoS tree branches. This will increase the number of nodes in the final path.

For both the LoST and RRT algorithms, the completion rate similarly is related to the amount of free space in the environment. As the amount of free space decreases, connecting nodes becomes more difficult for the RRT algorithm and it is more difficult for the LoST algorithm to find common regions. Although, as more obstacles are introduced the LoST algorithm finds a higher number of intermediate nodes which can improve the completion rate. As previously mentioned, the maximum index level highly influences the LoST algorithm. While higher maximum index levels will result in higher completion rates, a minimum level is necessary to find solutions. As seen in the results when the minimum level is too low, the LoST algorithm terminates prematurely significantly reducing the completion rate or potentially finding no solution at all.

The large number of total nodes found by the LoST algorithm also occurs due to the indiscriminate nature of the ray cast operation. As the ray casting is omnidirectional with no considerations made to the location of the current node, subsequent intermediate node checks may find new intermediate nodes in close proximity to the current node or other nodes within the tree. Subsequently, the visible regions from these nodes may be similar when compared to nodes within their proximity providing limited new information. This may be beneficial in some cases where a slight change in position will reveal additional, unseen regions. However in the majority of cases it will cause unnecessary computation.

The LoST algorithm performs a significantly larger number of collision checks when compared to the RRT algorithm. The RRT algorithm performs collision checks only when attempting to connect nodes to an existing tree, whereas each individual ray cast operation is a single collision check with omnidirectional ray casting required to determine a node's visible region. To minimise the number of collision checks, the resolution between rays should be specified to minimise the total number of ray casting operations while ensuring the surrounding region is complete visible. In 3D environments the number of collision checks increases as the visible region is volumetric.

The final path length, as traversed by the simulated point robot, is related to the number of nodes in the final path. As the complexity of the environment increases more nodes are required to find a path as more obstacles potentially block the most direct route. At lower maximum index levels the final path length is generally lower as the number of nodes in the final path is lower. However, at these levels solutions are not guaranteed. The LoST algorithm also develops shorter paths as nodes are both placed directly beyond obstacles and intelligently selected based on weighting criteria as opposed to the RRT algorithm's random seeding.

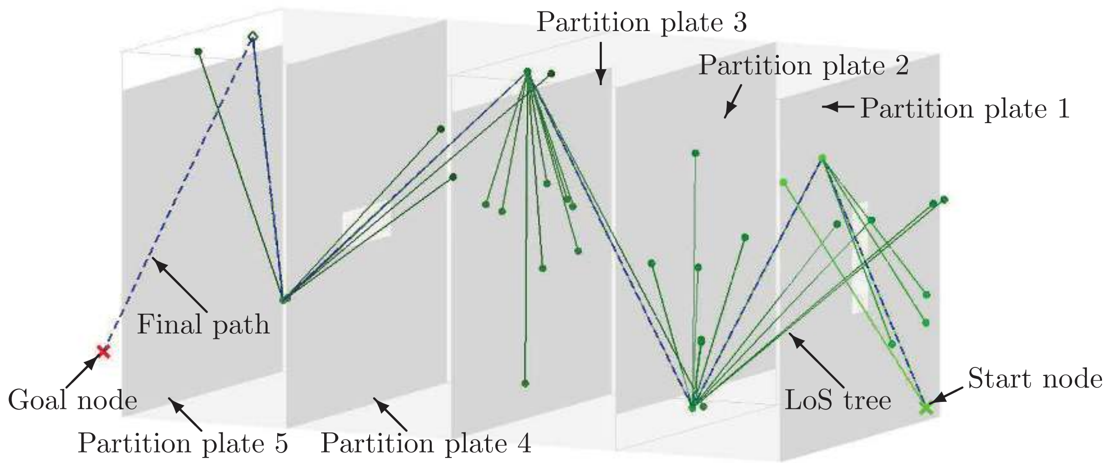
Similarly as the environmental complexity increases the total distance travelled also increases as more nodes are searched from by the simulated point robot. Although lower maximum index levels do not necessarily equate to a lower total distance travelled. The lowest total distance travelled occurs at each environment's optimal index level. This is more pronounced in the 3D environments. If the maximum index level is too low the simulated robot may terminate searches along branches of the LoS tree prematurely. While if the maximum index level is too high branches will potentially over search regions.

5.4.3 Verifying the LoST Algorithm in a Steel Bridge Tunnel Application Scenario

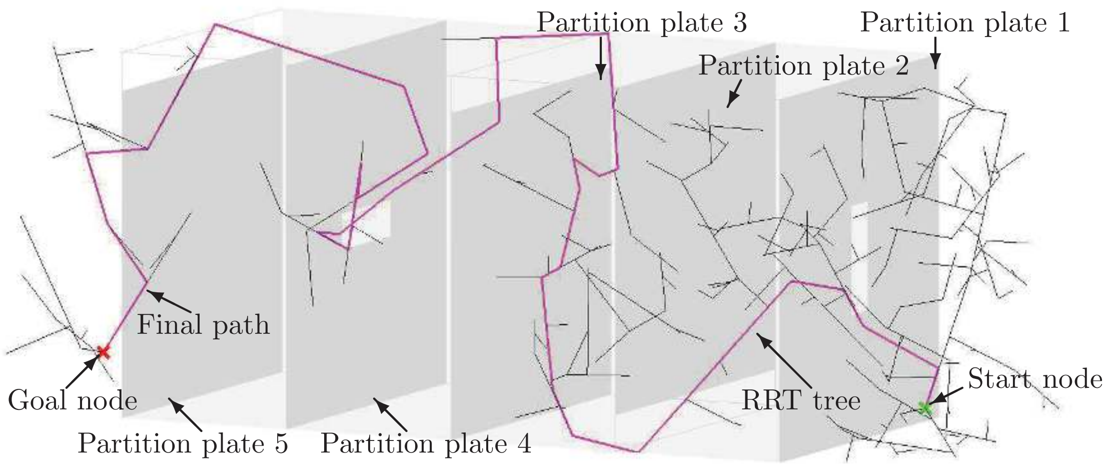
This simulation tests the LoST algorithm in a tunnel environment of a steel bridge (Figure 1.1). This tunnel environment consists of many sections separated by partition plates. The first partition plate, closest to the start node, has a centred longitudinal slit; the second and third partitions have gaps at the bottom and top respectively; the fourth has a centred crosswise slit; and the final has a gap at the top after which the goal node is placed (Figure 5.13). The top and front surfaces of the structure are not shown for clarity. The maximum index level for the LoST algorithm has been set to 5. The top and front surfaces of the structure are not shown for clarity.

The LoST and RRT algorithms were run 100 times to account for the probabilistic natures of the algorithms. The results are shown in Table 5.14. Example trees and the final paths generated by the LoST and RRT algorithms are shown in Figures 5.13a and 5.13b respectively.

Both algorithms had a 100% success rate (Table 5.14), however the LoST algorithm managed to find paths with a significantly lower number of nodes. The LoST algorithm always finds a solution along the first LoS tree branch it searches and in 5 nodes. Each node is used to "see" beyond a single partition plate which accounts for the 5 nodes. The LoST



(a) LoST path through a simulated steel bridge tunnel environment



(b) RRT path through a simulated steel bridge tunnel environment

FIGURE 5.13: Simulated steel bridge tunnel environment with example (a) LoST (b) and RRT paths shown. The start and goal nodes are green and red crosses. The LoS tree connectivity and intermediate nodes are shown green lines and dots with the final path the blue dotted line. The RRT tree is black with the final path magenta.

TABLE 5.14: 3D complex partitioned tunnel - averaged over 100 runs

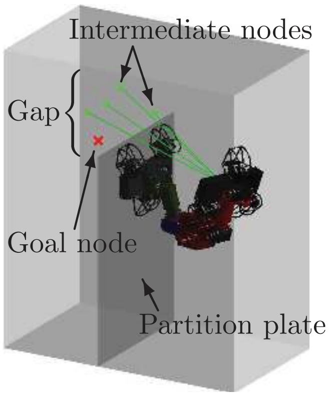
Type	LoST	RRT
Completion rate	100%	100%
Average number of nodes in final path	5.00	27.28
Average total number of nodes found	35.20	759.4
Average collision checks	8479.17	2372.09
Average final path length	5.24m	5.54m
Average total distance travelled	5.24m	-

algorithm is well suited to find paths between gaps, provided the resolution between rays is significantly small. While these small gaps are beneficial for the LoST algorithm, the RRT struggles to find interconnecting nodes through the obstacles as evident by the total number of nodes in the final path and found in total. In comparison the RRT algorithm required on average 446% more nodes in the final path and 2157% more in total. The LoST algorithm's final path length is shorter as intermediate nodes are positioned just beyond the partitions, as opposed to the RRT algorithm's randomly seeded nodes which prevents direct paths and increases the final length travelled. As the total distance travelled is the same as the final path length, the LoST algorithm was able to find solutions without needing to backtrack along its path.

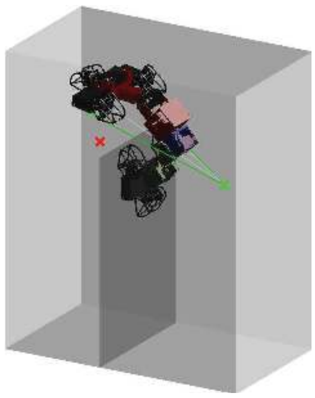
5.4.4 Simulations in a Steel Bridge Tunnel Application Scenario using a Deterministic Robot Model

The combined LoST and 3D-F^2 approach was used to develop a path for a 7DOF inchworm robot within an unknown environment derived from the steel bridge (Figure 1.1b). The steel bridge environment has a partition plate - a solid plate with access through a gap at the top (Figure 5.14a), and is used to verify the functionality of the approach. The inchworm robot's base is placed on the partition plate itself with the goal node positioned on the opposite side. To prevent waypoints from being generated within the base, a bounding box was used to encompass the base and act as an obstacle. A simulated Microsoft Kinect sensor is used to perform the necessary ray cast operations. As the deterministic model of the inchworm robot is used, the 3D-F^2 algorithm is used in this simulation. A major difficulty arises due to the size of the force field surrounding the end-effector in comparison to the size of the partition plate's gap. In a worst case situation, the relative size of the force field is 116% the size of the gap which requires the end-effector to be orientated correctly to bypass the gap.

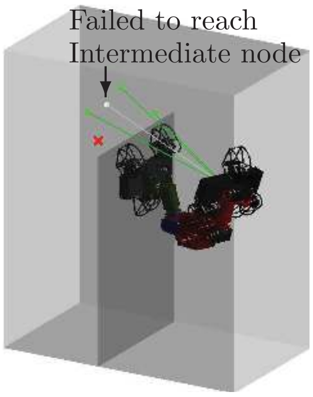
First the region around the inchworm robot's initial pose is scanned with common regions and intermediate nodes determined (Figure 5.14a). The scan data is shown in Figure 5.15. The inchworm robot attempts to find a smooth, collision-free path to the highest weighted waypoint which, in this case, is an intermediate node. However, the 3D-F^2 algorithm enters



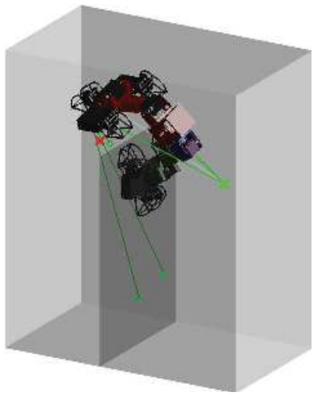
(a) Initial pose



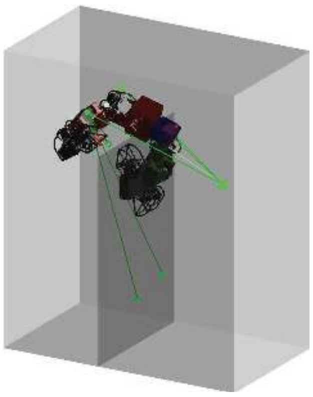
(b) Final pose in failed attempt



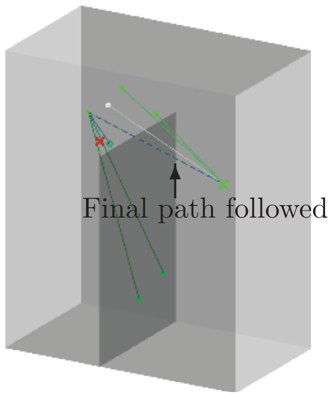
(c) Return to the initial pose



(d) Pose successfully reached beyond partition



(e) Final pose



(f) Final LoS tree

FIGURE 5.14: The simulated partition plate environment used to develop a collision free path for an inchworm robot. From (a) the initial pose the robot attempts to move to the goal node but fails (b) and subsequently returns to the initial pose (c). Eventually (d) a pose is reached beyond the partition from where (e) the robot moves to the goal node. In (f) the final LoS tree shown as a blue dotted line.

a local minima and fails to find a successful path (Figure 5.14b). The inchworm robot backtracks to the pose it started at when beginning to move to the waypoint, which is the start pose, and the LoST algorithm is reinvoked to find the next best waypoint. Potentially the inchworm robot would need to backtrack further if no additional waypoints were available at the current node; however, this is not the case (Figure 5.14c). Additionally the inchworm robot should backtrack along the generated path as generating a new path back may fail. From this pose, the 3D-F^2 algorithm successfully determines a path to the next best waypoint (Figure 5.14d). The LoST algorithm is updated from this pose and the goal node found. The 3D-F^2 algorithm again successfully develops a path for the inchworm robot to follow to the goal node (Figure 5.14e). The final LoS tree is shown in Figure 5.14f. Without the LoST algorithm providing waypoints for the inchworm robot to follow, the 3D-F^2 algorithm alone would not have been able to successfully find a path through the environment.

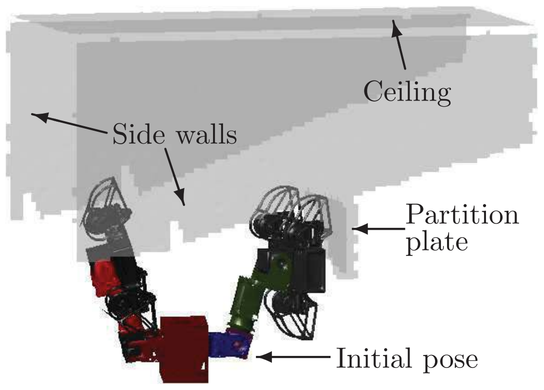
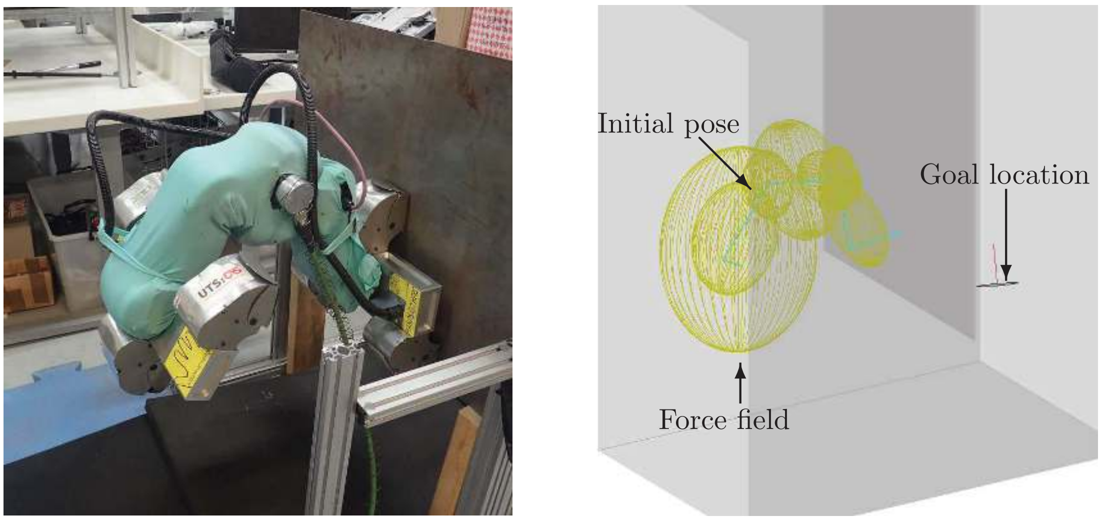


FIGURE 5.15: Simulation scan data taken from the initial pose.

5.4.5 Experiment in an Application Environment with the LoST and 3D-PF^2 Algorithms using a Robot Model with Uncertainty

This experiment uses the LoST algorithm with the 3D-PF^2 algorithm in the partition plate environment extracted from the application scenario (Figure 5.16). The inchworm robot's base is placed on one side of the partition plate and is required to move to a target position on the opposite side. The kinematic model with uncertainty uses the allowed maximum structural uncertainty values in Table 5.15. Paths were developed offline using



(a) The initial pose in the partition plate environment. (b) The derived partition plate environment.

Figure 5.16: The inchworm robot at the initial pose as seen from the base side. The cyan line is the initial robot pose.

the LoST algorithm with the 3D-PF^2 algorithm, then the paths were implemented on the 7DOF inchworm robot.

The LoST algorithm is used to search for intermediate nodes and common region centres around the initial pose (Figure 5.17b). The 3D-PF^2 algorithm finds a successful path to the highest weighted intermediate node for the inchworm robot to follow (Figure 5.17d). From this pose the LoST algorithm finds a common region centre. The 3D-PF^2 algorithm successfully develops a path to the common region centre and the goal node (Figure 5.17f). The inchworm robot after following the final path is shown in Figure 5.18.

Figure 5.19a shows the generated attractive and repulsive forces and joint angles while Figure 5.19b shows the joint angle from the inchworm robot over the generated path

Table 5.15: Table of allowed maximum structural uncertainty values used during experiments.

Allowed maximum structural translation (m)	0.065
Allowed maximum structural rotation (°)	8.5
Allowed maximum structural translation variance (m)	0.04

with the generated attractive and repulsive forces. The difference in the joint angles is accounted for by the inchworm robot controller.

The first set of high repulsive forces in Figures 5.19a and 5.19b occurs as multiple repulsive forces act on the force field surrounding the end-effector while moving towards the intermediate node. As LoS is used to determine the intermediate node, the path to the node is relatively unimpeded and no significantly high repulsive forces are generated. The remaining repulsive forces are generated due to the partition plate entering the force field surrounding the robot body as the end-effector successfully aligns with the goal location.

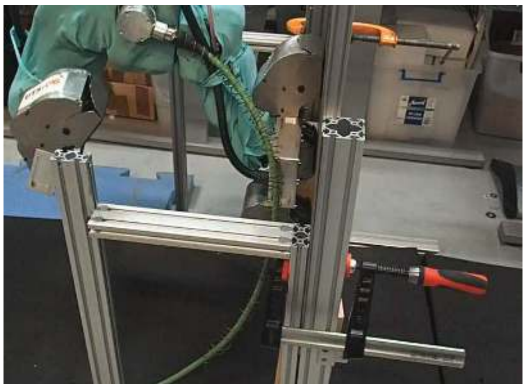
5.5 Discussion

The major issue with the LoST algorithm, which is common for many planners, is the difficulty in finding paths in gaps between closely spaced obstacles. Figures 5.20a and 5.20b show two environmental situations where this may occur. Two factors contribute to this issue. First, if the angle between two adjacent rays, from either ray casting or the sensor, is too low, rays will not penetrate small gaps preventing the LoST algorithm from detecting them. Secondly, if the threshold which dictates if two rays are discontinuous is too low, small gaps will not be detected.

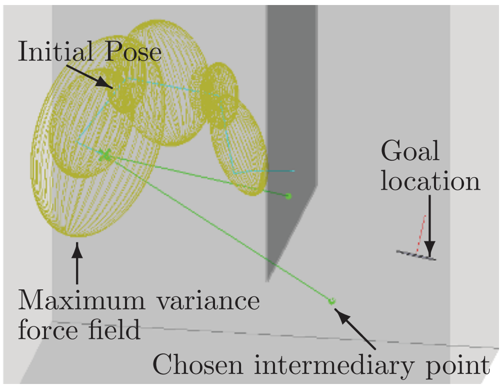
The number of nodes and the final path length found by the RRT algorithm may be improved by optimising the final path. This could be achieved through resampling nodes along the path or by applying smoothing techniques to find shorter paths between nodes. Optimisation techniques could also be applied to the results of the LoST algorithm to improve the path.

5.6 Conclusion

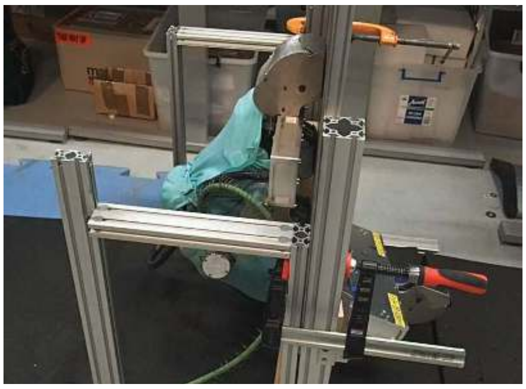
The combined LoST and 3D-PF^2 approach was presented which uses sensor information to navigate a robot through complex environments. This is achieved in a similar manner to how a person perceives a scene whereby their gaze tends towards information-rich regions, such as edges of objects, which then leads their gaze to other regions. The robot "looks"



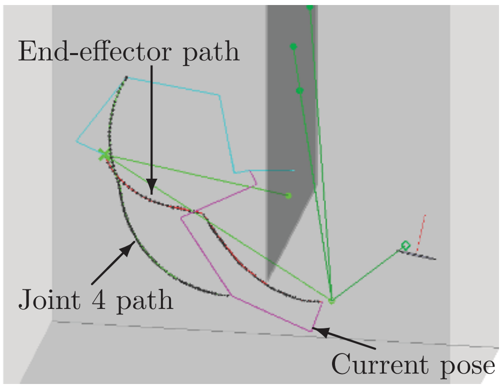
(a) Initial inchworm robot pose.



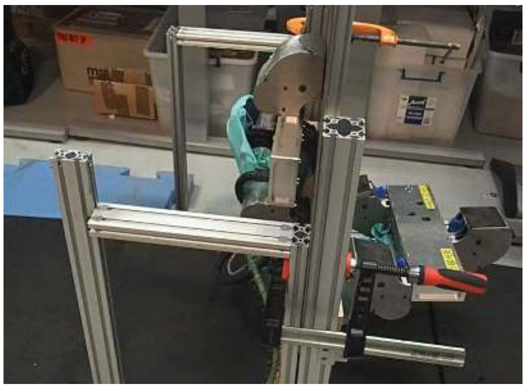
(b) Initial inchworm robot pose with found intermediate nodes.



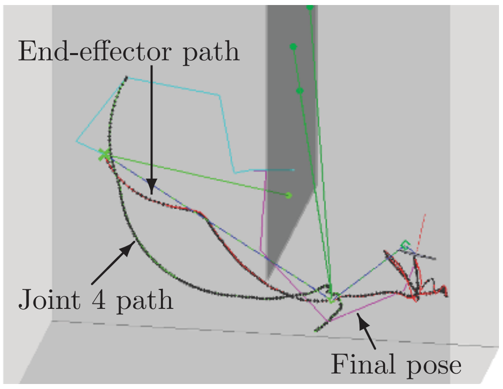
(c) Pose at the supplied intermediate node.



(d) Pose at the supplied intermediate node with found intermediate nodes and common regions.

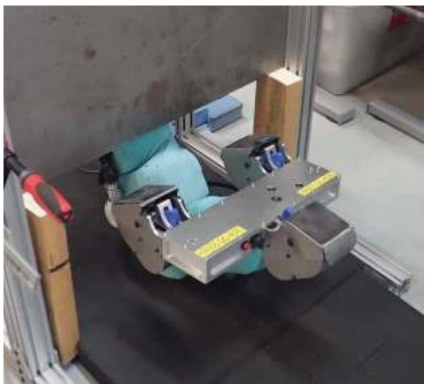


(e) Final pose.

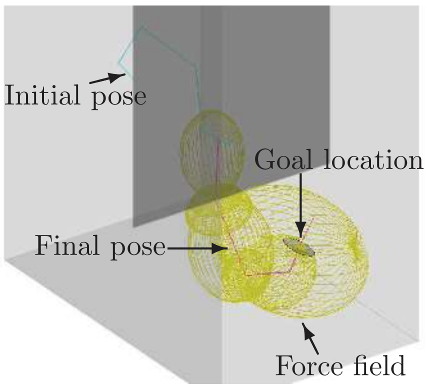


(f) Final pose and final LoST path followed through the environment.

FIGURE 5.17: The inchworm robot successfully following a path generated using the presented approach through a derived partition plate environment. The yellow ellipsoids are the maximum variance force fields, the cyan line is the initial robot pose, the blue lines are the position of the joints over the generated path, the red lines are the position and orientation of the end-effector over the generated path, the magenta lines show the given pose, the green lines are the LoS tree connections, green dots and diamonds are the intermediate nodes and common region centres respectively and the dotted blue line is the followed LoST path.

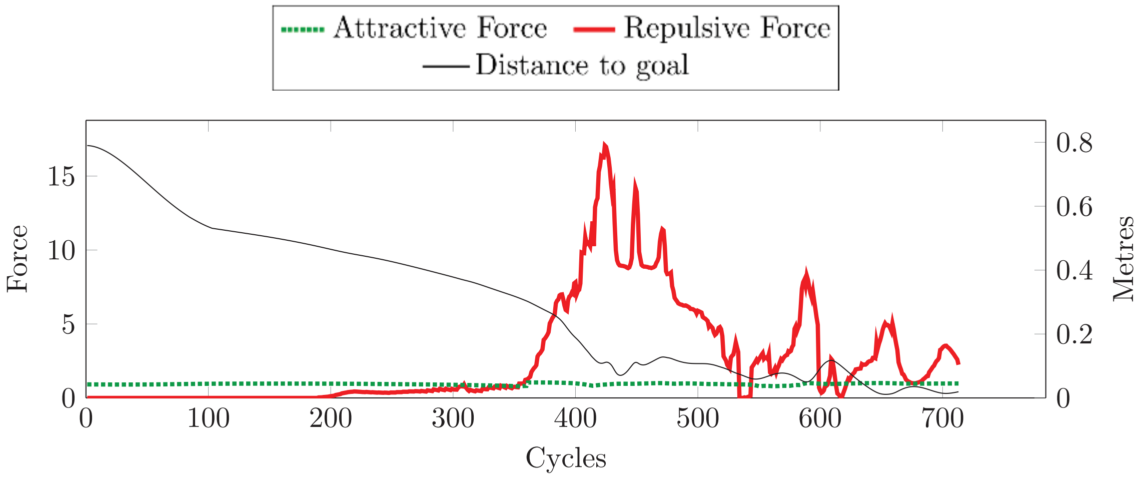


(a) The partition plate environment.



(b) The derived partition plate environment.

FIGURE 5.18: The inchworm robot at the final pose as seen from the goal side. The cyan line is the initial robot pose, and the magenta lines show the final pose.



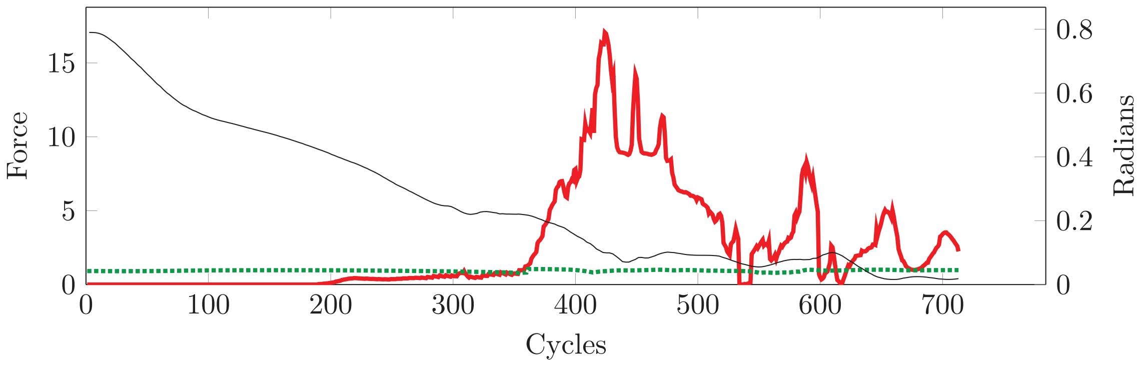


FIGURE 5.19: 5.19a Graph of the generated forces and joint angles in the derived partition plate environment. 5.19b Graph of the generated forces and the feedback from the joint angles on the inchworm robot in the partition plate environment. Allowed maximum structural translation at 0.065m and rotation 8.5°, variance at 0.04m and allowed maximum hand position variance at 0.05m using the LoST and 3D-PF^2 algorithms.

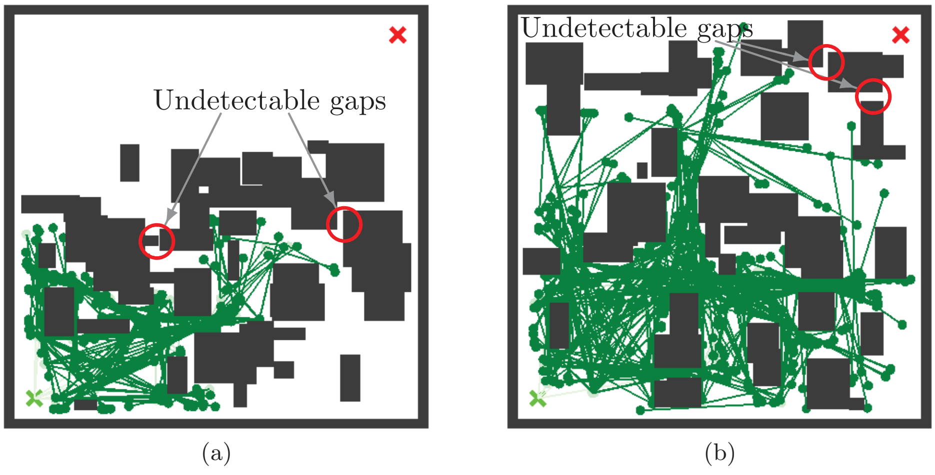


FIGURE 5.20: Two example environments with unreachable goal nodes due to undetectable gaps.

around its current position to determine obstacle edges and continue searching for a goal node. Results showed that the LoST algorithm is capable of performing path planning for navigation of a robot through complex environments by determining waypoints in Cartesian space. Simulations are performed using the 3D-F^2 algorithm with an experiment performed using the 3D-PF^2 algorithm with a 7DOF inchworm robot.

The LoST algorithm was also compared to an RRT algorithm. In all cases studied in this thesis the LoST algorithm generated paths with fewer intermediate nodes, and a shorter final path length than the tested RRT algorithm. Future work on the LoST algorithm includes methods to end branches of the LoS tree when subsequent searches will not provide significant new information, and filter out new nodes if they are within close proximity to existing nodes within the tree. These improvements would reduce the number of collision checks and allow a solution to be found quicker.

Future work includes development of methods for handling uncertainties other than those described in this thesis such as sensor uncertainty. Additionally, while the algorithms are capable of functioning in real-time, it has not been implemented in the inchworm robot. Further work is required to integrate the algorithms into the inchworm robot architecture.

Chapter 6

Conclusion

This thesis has developed models and algorithms used in an approach for planning smooth collision-free paths for a multi-link serial robot with structural and hand position uncertainties. These uncertainties were incorporated into a model for a multi-link serial robot. This model described the mean joint locations as variations from the deterministic model with matrices used to represent the maximum variation in the possible robot locations. A 3D Probabilistic Force Field (3D-PF^2) algorithm used this model to develop collision-free, short time-horizon paths for a multi-link serial robot. The 3D-PF^2 algorithm used force fields surrounding each link for collision avoidance. The force fields were sized to encompass the dimensions of the link and increased in size based on the uncertainty in the model. A Line of Sight Tree (LoST) algorithm was used for longer time-horizon motion planning by determining waypoints through an environment. These waypoints were then supplied to the 3D-PF^2 algorithm for short time-horizon motion planning. The 3D-PF^2 and LoST algorithms combine both short and longer time-horizon planning to develop paths for a multi-link serial robot with structural and hand position uncertainties.

6.1 Summary of Contributions

6.1.1 A Probabilistic Model with Structural and Hand Position Uncertainties

A model was developed for representing structural and hand position uncertainties in a multi-link serial robot. The structural and hand position uncertainties are described through mean joint locations and matrices to represent the maximum variance at these joint locations. The mean and maximum variance for each type of uncertainty is estimated by interpolation based on graphs which relate the current joint positions to a worst case state. The worst case state is determined such that, at that state, the variance for a specific type of uncertainty is at its highest value. The mean and maximum variance at the worst case state were determined experimentally. The model considered the mean joint locations as variations from the coordinate frame of reference of their joints in the deterministic model of the serial robot.

6.1.2 A 3D Probabilistic Force Field Algorithm

A 3D-PF^2 algorithm was used to develop short time-horizon paths for a multi-link serial robot with uncertainty in the joint coordinate frame of reference. The algorithm used a combination of an attractive force and repulsive forces to direct the robot to a goal location. The attractive force “pulled” the end-effector towards the goal location while repulsive forces “pushed” links away from potential collisions based on the proximity of the collisions to force fields surrounding each link. These force fields were sized to encompass the dimensions of the link and the uncertainty in the link’s joint’s coordinate frame of reference. When the uncertainty increased, the force fields increased. Simulations and experiments were performed which demonstrated the 3D-PF^2 algorithm capacity and suitability as a short time-horizon planner.

6.1.3 A Line of Sight Tree Algorithm

A LoST algorithm has been developed for longer time-horizon motion planning. The LoST algorithm generates waypoints as goal locations for the 3D-PF^2 algorithm. Waypoints are found in a manner loosely based on how a person perceives a scene whereby their gaze tends towards information rich regions, such as edges of obstacles, which then leads their gaze to other regions. The LoST algorithm "perceived" the environment by ray-casting at the current robot position and incrementally building a Cartesian-space tree of waypoints from a start node to a goal node. Waypoints are positioned at obstacle edges and, if visible, at the goal node with waypoints given weightings based on criteria. The highest weighted waypoint is selected by the LoST algorithm as a goal for the short time-horizon planner with waypoints associated with the goal node taking precedence over those at obstacle edges. If the waypoint was successfully reached by the short time-horizon planner, and it was the goal node, the LoST algorithm was successful. If the waypoint was successfully reached for the first time, ray-casting would be performed and the Cartesian-space tree further built upon. If the waypoint was not reached the short time-horizon planner backtracks, the waypoints weighting is set to zero and the next highest waypoint is selected. If all waypoints have been visited and the goal node has not been reached the LoST algorithm fails. Simulations and experiments were used to verify the LoST algorithm as a longer time-horizon planner both as a basic planner and with the 3D-PF^2 algorithm used as short time-horizon planners.

6.2 Discussion of Limitations and Future Work

This thesis has developed an approach for safe motion planning for a multi-link serial robot with structural and hand position uncertainties. This was achieved by developing a model to represent these uncertainties for use with a 3D-PF^2 algorithm for short time-horizon planning and a LoST algorithm for longer time-horizon planning which allowed smooth collision-free paths through an environment to be generated. However, this approach does have a number of limitations which prevent its adoption as a fully robust solution.

The 3D-PF^2 algorithm has a number of variables which vary the generated attractive and repulsive forces. If the attractive force is larger than the repulsive forces, collisions are more likely to occur. If the repulsive forces are too large, motion past or around obstacles may be prevented, resulting in failure. In the case of poorly tuned parameters, changes in the generated forces may result in erratic motion or slow response. Parameter tuning is required to ensure a robust solution with different parameters required for a robot and for a given environment.

Waypoints found by the LoST algorithm are described by their positions in the Cartesian coordinate frame. Waypoints are then used by the 3D-PF^2 algorithm without goal rotational vectors. This may increase the chances of a waypoint being reached as the final end-effector orientation is not restricted. However, in some instances gaps in the environment may require the end-effector to be orientated in a specific orientation. Determining this orientation is not currently possible using the LoST algorithm.

The LoST algorithm can be used in an unknown, static environment provided the robot model is known. Assuming a deterministic robot model, sensor scans may be used to build a map of the environment which the 3D-PF^2 algorithm can use to generate repulsive forces and perform collision avoidance. When the robot location is unknown, the 3D-PF^2 algorithm cannot be used in an unknown environment as maps generated from cumulative scans will be inherently uncertain. This prevents the approach from being used with a probabilistic robot model in an unknown environment.

The approach developed in this thesis is used to generate smooth, collision-free paths for a multi-link serial robot with structural and hand position uncertainties. However, there are a number of areas which can be improved to develop more robust solutions.

The models and algorithms developed in this thesis have been specifically developed for the task of path planning. However, it may be possible to use information gathered through the LoST algorithm to facilitate other tasks. This could include exploration or localisation. Additionally, this approach assumed the goal locations were known. Further research could be conducted into determining these goals based on given tasks such as moving along portions of the substructure of an environment or automated inspection. This could be further extended into intelligently determining orientations for waypoints which are used by the 3D-PF^2 algorithm to bypass obstacles in the environment when required.

This thesis does not use sensor information to verify surface attachment as the 7DOF inchworm robot does not have a sensor capable of safely and accurately determining the distance to the surface. Additional testing and verification is required to ensure that surface attachment is possible once a sensor is installed with this capability.

Currently only uncertainty in the joint coordinate frame of reference is considered in this thesis. A research topic could be developed which also considers control system, sensor and environmental uncertainty. This would be useful in developing a more robust solution and would allow the current iteration of the approach to be considered in unknown environments. However, uncertainty that could be expressed in a similar manner to the structural and hand position uncertainties could be readily adapted.

The algorithms presented in this thesis are intended for static environments only. Future work could be extending the algorithms to dynamic environments. Provided the expected sensory information is available, the 3D-PF^2 may be adapted to react to dynamic obstacles. While the LoST algorithm may need to be modified to distinguish between obstacles edges from dynamic or static obstacles.

Bibliography

[1] Ward, P., Paul, G., Quin, P., Pagano, D., Yang, C.-H., Liu, D., Waldron, K., Dissanayake, G., Brooks, P., Mann, P., Kaluarachchi, W., Manamperi, P., and Matkovic, L. Climbing Robot For Steel Bridge Inspection: Design Challenges. In *9th Austroads Bridge Conference*, pages 1–12, Sydney, New South Wales, 2014.

[2] New South Wales Government. WorkCover. `http://www.workcover.nsw.gov.au/`, June 2011.

[3] Romey, P., Nhan, A., Williams, K., and Dunn, M. Sydney Harbour Bridge Conservation Management Plan 2007. Technical report, Roads and Traffic Authority, 2007.

[4] Sintov, A., Avramovich, T., and Shapiro, A. Design and Motion Planning of an Autonomous Climbing Robot with Claws. *Robotics and Autonomous Systems*, 59 (11):1008 – 1019, 2011.

[5] Dung, N. A. and Shimada, A. A Path-planning Algorithm for Humanoid Climbing Robot using Kinect Sensor. In *2014 Proceedings of the SICE Annual Conference (SICE)*, pages 1549–1554, Japan, Sept 2014.

[6] Murphy, M. P., Kute, C., Meng, Y., and Sitti, M. Waalbot II: Adhesion Recovery and Improved Performance of a Climbing Robot using Fibrillar Adhesives. *The International Journal of Robotics Research*, 30(1):118–133, 2011.

[7] Friesen, J. M., Glick, P., Fanton, M., Manovi, P., Xydes, A., Bewley, T., and Sunspiral, V. The Second Generation Prototype of a Duct Climbing Tensegrity Robot,

DuCTTv2. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2123–2128. IEEE, 2016.

[8] Schmidt, D. and Berns, K. Climbing Robots for Maintenance and Inspections of Vertical Structures A Survey of Design Aspects and Technologies. *Robotics and Autonomous Systems*, 61(12):1288 – 1305, 2013.

[9] Eich, M. and Vogele, T. Design and Control of a Lightweight Magnetic Climbing Robot for Vessel Inspection. In *Proceedings of the 19th Mediterranean Conference Control & Automation (MED)*, pages 1200–1205, Corfu, Greece, June 2011.

[10] Shen, W., Gu, J., and Shen, Y. Proposed Wall Climbing Robot with Permanent Magnetic Tracks for Inspecting Oil Tanks. In *Proceedings of the IEEE International Mechatronics and Automation Conference*, volume 4, pages 2072–2077, Ontario, Canada, July 2005.

[11] Tache, F., Fischer, W., Siegwart, R., Moser, R., and Mondada, F. Compact Magnetic Wheeled Robot With High Mobility for Inspecting Complex Shaped Pipe Structures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 261–266, California, USA, October 2007.

[12] Longo, D. and Muscato, G. The Alicia^3 Climbing Robot: A Three-Module Robot for Automatic Wall Inspection. *IEEE Robotics & Automation Magazine*, 13(1):42–50, 2006.

[13] Mazumdar, A. and Asada, H. Mag-Foot: A Steel Bridge Inspection Robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1691–1696, St. Louis, USA, October 2009.

[14] Peters, G., Pagano, D., Liu, D., and Waldron, K. A Prototype Climbing Robot for Inspection of Complex Ferrous Structures. In *Proceedings of the 13th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 150–156, Nagoya, Japan, August 2010.

[15] Lozano, A., Peters, G., Liu, D. K., and Waldron, K. Study of Ant Locomotion in Surface Transitions for Climbing Robot Design. In *Proceedings of the 14th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 174–181, Paris, France, September 2011.

[16] Lozano, A., Peters, G., and Liu, D. K. Analysis of an Arthropodal System for Design of a Climbing Robot. In *Proceedings of the 28th International Symposium of Automation and Robotics in Construction*, pages 832–838, Seoul, Korea, June 2011.

[17] Pagano, D., Liu, D., and Waldron, K. A Method for Optimal Design of an Inchworm Climbing Robot. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1293–1298, Guangzhou, China, December 2012.

[18] Balaguer, C., Gimenez, A., Huete, A. J., Sabatini, A. M., Topping, M., and Bolmsjo, G. The MATS Robot: Service Climbing Robot for Personal Assistance. *IEEE Robotics & Automation Magazine*, 13(1):51–58, 2006.

[19] Zhu, H., Guan, Y., Cai, C., Jiang, L., Zhang, X., and Zhang, H. W-Climbot: A Modular Biped Wall-Climbing Robot. In *2010 International Conference on Mechatronics and Automation (ICMA)*, pages 1399–1404, Xi'an, China, August 2010.

[20] Longo, D. and Muscato, G. Adhesion Techniques for Climbing Robots: State of the Art and Experimental Considerations. In *Proceedings of the 11th International Conference on Climbing and Walking Robots CLAWAR*, pages 6–28. World Scientific Publishing Co. Pte Ltd, 2008.

[21] Zhu, H., Guan, Y., Chen, S., Su, M., and Zhang, H. Single-step Collision-free Trajectory Planning of Biped Climbing Robots in Spatial Trusses. *Robotics and biomimetics*, 3(1):1–9, 2016.

[22] Yao, J., Huang, Y., Wan, Z., Zhang, L., Sun, C., and Zhang, X. Minimum-time Trajectory Planning for an Inchworm-like Climbing Robot Based on Quantum-behaved Particle Swarm Optimization. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, pages 1–12, 2016.

[23] Pagano, D. and Liu, D. An Approach for Real-time Motion Planning of an Inchworm Robot in Complex Steel Bridge Environments. In *Robotica*, pages 1–30. Cambridge Univ Press, 2015.

[24] Kurniawati, H., Bandyopadhyay, T., and Patrikalakis, N. M. Global Motion Planning Under Uncertain Motion, Sensing, and Environment Map. *Autonomous Robots*, 33(3):255–272, 2012.

[25] Stachniss, C., Grisetti, G., and Burgard, W. Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. In *Robotics: Science and Systems*, volume 2, pages 65–72, 2005.

[26] van den Berg, J., Abbeel, P., and Goldberg, K. LQG-MP: Optimized Path Planning For Robots with Motion Uncertainty and Imperfect State Information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

[27] Malzahn, J., Phung, A. S., Hoffmann, F., and Bertram, T. Vibration Control of a Multi-Flexible-Link Robot Arm Under Gravity. In *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1249–1254. IEEE, 2011.

[28] Malzahn, J., Reinhart, R., and Bertram, T. Dynamics Identification of a Damped Multi Elastic Link Robot Arm Under Gravity. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2170–2175, May 2014.

[29] Korayem, M., Doosthoseini, M., Kadkhodaei Elyaderani, B., and Shafei, A. Trajectory Closed Loop Control Experiment for a Two-link Elastic Manipulator in Presence of Load. In *2014 International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 522–525, Nov 2014.

[30] Alterovitz, R., Siméon, T., and Goldberg, K. Y. The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty. In *Robotics: Science and systems*, volume 3, pages 233–241, 2007.

[31] Fu-Guang, D., Peng, J., Xin-qian, B., and Hong-jian, W. AUV Local Path Planning Based on Virtual Potential Field. In *IEEE International Conference Mechatronics and Automation*, volume 4, pages 1711–1716, 2005.

[32] Korkmaz, O., Ider, S., and Ozgoren, M. Control of an Underactuated Underwater Vehicle Manipulator System in the Presence of Parametric Uncertainty and Disturbance. In *2013 American Control Conference (ACC)*, pages 578–584, June 2013.

[33] Lightcap, C., Hamner, S., Schmitz, T., and Banks, S. Improved Positioning Accuracy of the PA10-6CE Robot with Geometric and Flexibility Calibration. *IEEE Transactions on Robotics*, 24(2):452–456, April 2008.

[34] Backer, J. D. and Bolmsj, G. Deflection Model for Robotic Friction Stir Welding. *Industrial Robot: An International Journal*, 41(4):365–372, 2014.

[35] Li, J., Ma, H., Yang, C., and Fu, M. Discrete-time Adaptive Control of Robot Manipulator with Payload Uncertainties. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 1971–1976, June 2015.

[36] Hafezipour, M. and Khodaygan, S. An Uncertainty Analysis Method for Error Reduction in End-effector of Spatial Robots with Joint Clearances and Link Dimension Deviations. *International Journal of Computer Integrated Manufacturing*, 30(6): 653–663, 2016.

[37] Zhang, X., Yang, W., Cheng, X., and Chen, Y. Stiffness Identification for Serial Robot Manipulator Based on Uncertainty Approach. In Jeschke, S., Liu, H., and Schilberg, D., editors, *Intelligent Robotics and Applications*, volume 7102 of *Lecture Notes in Computer Science*, pages 378–388. Springer Berlin Heidelberg, 2011.

[38] Schneider, U., Momeni-K, M., Ansaloni, M., and Verl, A. Stiffness Modeling of Industrial Robots for Deformation Compensation in Machining. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 4464–4469, Sept 2014.

[39] Sun, D. and Chen, G. Kinematic Accuracy Analysis of Planar Mechanisms with Clearance Involving Random and Epistemic Uncertainty. *European Journal of Mechanics - A/Solids*, 58:256–261, 2016.

[40] Heidari, H., Haghpanahi, M., and Korayem, M. Payload Maximization for Mobile Flexible Manipulators in an Environment with Obstacle. *Journal of Theoretical and Applied Mechanics*, 53(4):911–923, 2015.

[41] Mayyas, M. Bioinspired Legged-robot Based on Large Deformation of Flexible Skeleton. *Bioinspiration & Biomimetics*, 9(4):1–17, 2014.

[42] Queisser, J., Neumann, K., Rolf, M., Reinhart, R., and Steil, J. An Active Compliant Control Mode for Interaction with a Pneumatic Soft Robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 573–579, Sept 2014.

[43] Prentice, S. and Roy, N. The Belief Roadmap: Efficient Planning in Linear POMDPs by Factoring the Covariance. In *Robotics Research*, pages 293–305. Springer, 2010.

[44] Lee, D., Jeon, H., and Myung, H. Pose Graph SLAM-based Displacement Estimation For A Multiple Structural Displacement Monitoring System. In *2014 14th International Conference on Control, Automation and Systems (ICCAS)*, pages 1395–1400, Oct 2014.

[45] Burns, B. and Brock, O. Sampling-Based Motion Planning With Sensing Uncertainty. In *2007 IEEE International Conference on Robotics and Automation*, pages 3313–3318, April 2007.

[46] Ma, J., Bajracharya, M., Susca, S., Matthies, L., and Malchano, M. Real-time Pose Estimation of a Dynamic Quadruped in GPS-denied Environments for 24-hour Operation. *The International Journal of Robotics Research*, 35(6):631–653, 2015.

[47] Dogar, M. R., Hemrajani, V., Leeds, D., Kane, B., and Srinivasa, S. Proprioceptive Localization for Mobile Manipulators. pages 1–24, 2010.

[48] Bloesch, M., Hutter, M., Hoepflinger, M. A., Leutenegger, S., Gehring, C., Remy, C. D., and Siegwart, R. State Estimation for Legged Robots-consistent Fusion of Leg Kinematics and IMU. *Robotics*, pages 1–17, 2013.

[49] Benallegue, M. and Lamiraux, F. Humanoid Flexibility Deformation can be Efficiently Estimated using only Inertial Measurement Units and Contact Information. In *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 246–251, Nov 2014.

[50] Paul, G., Mao, S., Liu, L., and Xiong, R. Mapping Repetitive Structural Tunnel Environments For a Biologically-Inspired Climbing Robot. In *ASSISTIVE ROBOTICS: Proceedings of the 18th International Conference on CLAWAR 2015*, pages 325–333, 2015.

[51] Xiao, D., Ghosh, B., Xi, N., and Tarn, T. Sensor-Based Hybrid Position/Force Control of a Robot Manipulator in an Uncalibrated Environment. *IEEE Transactions on Control Systems Technology*, 8(4):635 –645, July 2000.

[52] Dolgov, D., Thrun, S., Montemerlo, M., and Diebel, J. Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010.

[53] Kala, R., Shukla, A., and Tiwari, R. Robotic Path Planning in Static Environment using Hierarchical Multi-neuron Heuristic Search and Probability Based Fitness. *Neurocomputing*, 74(1415):2314 – 2335, 2011.

[54] Becker, M., Blatt, F., and Szczerbicka, H. Agent-based Approaches for Exploration and Pathfinding in Unknown Environments. In *2012 IEEE 17th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–4, September 2012.

[55] Um, D., Gutierrez, M., Bustos, P., and Kang, S. Simultaneous Planning and Mapping (SPAM) for a Manipulator by Best Next Move in Unknown Environments. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5273–5278, Nov 2013.

[56] Um, D. and Ryu, D. SPAM for a Manipulator by Best Next Move in Unknown Environments. *ISRN Robotics*, 2013:1–8, 2013.

[57] Nourani-Vatani, N., Bosse, M., Roberts, J., and Dunbabin, M. Practical Path Planning and Obstacle Avoidance for Autonomous Mowing. In *Proceedings of the*

Australasian Conference of Robotics and Automation, pages 1–9, Auckland, New Zealand, 2006.

[58] Ferguson, D., Kalra, N., and Stentz, A. Replanning with RRTs. In *2006 Proceedings of IEEE International Conference on Robotics and Automation*, pages 1243–1248, Orlando, Florida, 2006.

[59] Park, J., Kim, J., and Song, J. Path Planning for a Robot Manipulator based on Probabilistic Roadmap and Reinforcement Learning. *International Journal of Control Automation and Systems*, 5(6):674–680, 2007.

[60] Neuman, B. and Stentz, A. Anytime Policy Planning in Large Dynamic Environments with Interactive Uncertainty. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2670–2677, 2012.

[61] Flacco, F., Kröger, T., Luca, A. D., and Khatib, O. A Depth Space Approach to Human-Robot Collision Avoidance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 338–345, Saint Paul, MN, USA, May 2012.

[62] Vasquez-Gomez, J. I., Sucar, L. E., and Murrieta-Cid, R. View/state Planning for Three-dimensional Object Reconstruction under Uncertainty. *Autonomous Robots*, pages 1–21, 2015.

[63] Missiuro, P. E. and Roy, N. Adapting Probabilistic Roadmaps to Handle Uncertain Maps. In *2006 Proceedings IEEE International Conference on Robotics and Automation*, pages 1261–1267. IEEE, 2006.

[64] Guibas, L., Hsu, D., Kurniawati, H., and Rehman, E. Bounded Uncertainty Roadmaps for Path Planning. In Chirikjian, G., Choset, H., Morales, M., and Murphey, T., editors, *Algorithmic Foundation of Robotics VIII*, volume 57 of *Springer Tracts in Advanced Robotics*, pages 199–215. Springer Berlin Heidelberg, 2010.

[65] Wu, C. The Kinematic Error Model for the Design of Robot Manipulator. In *American Control Conference, 1983*, pages 497–502, June 1983.

[66] Phong, L. D., Choi, J., Lee, W., and Kang, S. A Novel Method for Estimating External Force: Simulation Study with a 4-DOF Robot Manipulator. *International Journal of Precision Engineering and Manufacturing*, 16(4):755–766, 2015.

[67] Zergeroglu, E., Dawson, D., de Queiroz, M., and Nagarkatti, S. Robust Visual-Servo Control of Robot Manipulators in the Presence of Uncertainty. In *1999 Proceedings of the 38th IEEE Conference on Decision and Control*, volume 4, pages 4137–4142, 1999.

[68] Zergeroglu, E., Dawson, D. M., de Queiroz, M. S., and Setlur, P. Robust Visual-Servo Control of Robot Manipulators in the Presence of Uncertainty. *Journal of Robotic Systems*, 20(2):93–106, 2003.

[69] Tao, P. Y., Yang, G., and Tomizuka, M. A Sensor-based Approach for Error Compensation of Industrial Robotic Workcells. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5240–5245, 2012.

[70] Wu, W. and Rao, S. Uncertainty Analysis and Allocation of Joint Tolerances in Robot Manipulators Based on Interval Analysis. *Reliability Engineering & System Safety*, 92(1):54 – 64, 2007.

[71] Guerin, D., Caro, S., Garnier, S., and Girin, A. Optimal Measurement Pose Selection for Joint Stiffness Identification of an Industrial Robot Mounted on a Rail. In *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1722–1727. IEEE, 2014.

[72] Aghili, F. Self-Tuning Cooperative Control of Manipulators with Position/Orientation Uncertainties in the Closed-Kinematic Loop. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4187–4193, September 2011.

[73] Joubair, A., Zhao, L. F., Bigras, P., and Bonev, I. Absolute Accuracy Analysis and Improvement of a Hybrid 6-DOF Medical Robot. *Industrial Robot: An International Journal*, 42(1):44–53, 2015.

[74] Dwivedy, Santosha Kumar and Eberhard, Peter. Dynamic Analysis of Flexible Manipulators, A Literature Review. *Mechanism and machine theory*, 41(7):749–777, 2006.

[75] Pereira, E., Aphale, S., Feliu, V., and Moheimani, S. Integral Resonant Control for Vibration Damping and Precise Tip-Positioning of a Single-Link Flexible Manipulator. *IEEE/ASME Transactions on Mechatronics*, 16(2):232–240, April 2011.

[76] Fazel, M. R., Moghaddam, M. M., and Poshtan, J. Application of GDQ Method in Nonlinear Analysis of a Flexible Manipulator Undergoing Large Deformation. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 227(12):2671–2685, 2013.

[77] Pradhan, S. and Subudhi, B. Nonlinear Adaptive Model Predictive Controller for a Flexible Manipulator: An Experimental Study. *IEEE Transactions on Control Systems Technology*, 22(5):1754–1768, Sept 2014.

[78] San-Millan, A., Cambera, J. C., and Feliu, V. Online Algebraic Identification of the Payload Changes in a Single-Link Flexible Manipulator Moving Under Gravity. In *World Congress*, volume 19, pages 8397–8402, 2014.

[79] Mishra, N., Singh, S., and Nakra, B. Dynamic Analysis of a Single Link Flexible Manipulator Using Lagrangian-assumed Modes Approach. In *2015 International Conference on Industrial Instrumentation and Control (ICIC)*, pages 1144–1149, May 2015.

[80] Mahapatro, K. A., Chavan, A. D., and Suryawanshi, P. V. Extended State Observer Based Control of Flexible Link Manipulator in Presence of Unknown Payload Dynamics. In *2015 International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 506–510, Feb 2015.

[81] Wilson, D., Robinett, R., and Eisler, G. Discrete Dynamic Programming for Optimized Path Planning of Flexible Robots. In *2004 Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004*, volume 3, pages 2918–2923, Sept 2004.

[82] Jiang, X., Yabe, Y., Konno, A., and Uchiyama, M. Vibration Suppression Control of a Flexible Arm using Image Features of Unknown Objects. In *2008. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3783–3788, Sept 2008.

[83] Abiri, R., Nadafi, R., and Kabganian, M. Design, Fabrication, and Nonlinear Control of a Flexible Minirobot Module by Using Shape Memory Alloy Actuators. *Journal of Intelligent Material Systems and Structures*, 27(10):1348–1361, 2015.

[84] Reddy, M. P. P. and Jacob, J. Accurate Modeling and Nonlinear Finite Element Analysis of a Flexible-link Manipulator. *International Journal of Mechanical, Aerospace, Industrial and Mechatronics Engineering*, 8(1):165–170, 2014.

[85] Farid, M. and Cleghorn, W. L. Dynamic Modeling of Multi-flexible-link Planar Manipulators Using Curvaturebased Finite Element Method. *Journal of Vibration and Control*, 20(11):1682–1696, 2014.

[86] Renda, F. and Laschi, C. A General Mechanical Model for Tendon-driven Continuum Manipulators. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3813–3818, May 2012.

[87] Khoshnam, M., Azizian, M., and Patel, R. Modeling of a Steerable Catheter Based on Beam Theory. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4681–4686, May 2012.

[88] Yang, W., Dong, W., and Du, Z. Kinematics Modeling for a Kinematic-mechanics Coupling Continuum Manipulator. In *2014 International Conference on Manipulation, Manufacturing and Measurement on the Nanoscale (3M-NANO)*, pages 95–99, Oct 2014.

[89] Du, Z., Yang, W., and Dong, W. Kinematics Modeling and Performance Optimization of a Kinematic-mechanics Coupled Continuum Manipulator. *Mechatronics*, 31: 196–204, 2015.

[90] Dehghani, M. and Moosavian, S. Dynamics Modeling Of Planar Continuum Robots By Finite Circular Elements For Motion Control. In *AI Robotics (IRANOPEN), 2015*, pages 1–6, April 2015.

[91] Shapiro, Y., Wolf, A., and Kósa, G. Piezoelectric Deflection Sensor for a Bi-Bellows Actuator. *IEEE/ASME Transactions on Mechatronics*, 18(3):1226–1230, June 2013.

[92] Li, Z., Du, R., Yu, H., and Ren, H. Statics Modeling of an Underactuated Wire-driven Flexible Robotic Arm. In *2014 5th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 326–331, Aug 2014.

[93] Katzschmann, R. K., Marchese, A. D., and Rus, D. Autonomous Object Manipulation Using a Soft Planar Grasping Manipulator. *Soft Robotics*, 2(4):155–164, 2015.

[94] Marchese, A. D., Tedrake, R., and Rus, D. Dynamics and Trajectory Optimization for a Soft Spatial Fluidic Elastomer Manipulator. *The International Journal of Robotics Research*, pages 1–8, 2015.

[95] Chung, J.-W., Lee, I.-H., Cho, B.-K., and Oh, J.-H. Posture Stabilization Strategy for a Trotting Point-foot Quadruped Robot. *Journal of Intelligent & Robotic Systems*, 72(3-4):325–341, 2013.

[96] Li, Z., Xiao, S., and Ge, S. S. Fuzzy Approximation Adaptive Control of Quadruped Robots with Kinematics and Dynamics Uncertainties. In *2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 701–706, July 2014.

[97] Wang, P., Li, J., and Zhang, Y. The Nonfragile Controller with Covariance Constraint for Stable Motion of Quadruped Search-Rescue Robot. *Advances in Mechanical Engineering*, 6:10, 2014.

[98] Wang, L., Liu, Z., Chen, C., and Zhang, Y. Interval Type-2 Fuzzy Weighted Support Vector Machine Learning for Energy Efficient Biped Walking. *Applied Intelligence*, 40(3):453–463, 2014.

[99] Degrave, J., Burm, M., Kindermans, P.-J., Dambre, J., and wyffels, F. Transfer Learning of Gaits on a Quadrupedal Robot. *Adaptive Behavior*, 23(2):69–82, 2015.

[100] Zheng, Y., Wang, H., Li, S., Liu, Y., Orin, D., Sohn, K., Jun, Y., and Oh, P. Humanoid Robots Walking on Grass, Sands and Rocks. In *2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, April 2013.

[101] Kim, D., Thomas, G., and Sentis, L. Continuous Cyclic Stepping on 3D Point-foot Biped Robots Via Constant Time to Velocity Reversal. In *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1637–1643, Dec 2014.

[102] Kim, D., Zhao, Y., Thomas, G. C., and Sentis, L. Assessing Whole-Body Operational Space Control in a Point-Foot Series Elastic Biped: Balance on Split Terrain and Undirected Walking. *Computing Research Repository*, abs/1501.02855, 2015.

[103] Zhu, Y.-g., Jin, B., and Li, W. Leg Compliance Control of a Hexapod Robot Based on Improved Adaptive Control in Different Environments. *Journal of Central South University*, 22(3):904–913, 2015.

[104] Liu, Z., Wang, L., Zhang, Y., and Chen, C. P. A {SVM} Controller for the Stable Walking of Biped Robots Based on Small Sample Sizes. *Applied Soft Computing*, 38:738 – 753, 2016.

[105] Yue, M., Wang, S., and Zhang, Y. Adaptive Fuzzy Logic-based Sliding Mode Control for a Nonholonomic Mobile Robot in the Presence Of Dynamic Uncertainties. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 229(11):1979–1988, 2015.

[106] Liu, Y. and Li, Y. Sliding Mode Adaptive Neural-Network Control for Nonholonomic Mobile Modular Manipulators. *Journal of Intelligent and Robotic Systems*, 44(3): 203–224, 2005.

[107] Shojaei, K., Shahri, A. M., and Tarakameh, A. Adaptive Feedback Linearizing Control of Nonholonomic Wheeled Mobile Robots in Presence of Parametric and Nonparametric Uncertainties. *Robotics and Computer-Integrated Manufacturing*, 27 (1):194 – 204, 2011.

[108] Morales, R., Sira-Ramrez, H., and Somolinos, J. Robust Control of Underactuated Wheeled Mobile Manipulators Using GPI Disturbance Observers. *Multibody System Dynamics*, 32(4):511–533, 2014.

[109] Fateh, M. and Shahri, M. A Fuzzy Logic Based Motion Control for Nonholonomic Mobile Manipulator Robots. In *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, pages 613–618, Oct 2014.

[110] Wu, X., Wang, Y., and Dang, X. Robust Adaptive Sliding-mode Control of Condenser-cleaning Mobile Manipulator Using Fuzzy Wavelet Neural Network. *Fuzzy Sets and Systems*, 235:62 – 82, 2014. Theme: Control and Applications.

[111] Long, M. and Nan, W. Adaptive Position Tracking System and Force Control Strategy for Mobile Robot Manipulators Using Fuzzy Wavelet Neural Networks. *Journal of Intelligent & Robotic Systems*, 79(2):175–195, 2015.

[112] Li, Z., Yang, C., Su, C.-Y., and Ye, W. Adaptive Fuzzy-based Motion Generation and Control Of Mobile Under-actuated Manipulators. *Engineering Applications of Artificial Intelligence*, 30:86 – 95, 2014.

[113] Peng, J., Yu, J., and Wang, J. Robust Adaptive Tracking Control for Nonholonomic Mobile Manipulator with Uncertainties. *ISA Transactions*, 53(4):1035 – 1043, 2014. Disturbance Estimation and Mitigation.

[114] Wu, K. and Sun, W. Adaptive Tracking Control for a New Mobile Manipulator Model. In *2014 UKACC International Conference on Control (CONTROL)*, pages 98–103, July 2014.

[115] Nguyen, K.-D. and Dankowicz, H. Adaptive Control of Underactuated Robots with Unmodeled Dynamics. *Robotics and Autonomous Systems*, 64:84 – 99, 2015.

[116] Morales, J., Martinez, J., Mandow, A., Seron, J., Garcia-Cerezo, A., and Pequeño Boter, A. Center of Gravity Estimation and Control for a Field Mobile Robot with a Heavy Manipulator. In *2009 IEEE International Conference on Mechatronics*, pages 1–6, April 2009.

[117] Wen, S., Zheng, W., Zhu, J., Li, X., and Chen, S. Elman Fuzzy Adaptive Control for Obstacle Avoidance of Mobile Robots Using Hybrid Force/Position Incorporation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):603–608, July 2012.

[118] Lee, H.-J. and Lee, M. C. Technique for Localization and Visual Servoing of Mobile Manipulators. In *2009 IEEE International Symposium on Industrial Electronics*, pages 652–657, July 2009.

[119] Hamner, B., Koterba, S., Shi, J., Simmons, R., and Singh, S. An Autonomous Mobile Manipulator For Assembly Tasks. *Autonomous Robots*, 28(1):131–149, 2010.

[120] Hvilshoej, M., Boegh, S., Madsen, O., and Kristiansen, M. Calibration Techniques for Industrial Mobile Manipulators: Theoretical configurations and Best practices. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–7, June 2010.

[121] Cheng, H., Chen, H., and Liu, Y. Object handling using Autonomous Industrial Mobile Manipulator. In *2013 IEEE 3rd Annual International Conference on Cyber Technology in Automation, Control and Intelligent Systems (CYBER)*, pages 36–41, May 2013.

[122] Igawa, H., Tanaka, T., Kaneko, S., Tada, T., Suzuki, S., and Ohmura, I. Base Position Detection of Grape Stem Considering its Displacement for Weeding Robot in Vineyards. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 2567–2572, Oct 2012.

[123] Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-whyte, H. F., and Csorba, M. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation*, 17:229–241, 2001.

[124] Grisetti, G., Stachniss, C., and Burgard, W. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, Feb 2007.

[125] Valencia, R., Morta, M., Andrade-Cetto, J., and Porta, J. Planning Reliable Paths With Pose SLAM. *IEEE Transactions on Robotics*, 29(4):1050–1059, Aug 2013.

[126] Komendera, E. and Correll, N. Precise Assembly of 3D Truss Structures using MLE-based Error Prediction and Correction. *The International Journal of Robotics Research*, 34(13):1622–1644, 2015.

[127] Rucker, D. and Webster, R. Deflection-based Force Sensing for Continuum Robots: A Probabilistic Approach. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3764–3769, Sept 2011.

[128] Pilania, V. and Gupta, K. A Hierarchical and Adaptive Mobile Manipulator Planner with Base Pose Uncertainty. *Autonomous Robots*, 39(1):65–85, 2015.

[129] Thrun, S. and Montemerlo, M. The Graph SLAM Algorithm with Applications to Large-scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

[130] Schmidt, A. and Kasiński, A. The Visual SLAM System for a Hexapod Robot. In Bolc, L., Tadeusiewicz, R., Chmielewski, L., and Wojciechowski, K., editors, *Computer Vision and Graphics*, volume 6375 of *Lecture Notes in Computer Science*, pages 260–267. Springer Berlin Heidelberg, 2010.

[131] Grisetti, G., Kümmerle, R., Stachniss, C., and Burgard, W. A Tutorial on Graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.

[132] Mu, B., Paull, L., Agha-mohammadi, A., Leonard, J. J., and How, J. P. Information-based Active SLAM via Topological Feature Graphs. *Computing Research Repository*, abs/1509.08155:1–10, 2015.

[133] Melchior, N. and Simmons, R. Particle RRT for Path Planning with Uncertainty. In *2007 IEEE International Conference on Robotics and Automation*, pages 1617–1624, April 2007.

[134] Bai, H., Hsu, D., and Lee, W. S. Planning How to Learn. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2853–2859, 2013.

[135] Nazari, V. and Notash, L. Motion Analysis of Manipulators With Uncertainty in Kinematic Parameters. *Journal of Mechanisms and Robotics*, 8(2):1–9, 2016.

[136] Zeinali, M. and Notash, L. Adaptive Sliding Mode Control with Uncertainty Estimator for Robot Manipulators. *Mechanism and Machine Theory*, 45(1):80 – 90, 2010.

[137] Tripathy, N. S., Kar, I., and Paul, K. Finite-time Robust Control of Robot Manipulator: A SDDRE Based Approach. In *Proceedings of the 2015 Conference on Advances In Robotics*, page 56. ACM, 2015.

[138] Jaulin, L. Solving Set-valued Constraint Satisfaction Problems. *Computing*, 94(2-4): 297–311, 2012.

[139] Jaulin, L. A Nonlinear Set Membership Approach for the Localization and Map Building of Underwater Robots. *IEEE Transactions on Robotics*, 25(1):88–98, 2009.

[140] Jaulin, L. Range-Only SLAM With Occupancy Maps: A Set-Membership Approach. *IEEE Transactions on Robotics*, 27(5):1004–1010, Oct 2011.

[141] Yu, W., Zamora, E., and Soria, A. Ellipsoid SLAM: A Novel Set Membership Method for Simultaneous Localization and Mapping. *Autonomous Robots*, pages 1–13, 2015.

[142] Roveda, L., Vicentini, F., Pedrocchi, N., Tosatti, L. M., and Braghin, F. Development of Impedance Control Based Strategies for Light-Weight Manipulator Applications Involving Compliant Interacting Environments and Compliant Bases. In *ASME 2014 12th Biennial Conference on Engineering Systems Design and Analysis*, pages 1–10. American Society of Mechanical Engineers, 2014.

[143] Roveda, L., Vicentini, F., Pedrocchi, N., Braghin, F., and Tosatti, L. Impedance Shaping Controller For Robotic Applications Involving Interacting Compliant Environments and Compliant Robot Bases. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2066–2071, May 2015.

[144] Karydis, K., Poulakakis, I., Sun, J., and Tanner, H. G. Probabilistically Valid Stochastic Extensions of Deterministic Models for Systems with Uncertainty. *The International Journal of Robotics Research*, 34(10):1278–1295, 2015.

[145] Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. In *1985 Proceedings IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, St. Louis, Missouri, March 1985.

[146] Borenstein, J. and Koren, Y. Real-Time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):1179–1187, Sep/Oct 1989.

[147] Koren, Y. and Borenstein, J. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In *1991 Proceedings IEEE International Conference on Robotics and Automation*, volume 2, pages 1398–1404, Sacramento, California, April 1991.

[148] Li, Q., Wang, L., Chen, B., and Zhou, Z. An Improved Artificial Potential Field Method for Solving Local Minimum Problem. In *2011 2nd International Conference on Intelligent Control and Information Processing (ICICIP)*, volume 1, pages 420–424, Harbin, China, July 2011.

[149] Wang, D., Liu, D., Kwok, N., and Waldron, K. A Subgoal-Guided Force Field Method for Robot Navigation. In *2008 IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, pages 488 –493, Oct. 2008.

[150] Olunloyo, V. and Ayomoh, M. Autonomous Mobile Robot Navigation Using Hybrid Virtual Force Field Concept. *European Journal of Scientific Research*, 31(2):204–228, May 2009.

[151] Mujahed, M., Jaddu, H., Fischer, D., and Mertsching, B. Tangential Closest Gap based (TCG) Reactive Obstacle Avoidance Navigation for Cluttered Environments. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, Linkping, Sweden, Oct 2013.

[152] Ghazal, M., Talezadeh, A., Taheri, M., and Nazemi-Zade, M. Obstacle Avoidance Dynamic Motion of Industrial Manipulators. *International Journal of Mechanical Engineering Research & Technology*, 1(1):1–7, 2015.

[153] Hargas, Y., Mokrane, A., Hentout, A., Hachour, O., and Bouzouia, B. Mobile Manipulator Path Planning Based on Artificial Potential Field: Application on RobuTER/ULM. In *2015 4th International Conference on Electrical Engineering (ICEE)*, pages 1–6, Dec 2015.

[154] Huptych, M. and Röck, S. Online Path Planning In Dynamic Environments Using The Curve Shortening Flow Method. *Production Engineering*, 9(5-6):613–621, 2015.

[155] Chotiprayanakul, P., Liu, D., and Dissanayake, G. Human-Robot-Environment Interaction Interface for Robotic Grit-Blasting of Complex Steel Bridges. *Automation in Construction*, 27(0):11 – 23, 2012.

[156] Ouyang, F. and Zhang, T. Virtual Velocity Vector-based Offline Collision-free Path Planning of Industrial Robotic Manipulator. *International Journal of Advanced Robotic Systems*, 12(9):1–14, 2015.

[157] Clifton, M., Paul, G., Kwok, N., Liu, D. K., and Wang, D.-L. Evaluating Performance of Multiple RRTs. In *Proceedings of the IEEE/ASME International Conference Mechatronic and Embedded Systems and Applications*, pages 564–569, Beijing, China, October 2008.

[158] Karaman, S. and Frazzoli, E. Sampling-Based Algorithms for Optimal Motion Planning. *Computing Research Repository*, abs/1105.1186:1–76, 2011.

[159] Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. Informed RRT*: Optimal Incremental Path Planning Focused through an Admissible Ellipsoidal Heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, volume abs/1404.2334, pages 2997–3004, Chicago, Illinois, 2014.

[160] Park, J. H., Park, W. J., Lee, C., Kim, M., Kim, S., and Kim, H. J. Endoscopic Camera Manipulation Planning of a Surgical Robot Using Rapidly-exploring Random Tree Algorithm. In *2015 15th International Conference on Control, Automation and Systems (ICCAS)*, pages 1516–1519, Oct 2015.

[161] Benevides, J. R. S. and Grassi, V. Autonomous Path Planning of Free-Floating Manipulators Using RRT-Based Algorithms. In *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, pages 139–144, Oct 2015.

[162] Lozano-Pérez, T. and Wesley, M. A. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[163] Rashid, A. T., Ali, A. A., Frasca, M., and Fortuna, L. Path Planning with Obstacle Avoidance Based on Visibility Binary Tree Algorithm. *Robotics and Autonomous Systems*, 61(12):1440–1449, 2013.

[164] Garrido, S., Blanco, D., Moreno, L., Abderrahim, M., and Munoz, M. *Sensor-based Global Planning for Mobile Manipulators Navigation using Voronoi Diagram and Fast Marching*. INTECH Open Access Publisher, 2007.

[165] Garrido, S., Moreno, L., Blanco, D., and Jurewicz, P. Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching. *International Journal of Robotics and Automation*, 2(1):42–64, 2011.

[166] Fedorenko, R. and Gurenko, B. Local and Global Motion Planning for Unmanned Surface Vehicle. In *MATEC Web of Conferences*, volume 42, pages 1–6. EDP Sciences, 2016.

[167] Kalra, N., Ferguson, D., and Stentz, A. Incremental Reconstruction of Generalized Voronoi Diagrams on Grids. *Robotics and Autonomous Systems*, 57(2):123 – 128, 2009. Selected papers from 9th International Conference on Intelligent Autonomous Systems (IAS-9).

[168] Menasri, R., Nakib, A., Daachi, B., Oulhadj, H., and Siarry, P. A Trajectory Planning of Redundant Manipulators Based on Bilevel Optimization. *Applied Mathematics and Computation*, 250:934 – 947, 2015.

[169] Schuetz, C., Baur, J., Pfaff, J., Buschmann, T., and Ulbrich, H. Evaluation of a direct optimization method for trajectory planning of a 9-dof redundant fruit-picking manipulator. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2660–2666, May 2015.

[170] Chen, W., Chen, Y., Li, B., Zhang, W., and Chen, K. Design of Redundant Robot Painting System for Long Non-regular Duct. *Industrial Robot: An International Journal*, 43(1):58–64, 2016.

[171] HashemZadeh, S. M., Khorashadizadeh, S., Fateh, M. M., and Hadadzarif, M. Optimal Sliding Mode Control of a Robot Manipulator under Uncertainty Using PSO. *Nonlinear Dynamics*, 84(4):2227–2239, 2016.

[172] Tanzmeister, G., Friedl, M., Wollherr, D., and Buss, M. Path Planning on Grid Maps with Unknown Goal Poses. In *2013 16th International IEEE Conference on Intelligent Transportation Systems-(ITSC)*, pages 430–435, The Hague, Netherlands, 2013. IEEE.

[173] Qureshi, A., Iqbal, K., Qamar, S., Islam, F., Ayaz, Y., and Muhammad, N. Potential Guided Directional-RRT* for Accelerated Motion Planning in Cluttered Environments. In *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, pages 519–524, Takamatsu, Kagawa, Japan, Aug 2013.

[174] Haddadin, S., Belder, R., and Albu-Schaeffer, A. Dynamic Motion Planning for Robots in Partially Unknown Environments. In *Proceedings of the 18th IFAC World Congress*, volume 18 of *World Congress*, pages 6842–6850, Milano, Italy, August 2011. International Federation of Automatic Control.

[175] Jaradat, M. A. K., Garibeh, M. H., and Feilat, E. A. Autonomous Mobile Robot Dynamic Motion Planning Using Hybrid Fuzzy Potential Field. *Soft Computing*, 16 (1):153–164, 2012.

[176] Nia, D. N., Tang, H. S., Karasfi, B., Motlagh, O. R. E., and Kit, A. C. Virtual Force Field Algorithm for a Behaviour-Based Autonomous Robot in Unknown Environments. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(1):51–62, 2011.

[177] Ranjbar, B., Mahmoodi, J., Karbasi, H., Dashti, G., and Omidvar, A. Robot Manipulator Path Planning Based on Intelligent Multi-resolution Potential Field. *International Journal of u-and e-Service, Science and Technology*, 8(1):11–26, 2015.

[178] Bierbaum, A., Rambow, M., Asfour, T., and Dillmann, R. A Potential Field Approach To Dexterous Tactile Exploration Of Unknown Objects. In *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*, pages 360–366. IEEE, 2008.

[179] Guan-chen, L., Jian-qiao, Y., Si-yu, Z., and Wei, Z. Artificial Potential Field Based Receding Horizon Control for Path Planning. In *Control and Decision Conference (CCDC), 2012 24th Chinese*, pages 3665–3669, 2012.

[180] Gillham, M. and Howells, G. A Dynamic Localized Adjustable Force Field Method For Real-time Assistive Non-holonomic Mobile Robotics. *International Journal of Advanced Robotic Systems*, 12:21, 2015.

[181] Wang, H., Liu, Y.-H., and Zhou, D. Dynamic Visual Tracking For Manipulators Using An Uncalibrated Fixed Camera. *IEEE Transactions on Robotics*, 23(3):610–617, 2007.

[182] Owan, P., Garbini, J., and Devasia, S. Uncertainty-based arbitration of human-machine shared control. *Computing Research Repository*, abs/1511.05996:1–8, 2015.

[183] Huang, Y. and Gupta, K. Collision-probability Constrained PRM for a Manipulator with Base Pose Uncertainty. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1426–1432, Oct 2009.

[184] Agha-mohammadi, A.-a., Chakravorty, S., and Amato, N. M. FIRM: Sampling-based Feedback Motion-planning Under Motion Uncertainty and Imperfect Measurements. *The International Journal of Robotics Research*, 33(2):268–304, 2014.

[185] Gonzalez, J. and Stentz, A. Planning with Uncertainty in Position an Optimal and Efficient Planner. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 2435–2442, 2005.

[186] Gonzalez, J. and Stentz, A. Using Linear Landmarks for Path Planning with Uncertainty in Outdoor Environments. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1203–1210, 2009.

[187] Rotella, N., Bloesch, M., Righetti, L., and Schaal, S. State Estimation for a Humanoid Robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 952–958, Sept 2014.

[188] Wan, E. A. and van der Merwe, R. *The Unscented Kalman Filter*, chapter 7, pages 221–280. John Wiley & Sons, Inc., 2002.

[189] Kaelbling, L. P. and Lozano-Pérez, T. Integrated Task and Motion Planning in Belief Space. *The International Journal of Robotics Research*, pages 1–60, 2013.

[190] De Schutter, J., De Laet, T., Rutgeerts, J., Decr, W., Smits, R., Aertbelin, E., Claes, K., and Bruyninckx, H. Constraint-Based Task Specification and Estimation for Sensor-Based Robot Systems in the Presence of Geometric Uncertainty. *The International Journal of Robotics Research*, 26(5):433–455, 2007.

[191] van den Berg, J., Patil, S., Alterovitz, R., Abbeel, P., and Goldberg, K. LQG-Based Planning, Sensing, and Control of Steerable Needles. In Hsu, D., Isler, V., Latombe, J.-C., and Lin, M., editors, *Algorithmic Foundations of Robotics IX*, volume 68 of *Springer Tracts in Advanced Robotics*, pages 373–389. Springer Berlin Heidelberg, 2011.

[192] Yadav, P. S. and Singh, N. Robust Control of Two Link Rigid Manipulator. *International Journal of Information and Electronics Engineering*, 5(3):198, 2015.

[193] Hauser, K. Randomized Belief-Space Replanning in Partially-Observable Continuous Spaces. In Hsu, D., Isler, V., Latombe, J.-C., and Lin, M., editors, *Algorithmic Foundations of Robotics IX*, volume 68 of *Springer Tracts in Advanced Robotics*, pages 193–209. Springer Berlin Heidelberg, 2011.

[194] Van Den Berg, J., Patil, S., and Alterovitz, R. Motion Planning Under Uncertainty Using Iterative Local Optimization in Belief Space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

[195] Du, Y., Hsu, D., Kurniawati, H., Lee, W., Ong, S., and Png, S. A POMDP Approach to Robot Motion Planning under Uncertainty. In *International Conference on Automated Planning and Scheduling, Workshop on Solving Real-World POMDP Problems*, page 8, 2010.

[196] Kurniawati, H., Du, Y., Hsu, D., and Lee, W. S. Motion Planning Under Uncertainty for Robotic Tasks with Long Time Horizons. *The International Journal of Robotics Research*, 30(3):308–323, 2011.

[197] Ross, S., Chaib-draa, B., and Pineau, J. Bayesian Reinforcement Learning in Continuous POMDPs with Application to Robot Navigation. In *IEEE International Conference on Robotics and Automation*, pages 2845–2851, 2008.

[198] Ross, S., Pineau, J., Chaib-draa, B., and Kreitmann, P. A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. *Journal of Machine Learning Research*, 12:1729–1770, July 2011.

[199] Dallaire, P., Besse, C., Ross, S., and Chaib-draa, B. Bayesian Reinforcement Learning in Continuous POMDPs with Gaussian Processes. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2604–2609, Oct 2009.

[200] Peraire, J. Lecture l3 - vectors, matrices and coordinate transformations. `http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec03.pdf`, 2009.

[201] Moshtagh, N. Minimum volume enclosing ellipsoid. `http://au.mathworks.com/matlabcentral/fileexchange/9542-minimum-volume-enclosing-ellipsoid`, 2009.

[202] Portal, R., Dias, J., and de Sousa, L. Contact Detection Between Convex Superquadric Surfaces. *Archive of Mechanical Engineering*, 57(2):165–186, 2010.

[203] Henderson, J. M. Human Gaze Control During Real-world Scene Perception. *Trends in Cognitive Sciences*, 7(11):498 – 504, 2003.

[204] Amanatides, J. and Woo, A. A Fast Voxel Traversal Algorithm for Ray Tracing. In *Eurographics*, pages 3–10, Amsterdam, Netherlands, 1987.

[205] Paul, G., Webb, S., Liu, D., and Dissanayake, G. Autonomous Robot Manipulator-based Exploration and Mapping System for Bridge Maintenance. *Robotics and Autonomous Systems*, 59(7 - 8):543 – 554, 2011.