

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Mechanical and Material Engineering Department



THESIS REPORT

Ph.D.

**TRAJECTORY PLANNING FOR INDUSTRIAL ROBOTS
USING GENETIC ALGORITHMS**

Fares Jawad Abu-Dakka

Valencia, 2011

THESIS REPORT

TRAJECTORY PLANNING FOR INDUSTRIAL ROBOTS USING GENETIC ALGORITHMS

By
Fares Jawad Abu-Dakka

**Dissertation submitted to the
Department of Mechanical and Material Engineering of the
Universitat Politècnica de València
in partial fulfilment of the requirements for the
degree of Doctor of Philosophy**

Advisor
Dr. D. Francisco José Valero Chuliá

Valencia, 2011

THESIS REPORT

TRAJECTORY PLANNING FOR INDUSTRIAL ROBOTS USING GENETIC ALGORITHMS

Name of degree candidate: Fares Jawad Abu-Dakka
Dissertation directed by: Dr. D. Francisco José Valero Chuliá

TRIBUNAL COMMITTEE

PRESIDENT: Dr. D. Juan Ignacio Cuadrado Iglesias
JURIES: Dr. D. Giorgio Figliolini
Dr. D. Javier Ros Ganuza
Dr. D. Juan Carlos García Prada
SECRETARY: Dr. D. Vicente Mata Amela

In Valencia, on 7th of March of 2011

<p><i>In the Name of Allah, the Most Beneficent, the Most Merciful</i></p> <p>(1) Read! In the name of your Lord, Who has created.</p> <p>(2) Has created man from a clot.</p> <p>(3) Read! And your Lord is the Most Generous,</p> <p>(4) Who has taught by the pen.</p> <p>(5) Has taught man what he knew not.</p> <p>(First words of the Holly Quran)</p> <p>[1-4, THE CLOT]</p>	<p><i>En el nombre de Allah, el Misericordioso, el Compasivo</i></p> <p>(1) ¡Lee en el nombre de su Señor que ha creado!</p> <p>(2) Ha creado al hombre de un coágulo.</p> <p>(3) ¡Lee, que tu Señor es el más Generoso!</p> <p>(4) El que enseñó por medio del cálamo,</p> <p>(5) Enseñó al hombre lo que no sabía.</p> <p>(Las primeras palabras del Noble Quran)</p> <p>[1-4, El Coágulo]</p>
---	---

ACKNOWLEDGEMENTS

I would like to express my deeply sincere thanks to my advisor and mentor Dr. Francisco Valero for providing guidance, advice and encouragement throughout the preparation of this thesis. Without his support, this work would never have been completed. No less appreciation goes to Professor Vicente Mata. I have learned more from their scholastic ability and exemplary guidance than I can condense in this acknowledgment.

Thanks also to Dr. Allan Tubaileh, my undergraduate instructor, for his encouragement to continue my education that led to a crucial change in my life. Much appreciation is also due to Mr. Eng. Jose Luis Suñer for his invaluable and continuous technical help in simulation applications.

I am grateful to my many student colleagues of the Department of Mechanical Engineering, Polytechnic University of Valencia, for the familiar environment that they had provided during the years I have stayed in Spain. Special thanks are due to Dr. Nidal Farhat for his support, advice and encouragement. Many thanks for all my friends for their supporting and encouragement, especially Dr. Alaa' Kullab, Mr. Iyad Asad, Mr. Adel Said, and Mr. Karim Thalji.

Last, but not the least, I would like to thank my entire family (parents, sisters, brothers, and my lovely wife) for providing me a loving environment, especially my mother for showing me remarkable tolerance and keeping up with many of my broken promises. This thesis is truly dedicated to my mother for her understanding, love and care. Thousands of miles away and across the Mediterranean Sea she has always been my constant source of encouragement throughout my thirty-two years of learning.

ABSTRACT

In the last decades, many researches have been proposed concerning the path and trajectory planning for manipulators. Path and trajectory planning have important applications in many areas, for example industrial robotics, autonomous systems, virtual prototyping, and computer-aided drug design. On the other hand, the evolutionary algorithms have been applied in this plethora of fields, which motivates the author's interest on its application to the path and trajectory planning for industrial robots.

In this work, an exhaustive search of the existing literature related to the thesis has been carried out, which has served to create a comprehensive database used to perform a detailed historical review of developments since its origins to the current state of the art and the latest trends.

This thesis presents a new methodology that uses genetic algorithms to develop and evaluate path and trajectory planning algorithms. Problem-specific knowledge and heuristic knowledge are incorporated into encoding, evaluation and genetic operators of the genetic algorithm.

This methodology introduces new approaches that aim at solve the problem of path planning and trajectory planning for industrial robotic systems operating in 3D environments with static obstacles. Therefore, two algorithms (somehow, they are similar, but with some variations) are created to solve the mentioned planning problems.

Obstacles modeling have been done by using combinations of simple geometric objects (spheres, cylinders, and plans) which provide an efficient algorithm for collision avoidance.

Path planning algorithm is based on global genetic algorithms optimization techniques, which aim to minimize the sum of the distances between significant points of the robot along the path considering the restrictions to avoid collisions with obstacles. The path is composed of adjacent configurations obtained by an optimization technique using genetic algorithms, seeking to minimize a multi-objective function that involves the distance between significant points of the two adjacent configurations, and the distance from the points of the current configuration to the final one. An evaluation method is designed according to the problem presentation by defining individuals and genetic operators capable of providing efficient solutions to the problem. The result of this algorithm is the shortest path between two configurations given by the user.

Trajectory planning algorithm is also based on genetic algorithms optimization techniques using the direct procedure. The algorithm is similar to the mentioned previously algorithm for path planning problem, but with some differences in the objective function and some details related to the conceptual difference between path and trajectory planning. The objective of this algorithm is to minimize the time required to move the robot from an initial configuration to another final one without colliding with obstacles, taking into consideration the limitation on the actuators. Each trajectory is constructed by means of adjacent configurations obtained through an optimization process using genetic algorithms aims to minimize a function of time required to move the robot between two adjacent configurations, the distance from the points of the current configuration to the final one, and the distance between significant points of the adjacent configurations along the trajectory. The restrictions of this algorithm may be one or a combination of the following: torque, power, and energy limitations. The result of the optimization algorithm is a trajectory with minimum time between two configurations of the robot.

Abstract

The algorithms presented in this thesis have been validated by its use to a significant number of examples. The analysis of the results sheds light on the characteristics and properties of the algorithms used, allowing obtaining the conclusions of the work and focusing on new ways to explore in future work.

RESUMEN

En las últimas décadas, debido a la importancia de sus aplicaciones, se han propuesto muchas investigaciones sobre la planificación de caminos y trayectorias para los manipuladores, algunos de los ámbitos en los que pueden encontrarse ejemplos de aplicación son; la robótica industrial, sistemas autónomos, creación de prototipos virtuales y diseño de fármacos asistido por ordenador. Por otro lado, los algoritmos evolutivos se han aplicado en muchos campos, lo que motiva el interés del autor por investigar sobre su aplicación a la planificación de caminos y trayectorias en robots industriales.

En este trabajo se ha llevado a cabo una búsqueda exhaustiva de la literatura existente relacionada con la tesis, que ha servido para crear una completa base de datos utilizada para realizar un examen detallado de la evolución histórica desde sus orígenes al estado actual de la técnica y las últimas tendencias.

Esta tesis presenta una nueva metodología que utiliza algoritmos genéticos para desarrollar y evaluar técnicas para la planificación de caminos y trayectorias. El conocimiento de problemas específicos y el conocimiento heurístico se incorporan a la codificación, la evaluación y los operadores genéticos del algoritmo.

Esta metodología introduce nuevos enfoques con el objetivo de resolver el problema de la planificación de caminos y la planificación de trayectorias para sistemas robóticos industriales que operan en entornos 3D con obstáculos estáticos, y que ha llevado a la creación de dos algoritmos (de alguna manera similares, con algunas variaciones), que son capaces de resolver los problemas de planificación mencionados.

El modelado de los obstáculos se ha realizado mediante el uso de combinaciones de objetos geométricos simples (esferas, cilindros, y los planos), de modo que se obtiene un algoritmo eficiente para la prevención de colisiones.

El algoritmo de planificación de caminos se basa en técnicas de optimización globales, usando algoritmos genéticos para minimizar una función objetivo considerando restricciones para evitar las colisiones con los obstáculos. El camino está compuesto de configuraciones adyacentes obtenidas mediante una técnica de optimización construida con algoritmos genéticos, buscando minimizar una función multiobjetivo donde intervienen la distancia entre los puntos significativos de las dos configuraciones adyacentes, así como la distancia desde los puntos de la configuración actual a la final. El planteamiento del problema mediante algoritmos genéticos requiere de una modelización acorde al procedimiento, definiendo los individuos y operadores capaces de proporcionar soluciones eficientes para el problema.

El algoritmo de planificación de trayectorias también se basa en técnicas de optimización que usan algoritmos genéticos mediante el procedimiento directo; similares al algoritmo del problema de la planificación de caminos, pero con algunas diferencias en la función objetivo y detalles relacionados con la diferencia conceptual entre la planificación de trayectorias y caminos. El objetivo de este algoritmo es minimizar el tiempo necesario para mover el robot de una configuración inicial a otra final sin colisionar con los obstáculos, considerando los límites de los actuadores. Cada trayectoria esta construida por configuraciones adyacentes obtenidas mediante un proceso de optimización utilizando algoritmos genéticos para minimizar una función del tiempo necesario para mover el robot entre dos configuraciones adyacentes, la distancia desde los puntos de la configuración actual a la final y la distancia entre los puntos significativos de las configuraciones adyacentes a lo largo de la trayectoria. Las restricciones de este

Resumen

algoritmo pueden ser una o una combinación de lo siguiente: los límites de par, potencia y energía. El resultado del algoritmo de optimización es una trayectoria con un tiempo mínimo entre dos configuraciones del robot.

Los algoritmos presentados en esta tesis han sido validados por su uso a un número significativo de ejemplos. El análisis de los resultados arroja luz sobre las características y propiedades de los algoritmos utilizados, que se reflejan en dos grandes capítulos creadas para este propósito, permitiendo obtener las conclusiones del trabajo y orientando sobre nuevas vías a explorar en trabajos futuros.

RESUM

En les últimes dècades, s'han proposat moltes investigacions sobre la planificació de camins i trajectòries per als manipuladors donada la importància de les seues aplicacions, alguns dels àmbits en què poden trobar exemples d'aplicació són: la robòtica industrial, sistemes autònoms, creació de prototips virtuals i disseny de fàrmacs assistit per ordinador. D'altra banda, els algorismes evolutius s'han aplicat en aquesta gran quantitat de camps, el que motiva l'interès de l'autor per investigar sobre la seva aplicació a la planificació de camins i trajectòries en robots industrials.

En aquest treball s'ha dut a terme una recerca exhaustiva de la literatura existent relacionada amb la tesi, que ha servit per a crear una completa base de dades utilitzada per realitzar un examen detallat de l'evolució històrica des dels seus orígens a l'estat actual de la tècnica i les últimes tendències.

Aquesta tesi presenta una nova metodologia que utilitza algorismes genètics per a desenvolupar i avaluar algorismes per a la planificació de camins i trajectòries. El coneixement de problemes específics i el coneixement heurístic s'incorporen a la codificació, l'avaluació i els operadors genètics de l'algorisme genètic.

Aquesta metodologia introdueix nous enfocaments per tal de resoldre el problema de la planificació de camins i la planificació de trajectòries per a sistemes robòtics industrials que operen en entorns 3D amb obstacles estàtics, i que ha portat a la creació de dos algorismes (d'alguna manera similars, amb algunes variacions), que són capaços de resoldre els problemes de planificació esmentats.

El modelat dels obstacles s'ha realitzat mitjançant l'ús de combinacions d'objectes geomètrics simples (esferes, cilindres, i els plànols), de manera que s'obté un algorisme eficient per a la prevenció de col·lisions.

L'algorisme de planificació de camins es basa en tècniques d'optimització globals, usant algorismes genètics per minimitzar la suma de les distàncies entre els punts significatius del robot al llarg del camí considerant restriccions per evitar les col·lisions amb els obstacles. El camí està compost de configuracions adjacents obtingudes mitjançant una tècnica d'optimització construïda amb algorismes genètics, buscant minimitzar una funció multiobjectiu on intervenen la distància entre els punts significatius de les dues configuracions adjacents, així com la distància des dels punts de la configuració actual a la final. El plantejament del problema mitjançant algorismes genètics requereix d'una modelització d'acord al procediment, definint els individus i operadors capaços de proporcionar solucions eficients per al problema. El resultat d'aquest algorisme és el camí més curt entre dues configuracions donades per l'usuari.

L'algorisme de planificació de trajectòries també es basa en tècniques d'optimització que fan servir algorismes genètics mitjançant el procediment directe, similars a l'algorisme del problema de la planificació de camins, però amb algunes diferències en la funció objectiu i detalls relacionats amb la diferència conceptual entre la planificació de trajectòries i camins. L'objectiu d'aquest algorisme és minimitzar el temps necessari per moure el robot d'una configuració inicial a una altra final sense topar amb els obstacles, considerant els límits dels actuadors. Cada trajectòria està construïda per configuracions adjacents obtingudes mitjançant un procés d'optimització utilitzant algorismes genètics per minimitzar una funció del temps necessari per moure el robot entre dues configuracions adjacents, la distància des dels punts de la configuració actual a la final i la distància entre els punts significatius de les configuracions adjacents al llarg de la

Resum

trajectòria. Les restriccions d'aquest algorisme poden ser una o una combinació del següent: els límits de parell, potència i energia. El resultat de l'algorisme d'optimització és una trajectòria amb un temps mínim entre dues configuracions del robot.

Els algorismes presentats en aquesta tesi han estat validats pel seu ús a un nombre significatiu d'exemples. L'anàlisi dels resultats llança llum sobre les característiques i propietats dels algorismes utilitzats, que es reflecteixen en dos grans capítols creats per a aquest propòsit, permetent obtenir les conclusions del treball i orientant sobre noves vies a explorar en treballs futurs.

CONTENTS

NOMENCLATURE	VII
ABBREVIATIONS	XI
CHAPTER 1 INTRODUCTION	1
1.1. MOTIVATION AND DOMAIN OF APPLICATION	1
1.2. PATH PLANNING: STATE OF THE ART	3
1.2.1. Classical Path Planning Approaches	16
1.2.2. Probabilistic Path Planning Approaches:	23
1.3. TRAJECTORY PLANNING: STATE OF THE ART	27
1.4. TRAJECTORY AND PATH PLANNING USING GENETIC ALGORITHM: STATE OF THE ART	38
1.5. OBJECTIVES	45
1.6. ORGANIZATION OF THE THESIS	46
CHAPTER 2 PROBLEM MODELING	47
2.1. ROBOT MODELING	47
2.2. KINEMATIC PROBLEM	49

2.2.1.	Coordinate System	50
2.2.2.	Forward Kinematics	52
2.2.3.	Inverse Kinematics	54
2.3.	THE DYNAMIC MODEL	59
2.3.1.	Inverse Dynamics Problem	61
2.4.	ENVIRONMENT MODELLING	64
2.4.1.	Workspace Modeling	64
2.4.2.	Obstacle Modeling	66
2.5.	COLLISION AVOIDANCE FORMULATION	68
2.5.1.	Spherical Obstacles	69
2.5.2.	Cylispherical Obstacles	72
2.5.3.	Quadri-lateral Plane Obstacles	78
CHAPTER 3	PATH PLANNING	87
3.1.	ADJACENT CONFIGURATIONS FOR PATH PLANNING	89
3.1.1.	Adjacent Configurations Definition	89
3.1.2.	Workspace Discretization	90
3.1.3.	Obtaining The Configuration C^k	91

3.2.	GA PROCEDURE FOR PATH PLANNING	94
3.2.1.	Genetic Algorithms Operators and Parameters	96
3.3.	TRAJECTORY PLANNING: INDIRECT METHOD	101
3.4.	TIME OPTIMIZER	102
3.4.1.	Formulation of Cubic Polynomial Joint Trajectory	102
3.4.2.	Optimization of Cubic Polynomial Joint Trajectory	108
3.5.	APPLICATION EXAMPLES	110
3.5.1.	Path Planning Procedure Examples	110
3.5.2.	Time Optimizer Examples	122
3.6.	DISCUSSION OF RESULTS	126
CHAPTER 4	TRAJECTORY PLANNING	129
4.1.	ADJACENT CONFIGURATIONS FOR TRAJECTORY PLANNING	132
4.1.1.	Adjacent Configurations Formulation	132
4.1.2.	Third-Order Polynomial Trajectory Planning Between Adjacent Configurations	134
4.1.3.	Workspace Discretization	136
4.1.4.	Sequential Quadratic Programming Algorithm	137

4.1.5.	Genetic Algorithm Optimization Procedure	140
4.1.6.	Comparison Results Between SQP and GA	141
4.2.	OBTAINING THE TRAJECTORY	143
4.2.1.	Genetic Algorithm Procedure	144
4.3.	APPLICATION EXAMPLES & RESULTS	147
4.3.1.	Example 1: Comparison Results with Rubio et al. 2009b	148
4.3.2.	Example 2: Indirect and Direct Methods Comparison	155
4.3.3.	Example 3: Industrial Application – Comparison Results	156
4.3.4.	Example 4: Industrial Application – Case With and Without Obstacles	158
4.3.5.	Example 5: Typical Industrial Application	160
4.4.	DISCUSSION OF RESULTS	161
CHAPTER 5	CONCLUSIONS AND FUTURE WORK	163
CHAPTER 6	REFERENCES	167
APPENDIX A	PUMA 560 ROBOTIC MANIPULATOR	189
A.1.	DENAVIT-HARTENBERG PARAMETERS	190
A.2.	DYNAMIC PARAMETERS	191

A.3.	LOCAL POSITIONS OF SIGNIFICANT AND INTERESTING POINTS	192
A.4.	JOINTS LIMITS	193
	LIST OF FIGURES	195
	LIST OF TABLES	199

NOMENCLATURE

$d_i, \theta_i, a_i, \alpha_i$	Denavit-Hartenberg parameters.
q_i	Generalized position of body i .
q_i^j	Generalized position of body i in configuration j .
\dot{q}_i	Generalized velocity of body i .
\ddot{q}_i	Generalized acceleration of body i .
a_j, b_j, c_j, d_j	Coefficients of the interpolation functions
${}^i\ddot{\vec{r}}_{O_i}$	Linear acceleration of the origin of the reference system attached to body i .
${}^i\ddot{\vec{r}}_{G_i}$	Centre of gravity linear acceleration of body i .
${}^k\vec{r}_{O_i, O_j}$	Vector from the origin of the reference system i to the origin of the reference system j , expressed in the reference system k .
${}^i\vec{r}_{O_i, G_j}$	Vector from the origin of the reference system attached to body i to its centre of gravity expressed in the same reference system.
${}^i\mathbf{R}_j$	Rotation matrix between the reference systems i and j .
τ_i	Vector of generalized forces.
n_{pts}	Number of points.
ω_i	Angular velocity of body i .
$\dot{\omega}_i$	Angular acceleration of body i .

${}^i\bar{z}_j$	Unit vector in the z-axis direction of the reference frame attached to body j expressed with respect to the reference frame i .
C^i	i^{th} configuration of the robot
C_s	Configuration space
C_o	Forbidden regions
C_f	Free space
γ_i^j	Significant Point i for the configuration j of the robot
λ_k^j	Interesting Point k for the configuration j of the robot
D	Number less than the size of the smallest obstacle in the workspace
S_i	Spherical Obstacle i .
$Q_{s,i}$	The centre of the spherical obstacle i .
$r_{s,i}$	The radius of the spherical obstacle i .
CL_k	Cylindrical obstacle k .
$Q_{c1,i}$	The centre of the base 1 of the cylindrical obstacle i .
$Q_{c2,i}$	The centre of the base 2 of the cylindrical obstacle i .
$r_{c,i}$	The radius of the cylindrical obstacle i .
PR_l	Quadri-lateral plane l
$V_{P1,l}$	Vertex 1 of the Quadri-lateral plane l
$V_{P2,l}$	Vertex 2 of the Quadri-lateral plane l
$V_{P3,l}$	Vertex 3 of the Quadri-lateral plane l
$e_{p,l}$	Height of the Quadri-lateral plane l

Δ_x	Discretization size of the prism workspace in X-direction
Δ_y	Discretization size of the prism workspace in Y-direction
Δ_z	Discretization size of the prism workspace in Z-direction
Pts_x	Number of discretized stops of the prism workspace in X-direction
Pts_y	Number of discretized stops of the prism workspace in Y-direction
Pts_z	Number of discretized stops of the prism workspace in Z-direction
$O(n)$	Computational complexity.
m_i	The total mass of link i
\dot{v}_C	The acceleration of centre of mass
\dot{v}	The linear acceleration for point P
${}^C I$	The inertia tensor of the link written in a frame, $\{C\}$
${}^{i-1} T_i$	The transformation matrix
${}^i F_i$	Force acting at the centre of mass of the body i
${}^i N_i$	Torque acting at the centre of mass of the body i
n_i	The torque exerted on body i

ABBREVIATIONS

DOF	Degrees Of Freedom.
n -DOF	Number of Degrees Of Freedom
S	Spherical joint type.
SQP	Sequential Quadratic Programming.
NP	Nondeterministic Polynomial
NPC	NP-Complete
C-space	Configuration Space
OBPRM	Obstacle-Based Probabilistic Roadmap Method
RRT	Rapidly exploring Random Tree
RRS	Retrieval RRT Strategy
SVM	Support Vector Machine
HVGV	Hierarchal Generalized Voronoi Graph
PRM	Probabilistic Roadmap Method
RPP	Randomized Path Planner
MTC	Minimum Time Control
GA	Genetic Algorithm
NE	Newton-Euler
RNEA	Recursive NE Algorithm
NAG	Numerical Algorithms Group
SSGA	Steady State Genetic Algorithm

Modified-DH Modified Denavit-Hartenberg

CHAPTER 1

INTRODUCTION

1.1. MOTIVATION AND DOMAIN OF APPLICATION

In the last few decades, the number of robots has grown in many areas. Upon industrial applications, robots also are used in surgery, agriculture, underwater, and for transportation. In industrial applications, they have many purposes like; pick and place operations, assembly tasks, spray-painting, and many other tasks.

In some cases it is required to control and program the robots in real-time. On the contrary, to meet demands on flexibility, quality, and efficiency in industrial systems, off-line programming is required. In off-line programming systems, the programmer uses a three-dimensional computer model of the robot and its work cell, in which the virtual robot can be controlled easily and moved to the desired configurations. When the program is completed, the motion can be verified, simulated, and optimized before its application on the actual robot. Another advantage of the off-line programming is the improved safety for the operator as well as the robot. Despite the fact that, off-line programming improves

efficiency in many aspects, the programming work is still performed manually. When the robot is to be moved from one position to another, it is then necessary to obtain the path that connects these points avoiding collisions. Therefore, the planning of paths or trajectories is one of the most important areas in robotics research.

Path planning and trajectory planning problems are two distinct parts of the robotics that intimately are related. They are considered as a very active research area and there are many algorithms to solve such problems. Actually, a clear difference exists between those algorithms devoted for path planning problem and those devoted for determining the optimal trajectory for robotic systems. The first ones try, essentially, to obtain a sequence (“a path”) of robot configurations (generalized coordinates) between an initial configuration (start) and a final configuration (goal) that fulfils some conditions, mainly, collision avoidance. Whereas, the second ones try to obtain a temporal history of the evolution for the robot joint coordinates, by minimizing aspects, such as; the required time or the energy consumption. Therefore, path planning is a subset of trajectory planning, wherein the dynamics of robot are neglected. In trajectory planning, path planning is searched firstly and then finding an optimal time scaling for the path subject to the actuator limits; such approach known as decoupled (indirect) approach. In the other hand, the direct approach of trajectory planning, the search takes place in the system’s state space.

In this thesis, both, path planning and trajectory planning are presented as two distinct fields, and each one is going to be reviewed separately.

1.2. PATH PLANNING: STATE OF THE ART

Path planning has a very important role for getting to a desired goal in mobile robots. Path planning, as mentioned earlier, tries to determine a sequence of robot configurations between an initial configuration and a goal configuration under certain restrictions, such as; collision avoidance, which is can be easily stated “How to get from here to there?”.

The basic path-planning problem involves computing a collision-free path between an initial configuration of the robot and a final one in a static environment of known obstacles, and that the planned motion is consistent with the kinematic constraints of the robot. In this case, the constraints on the solution path arise from the geometry of both the obstacles and the robot.

Path planning has important applications in many areas, for example, industrial robotics, autonomous systems, assembly planning and virtual prototyping, Chang and Li 1995, computer graphics simulations, Kuffner and Latombe 2000, and computer-aided drug design, Finn et al. 1997.

According to Hwang and Ahuja 1992a path planning algorithms can be classified into two aspects: the scope (global or local) and the completeness. Global algorithms assume that the robot’s environment is completely known. Global algorithms take into account all the information in the environment, and they plan a path from the initial to the goal configuration. Therefore, their strength is global path planning, but they are not appropriate for fast obstacle avoidance. On the other hand, local algorithms use only a small fraction of the world model to generate robot control. They are used when the start and goal configuration of the robot are close. However, the key advantage of local techniques over global ones lies in their low computational complexity, which is particularly important when

the world model is updated frequently based on sensor information. In the other hand, with respect to the completeness, Hwang and Ahuja classified it into three types. Exact (or complete) algorithms either find a solution or prove that there is no solution. They are usually computationally expensive. Most complete algorithms, however, are applicable in low-dimensional configuration spaces problems. Resolution complete algorithms discretize some continuous quantities such as object dimensions or configuration parameters, but become exact in the limit as the discretization approaches a continuum. For probabilistically complete algorithms, the probability of finding a solution can be made to approach one if the problem is indeed solvable. Most such algorithms use a randomized search procedure, which is guaranteed to find a solution if it is allowed to run long enough. Finally, the heuristic algorithms are often non-complete as they may fail to find a solution even when one exists. They are aimed to generating a solution in a short time. Exact algorithms determine the complexities of the problems, while heuristic algorithms are important in engineering applications.

For complexity analysis, some definitions should be cleared. Cormen et al. 2001 classify the problems to three classes: P , NP , and NPC . The problem is said to be in P , if there is a polynomial time algorithm to solve it. If there is a polynomial time algorithm to verify a solution to the problem (thus $P \subseteq NP$), the problem is said to be in NP (Nondeterministic Polynomial). This means that the problem in NP needs a very long computation time to solve if the problem size is large. A problem is NP -hard if it is at least as difficult as any NP problem. If the problem is NP and NP -hard, it is said to be NPC (NP -Complete). If the problem requires a space polynomial in the problem size, it is considered in $PSPACE$. The same definitions apply to $PSPACE$ -hard and $PSPACE$ -complete. $PSPACE$ hardness results have been demonstrated for various special cases of motion planning. Reif 1979 presented the first theoretical investigation of the inherent computational complexity of the path planning problem, showing that path

planning for a 3-D linkage made of polyhedral links among fixed obstacles is *PSPACE-hard*. A few years later, Hopcroft, Schwartz et al. 1984 proved that motion planning for multiple independent rectangular boxes sliding inside a rectangular box is *PSPACE-hard*. Hopcroft, Joseph et al. 1984 improved that the movers' problem for two-dimensional linkages is *PSPACE-hard*. One year later, Reif and Sharir 1985 proved that the dynamic movement in the case of bounded velocity is *PSPACE-hard*, even in the case where the moving body is a disc moving in three-dimension. After that, Reif and Storer 1988 and Reif and Storer 1994 presented an algorithm for finding the shortest path between points in the Euclidean plane with polygonal obstacles. Sun and Reif 2003 introduce an empirical method, called discretization method, that improve the results of the weighted region optimal path problem, by placing discretization points only in areas that may contain optimal paths.

Path planning for robots and manipulators is a problem for which new contributions are still provided almost every day, since Nilsson 1969 introduced the visibility graph method (combined with A^* search algorithm, Hart et al. 1968) to find the shortest collision-free path for his mobile robot system (Shakey) represented by a point amidst polygonal obstacles.

Liebermann and Wesley 1977 and Lozano-Pérez 1976 presented the first attempts to build integrated systems for automatically programming robot arms. The input of these systems was the description of a mechanical assembly, in the form of a set of geometrical models of the individual parts and goal assembly relations among the parts. The task of the systems was to generate the robot programs automatically for assembling the parts. Although these systems were never fully implemented, they have contributed in emphasizing the importance of geometrical reasoning in robot planning and in pointing out key motion planning problems in the context of mechanical assembly.

Udupa 1977 proposed the idea of growing the obstacle and shrinking a robot to a point for planning collision-free paths for computer-controlled manipulators. Moravec 1980 bounded all the obstacles and the moving robot by circles. The moving circle is shrunk to its center and the obstacle circles are inversely expanded. The finding path problem reduced to find the path for the center of moving circle to stay outside of the grown circles. In this case, the rotation was ignored. Brooks and Lozano-Pérez 1983 introduced a subdivision algorithm for computing with the curve surfaces of the grown obstacles. That algorithm had the ability to find hard paths for 2-D moving robots. Moreover, it could be directly applied to configuration spaces for three dimensional polyhedral whose orientation is fixed.

Lozano-Pérez and Wesley 1979 exploited the Udupa 1977 idea in a more general and systematic manner, and proposed the first two-dimensional path planning algorithm for polygonal and polyhedral robots moving among polygonal and polyhedral obstacles. In addition, they introduced the concept of configuration space (C_s), which influenced motion planning more than any other idea. In C_s , the obstacles in the workspace are mapped as forbidden regions (C_o), and the complement of the C_o constitutes the free space (C_f). Path planning for a robot with n degrees-of-freedom (DOF) can thus be converted to the problem of planning a path for a particle in an n -dimensional C_s . Many methods of many authors have been proposed for the construction of the configuration space C_s . Lozano-Pérez 1987 considered the case where both the robot and the obstacles were convex polygons or polyhedral, and the C_o boundary for an n -DOF manipulator was approximated by sets of $(n-1)$ -dimensional slices recursively built up from one-dimensional slices. Maciejewski and Fox 1993 studied the analytical description of the boundaries of C_o and derived the connectivity of C_s for revolute manipulators. Zhao et al. 1995 developed an analytical representation of C_o using a set of parametric equations resulted from mapping the boundaries of

the obstacles from workspace into the C_s through using the inverse pseudo kinematics. Recently, Wu et al. 2007 studied a new two-phase approach for the construction of C_s . The approach based upon pre-computing the global topology of a robot's free space, and consisted of an offline phase and an online phase. In the offline phase, a C_o database (COD) for a given robot was developed in which the C_o maps were stored and indexed by the cells of the workspace; in the online phase when the same robot is operated in a real environment, those maps whose indices match the real obstacle cells were identified and then extracted from the COD. This approach is a generic one and can be applied to manipulators with arbitrary kinematic structures and geometries. The authors used a series of simulation cases involving a 3-DOF manipulator and a 5-DOF manipulator to demonstrate the performance of the proposed scheme.

Moreover, Lozano-Pérez presented the principle of the approximate cell decomposition approach, see Lozano-Pérez 1981, 1983. Chatila 1982 was the first to base his planner on an exact decomposition into convex cells to solve the planning problem with incomplete knowledge for a mobile robot represented as a point in a two-dimensional workspace. The decomposition was periodically updated in order to include new environmental data. Gouzènes 1984 introduced the first implemented approximate cell decomposition method for the motion planning of arm robot with revolute joints.

In the solution of several path-planning problems, the notion of Voronoi diagram has proved to be a useful tool. Ahmed 1997 said that the use of Voronoi diagram for motion planning first appeared in the doctoral research work of Rowat 1979 who uses it as a heuristic for a digitized space. O'Dunlaing and Yap 1982 introduced retraction as a new theoretical approach for path planning. His method is based on the generalized Voronoi diagram, which is the locus of points equidistant to two or more obstacles, to motion planning for a disk in the plane.

His method requires full knowledge of the world's geometry prior to the planning event. Brooks 1983a approximated generalized Voronoi graphs with generalized cones through in freeway method in order to find a path for mobile robots. In the same year also, Brooks 1983b used the cones to find quick paths for the Puma arm. A definition and new theoretical results are presented one year later by Donald 1984 for a six-dimensional C-space extension of the generalized Voronoi diagram, named *C-Voronoi* diagram. He described the first known implementation of a complete algorithm for six degrees of freedom Mover's problem by transforming it into a point navigation problem in a six-dimensional configuration space. Based on part of Donald's algorithm, Lengyel et al. 1990 developed a fast path planning based entirely on complete and provably-good approximation algorithms. The planner can handle any polyhedral geometry of robot and obstacles.

Schwartz and Sharir presented a series of papers called the Piano Movers' Problem, representing the first complete path planning approach based on an algebraic decomposition of the robot's configuration space known as Collins decomposition. In the first one of the series, Schwartz and Sharir 1983a introduced the first algorithm polynomial in the number of obstacles in two-dimensions. He gave a topological analysis of the space of positions of a polygon moving in the plane in the presence of polygonal obstacles. In the second paper, Schwartz and Sharir 1983b used manifold and reduced the motion planning problem to the problem of finding the connected components of an algebraic manifold. This algorithm takes time doubly exponential in the degrees of freedom. A few years later, this result was improved by Canny 1988 to a single exponential time. In their next paper Schwartz and Sharir 1983c, they proposed algorithms for solving the case of two-dimensional disks moving inside a polygon with avoiding to collide with the polygon edges and with each other. This algorithm is exponential in the number of moving disks. After that, Spirakis and Yap 1984 proved the strong *NP*-

hardness of moving many disks. Later, Sharir and Ariel-Sheffi 1984 addressed various special problems involving arbitrarily many degrees of freedom which have relatively simple solutions by the techniques of determining the non-critical regions and using a connectivity graph. Finally, Schwartz and Sharir 1984 studied the motion of a rod among polyhedral obstacles in three-dimension. Kedem and Sharir 1985, 1988 presented an exact and efficient algorithm for polygonal robot moving among polygonal obstacles.

One of the most general and simple ways to arrange the path planning problem is based on the utilization of Artificial Potential Fields (APF). This concept was pioneered by Khatib 1986. He proposed this method for the real-time collision avoidance in a continuous space. A drawback to this approach is that it is known to suffer from local minima effects when the net force sums to zero in certain portions of the search space. A year later, Koditschek 1987 redefined the artificial potential field function in a way that does not contain a local-minimum, which known as navigation function, and Rimon and Koditschek 1988 extended the last one to n-dimensional Euclidean space for a point robot moving among disjoint spheres. Hereafter, many authors such as Khosla and Volpe 1988 and Volpe and Khosla 1990, directed their efforts to finding an obstacle associated potential function based on superquadrics to counter the local-minimum problem with better behavior enabling the robotic system to both avoid and smoothly approach. As an alternative to the potential field method, Faverjon and Tournassoud 1987 introduced a local approach, named, the constraints method to plan the motion of high degrees of freedom manipulators, which separate the description of the task from constraints of anti-collision. The same authors, Faverjon and Tournassoud 1988 presented a learning scheme to avoid falling into the local-minimum of the potential field. Barraquand and Latombe 1989 proposed the randomized potential field planner (RPP) for generating paths with local-minimum-free for robots with high degrees of freedom. Connolly et al. 1990

introduced the idea of generating functions that satisfy Laplace's equation as a way to build a local-minimum-free navigational potential field. Latombe 1991 expanded upon the detail of the RPP approach that proposed by Barraquand and Latombe 1989. Latombe explained the motion planning concepts in his book and provided a comprehensive description of the subject and fundamental techniques. Later, Kim and Khosla 1992 proposed an artificial potential function approach to obstacle avoidance based on the panel method. Hwang and Ahuja 1992b constructed a potential function defined in terms of the boundary equations of polyhedral obstacles to develop a path planner compromise between the exact and heuristic algorithms. They extracted firstly the topological structure of the free space in the form of the minimum potential valleys. Then, the potential field is used to derive the most efficient, collision-free path corresponding to a given topological path. One year later, Zelinsky and Yuta 1993 presented a local obstacle avoidance scheme called "reactive planning" based on "path transform" which was first developed by Zelinsky 1991. The path transform can be regarded as numeric potential field path planners without suffering the local-minimum problem. The path transform is considered as an expansion of the distance transforms which was first presented by Jarvis and Byrne 1986. Chuang 1998 suggested an analytic potential field function for three-dimensional workspace to solve path-planning problem with obstacle avoidance.

Faverjon and Tournassoud 1987 were first introduced a subgoal network method. This algorithm divides the C-space into cells and assigns each cell a probability that a local algorithm would succeed in that cell. Initially, the probabilities of the cells are equal and then and they are updated using a local algorithm. A sequence of regions with high probabilities will be searched by A^* algorithm, then the potential field applied to that sequence of cells. Glavina 1990 proposed an algorithm to solve the find-path problem by combining a goal-directed straight-and-slide search and a randomize generation of subgoals. Chen

and Hwang 1992, 1998 developed a new search strategy called SANDROS (Selective And Non-uniformly Delayed Refinement Subgoals). At first, distance computations are performed to determine whether a given point is in free space C_f . Then, a two-level hierarchical planning method is used to reduce memory requirements. This algorithm builds a sparse network of robot subgoals with the use of a simple and a computationally expensive planner. This algorithm has been implemented and tested for planning paths for Puma robot. An efficient path planning algorithm for general 6 degrees of freedom robots is presented by Isto 1996. The path planner is based on multiheuristic A^* search algorithm with dynamic subgoal generation for rapid escaping from deep local-minimum wells. One year later, Isto 1997 developed an algorithm that combines a multiheuristic local search algorithm with a subgoal graph based guidance. Moreover, the algorithm can adjust the balance between local and global planning.

Lozano-Pérez 1987 introduced the first resolution complete planner for general manipulators. Lozano-Pérez et al. 1987 described a new integrated robot system, called Handey. Handey used a simplify version of the path planner described in Lozano-Pérez 1986, 1987. This system is capable to plan the motions of a manipulator robot for constructing simple assemblies made of polyhedral objects, and to execute the plans, assuming no uncertainty. Hwang and chen 1995 proposed a complete path planner based on a hierarchical and multi-resolution search strategy based on the SANDROS search strategy developed by Chen and Hwang 1992. In this planner the lowest possible resolution has to be defined in advance and does not adapt to the particular workspace.

Valero 1990 presented a collision-free path-planning algorithm for a plane manipulator with three degrees of freedom moving among polyhedral obstacles. The manipulator consists of three rigid bodies connected by revolute joints. Firstly, he generated a space of robot configurations, and then searched for a

sequence of configurations with minimum distance to obtain a path between the initial and final configurations of the robot. Valero et al. 1997, Valero et al. 2000 proposed a technique for collision-free path planning as an optimization problem in complex environments based on the concept of adjacent configurations.

Barraquand and Latombe 1990, 1991 developed the Randomized Path Planner (RPP) to solve path-planning problem in high-dimensional configuration space. They had applied the RPP a general potential field method that uses random motions for escaping spurious local-minimum. Additionally, RPP has been embedded in a larger manipulation planner to automatically animate scenes involving human figures modeled with 62 degrees of freedom, Koga et al. 1994. Many years later, Caselli et al. 2001 introduced RPP driven by potential field as a technique for solving path planning problem for 9 and 11 degrees of freedom robots. He presented a simple yet effective heuristics for escaping local minima, with the goal of improving overall planning performance.

The planner implemented by Kondo 1991 found paths for six degrees of freedom manipulators in three-dimensional space using heuristic search technique. This algorithm is fast because it minimizes the number of collision detection computations by limiting the search in the explored parts of the configuration space C_s .

Overmars 1992 presented a new technique uses a learning approach for path planning. He combined the simple potential fields with roadmap method, constructing a random network of possible motions. A disadvantage of that method is that it is uncompleted. Kavraki and Latombe 1994a, 1994b introduced another approach to path planning for many degrees of freedom robots moving in static environments. The algorithm consisted of preprocessing; which is done once for a given environment, generated a network of randomly, but probably selected, collision-free configurations. After that, the planning stage, which connected any

given initial and final configurations of the robot to two nodes of the network, then computed a path through the network between these two nodes. Independently Overmars and Svestka 1994 proposed a probabilistic algorithm for the learning path planning problem by combining a global roadmap approach with a local planner. A common paper from the two groups was published in Kavraki et al. 1996 combined the ideas developed by the two groups above, resulting an even more powerful planner for high degrees of freedom robots. This algorithm uses random sampling to construct a roadmap of the configuration space and tries to find a path between any two input configurations by connecting them to the roadmap. The main difficulty with a uniform random sampling of C -space is find connections through some "critical" regions of free space C_f . This difficulty is referred to as the narrow passage problem, and is common to randomized algorithms. Hsu et al. 1998 accepted samples that are not in free space, but for which the penetration distance of the robot into the obstacles is small. Then the colliding nodes are retracted to C_f by local re-sampling. Kavraki et al. 1998a provided an analysis of a recent path planning method, which uses probabilistic roadmaps. Then they studied the dependence of the failure probability to connect these configurations on: the length of the path, the distance function of the path from the obstacles, and the number of nodes of the probabilistic roadmap constructed. The probability of placing random configurations inside the passage and connecting them by straight-line paths is small. Kavraki and Latombe 1998 proposed a randomized method, which has been successfully applied for solving path-planning problem for robots with 3 to 6 degrees of freedom operating in known static environments. Boor et al. 1999 introduced a Gaussian non-uniform sampling strategy in order to create a higher density of nodes near the boundary of the C_f . Another approach, proposed by Wilmarth et al. 1999 sample the generalized Voronoi diagram of C_f , by retracting randomly sampled configurations using approximate values of clearance and penetration depth. Siméon et al. 2000 suggested a PRM that computes visibility roadmaps, which defined with two

classes of nodes: the guards and connectors. Collision-free samples are kept as a new guard node when they cannot be connected to the current roadmap or as a new connector if they improve the connectivity of the roadmap.

A hybrid approach was considered by Caselli and Reggiani 2000, which utilized a potential function (similar to RPP) on queries, but also saved information from past attempts in a graph to aid future queries in the same environment. Comparing with RPP, the performance advantage exhibited by ERPP is strictly due to the learning component of the experience-based planner.

Wu 1996 developed path-planning algorithm, namely, the obstacle-based probabilistic roadmap method (OBPRM) for robots with many degrees of freedom. The main novelty of his approach was a new method for generating roadmap candidate points randomly distributed on or near the surface of each C_o . Amato et al. 1998 described and evaluated several strategies for node generation and proposed a multi-stage connection strategy for OBPRM in cluttered three-dimensional workspaces.

The attractiveness of randomized path planners stems from their applicability to virtually any type of robots. Barraquand et al. 1997 introduced a unifying view of these planners. An estimate is given for the probability that the roadmap planner can find a path between two given configurations, assuming that a path of certain clearance exists. In addition, they have analyzed the probabilistic completeness of variants of the roadmap planners under the visibility volume and the path clearance assumptions. In each case, they have established a relation between the probability that the planner finds a path, when one exists, and its running time.

Other method, described by Hsu et al. 1997, build two trees rooted at the initial and goal configurations respectively. The trees are expanded by generating

new nodes randomly near the two trees, and connecting them to the trees by a local planner. LaValle 1998 proposed a new probabilistic technique called "Rapidly exploring Random Tree (RRT)". RRTs are suited particularly for path planning problems that involve algebraic constraints (arising from obstacles) and differential constraints (arising from nonholonomic and dynamics). LaValle and Kuffner 1999 introduced an RRT-based approach to path planning that generated and connected two RRTs in a state space, which generalizes configuration space. Recently, Oh et al. 2007 presented an algorithm named Retrieval RRT Strategy (RRS) which extended the RRT framework to deal with change of the task environments. This algorithm combines a support vector machine (SVM) and RRT and plans the robot path in the presence of the change of the surrounding environments. They applied the algorithm on robot manipulator with 6 degrees of freedom.

Henrich et al. 1998 showed a heuristic hierarchical search procedure for an industrial robot with six degrees of freedom in an on-line provided three-dimensional workspace to solve the path-planning problem. This search procedure based on the combination of multiple neighboring hypercubes resulting in step-sizes in free areas, while maintaining small steps in the vicinity of obstacles.

Helguera and Zegloul 2000 addressed the collision-free path-planning problem for manipulators based on a local approach. The task was defined as a combination of two displacements. The first one brings the robot closer to the goal configuration and the second one enables the robot to avoid the local minima. However, a zigzagging phenomenon appears in some heavy cluttered environments. To avoid this situation, a graph based on the local geometry of the environment is constructed and an A^* search is performed in order to find a new deadlock free position. Tests and heavy cluttered environments were successfully performed.

Rubio 2006 introduced in his thesis a sequential and simultaneous algorithms based on adjacent configurations to obtain a sequence of path configurations. Rubio et al. 2009a presented an approach in which the search of the path is made in the state space of the robotic system, and it makes use of the information generated about the characteristics of the process, introducing graph techniques for branching. The method poses an optimization problem that aims at minimizing the distance traveled by the significant points of the robot.

1.2.1. Classical Path Planning Approaches

There are a large number of methods for solving the basic path-planning problem. Some are applicable to a wide variety of path planning problems, whereas others have a limited applicability. The methods that will be treated in this section are based on few different general approaches: roadmap, cell decomposition, and potential field. Roadmap and cell decomposition approaches differ in the connectivity graphs constructed and their representations, while the potential field approach does not build connectivity graph explicitly. Instead, it constructs a potential function for which the gradient guides the robot to the goal. Roadmap and cell decomposition methods are global methods, while the potential field approach is local one.

1.2.1.1. Roadmap Approach

The roadmap approach consists of capturing the connectivity of the robot's free space C_f in a network of one-dimensional curves. Once a roadmap R has been constructed, it is used as a set of standardized paths. Path planning is thus reduced to connect the initial and goal configurations to points in R and searching R for a path between these points, Latombe 1991. The constructed path, if any, is the

concatenation of three sub-paths: a sub-path connecting the initial configuration to the roadmap, a sub-path contained in the roadmap, and a sub-path connecting the roadmap to the goal configuration. However, a good roadmap has the property that there is a collision-free path in the space between two configurations if and only if there is a collision-free path using only the curves represented in R . Algorithms that produce such roadmaps are clearly complete "exact".

A various sorts of roadmaps method has been produced based on different principles; visibility graphs, Voronoi diagrams, freeway nets, silhouettes. All of these roadmaps have a corresponding graph representation.

- Visibility Graph Method

This method is one of the earliest path-planning methods, Nilsson 1969. It can produce shortest paths in two-dimensional configuration spaces with polygonal obstacles. The principle of the visibility graph method is to construct a semi-free path as a simple polygonal line connecting the initial configuration C^i to the goal configuration C^f through vertices of C_o , Latombe 1991. The visibility graph is the undirected graph G . The nodes of G are C^i , C^f , and the vertices of C_o . The links of G are line segments, which connecting two nodes without intersecting the C_o region, see Figure (1.1). Once such G obtained, the shortest path can be searched using algorithms such as A^* algorithm or Dijkstra's algorithm, Dijkstra 1959. The time complexity of this algorithm is $O(n^3)$, where n is the total number of vertices of C_o , Lozano-Pérez and Wesley 1979. Later, more efficient algorithms have been proposed with time complexity $O(n^3)$, e.g. see Welzl 1985, Asano et al. 1986. Ghosh and Mount 1987 gave an output sensitive algorithm that takes $O(k + n \log n)$ time, where k is the number of edges in visibility graph. However, visibility graph produces paths that graze the obstacles and thus bring the robot dangerously close to the obstacles, which is undesirable in practice. For depth knowledge leaders should refer to Latombe 1991, Choset et al. 2005.

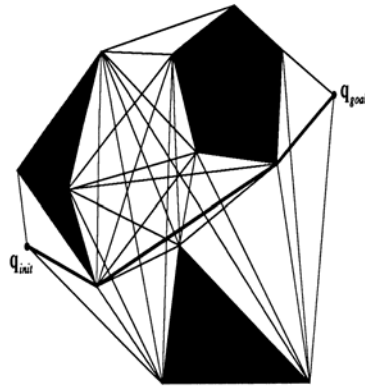


Figure 1.1: Visibility Graph Example, Latombe 1991.

- Voronoi Diagrams

As mentioned before, O'Dunlaing and Yap 1982 introduced retraction as a new theoretical approach for path-planning. This method consists of defining a continuous function of C_f onto a one-dimensional subset of itself, the roadmap, such that the restriction of this function to this subset is the identity map. In a three-dimensional C_f is retracted firstly onto a two-dimensional variant of the Voronoi diagram. In a two-dimensional configuration space, C_f is typically retracted on its Voronoi diagram. This diagram is the set of all the free configurations whose minimal distance to the C_o region, see Figure (1.2). Choset and Burdick 1996 described the Hierarchical Generalized Voronoi Graph (HGVG), which can be applied to higher dimensional workspaces. In Voronoi diagram, the robot stays as far away as possible from the obstacles, which is an advantage over the visibility graph approach. Both algorithms are complete for two-dimensional polygonal configuration spaces, González-Baños et al. 2006. For more details, leaders should back to Latombe 1991, Choset et al. 2005.

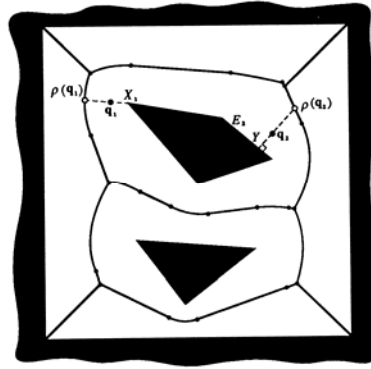


Figure 1.2: Voronoi Diagram Example, Latombe 1991.

- Freeway Method

The freeway was suggested by Brooks 1983a as a method of path planning for manipulators with 5 or 6 DOF. His algorithm applies to a polygonal robot translating and rotating among polygonal obstacles. The algorithm finds obstacles that face each other and generates a freeway to passing between them. This path segment is a generalized cylinder. This freeway may be described as overlapping generalized cones; it is essentially composed of straight lines with left and right free-space width functions, which could easily be inverted. A generalized cone is obtained by sweeping a two-dimensional cross section along a curve in space, called a spine, and deforming it according to a sweeping rule.

- Silhouette Method

It is the principle of a general roadmap method developed by Canny 1988. The Silhouette algorithm has many positive aspects; it is complete and it is not restricted to systems with few degrees of freedom, McHenry 1998. This method solves the basic motion-planning problem in time singly exponential in the dimension of the configuration space. Moreover, it supposes only that the obstacles are described as a semi-algebraic set. Roughly, it consists of constructing the silhouette of the robot's free space when it is viewed from a point at infinity,

and adding some curve segments linking critical points of the silhouette to other curve segments of the silhouette. The silhouette and the linking curves form the roadmap that is subsequently searched for a path.

1.2.1.2. Cell Decomposition

Cell decomposition methods would be the motion planning methods that have been the most extensively studied so far. They consist of decomposing the robot's free space into simple regions, called cells, such that a path between any two configurations in a cell can be easily generated. A non-directed connectivity graph representing the adjacency relation between the cells is then constructed and searched. Its nodes are the cells extracted from the free space and two nodes are connected by a link if and only if the two corresponding cells are adjacent. The outcome of the search is a sequence of cells called a channel. A continuous free path can be computed from this sequence.

Cell decomposition methods can be categorized into exact and approximate methods:

- Exact cell decomposition methods decompose the free space into cells whose union is exactly the free space. Many exact approaches have been developed for low dimensional workspace and with polygonal representations of the robot and obstacles, see Figure (1.3). Schwartz and Sharir 1983a described exact cell methods for decomposing the free space of a robot modeled as a polygon. Avnaim et al. 1988 developed a practical method where only the boundary of the free space is decomposed. Barbehenn and Hutchinson 1995 adopted a critical curve based exact cell decomposition of Schwartz and Sharir 1983a as their basic representation and developed the only truly incremental path planning system. Sleumer

and Tschichold-Gürman 1999 introduced a method for generating a map consisting of connectivity graph and information about the walls of a building that represent the environment of a mobile robot.

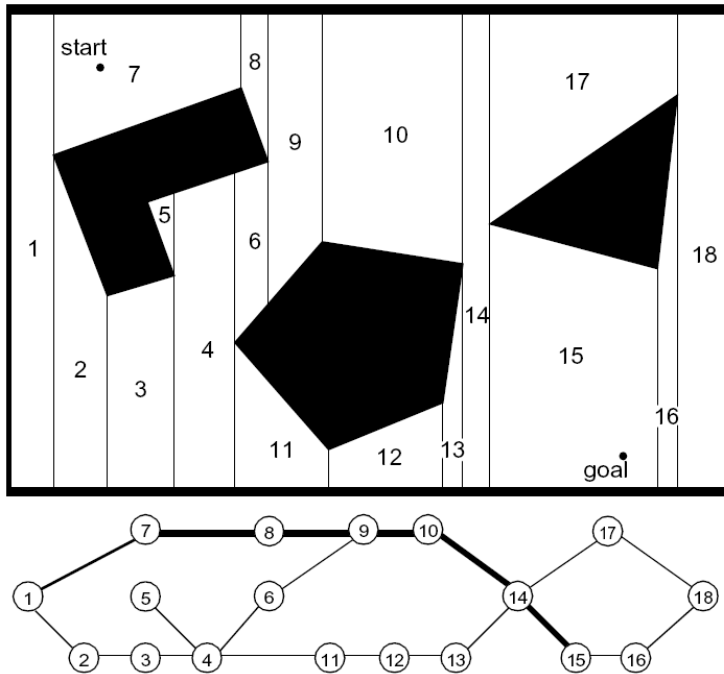


Figure 1.3: Exact Cell Decomposition, Latombe 1991.

- Approximate cell decomposition methods produce cells of predefined shape whose union is strictly included in the free space. The boundary of a cell does not characterize a discontinuity of some sort and has no physical meaning. Approximate cell decomposition approach introduced by Lozano-Pérez 1981, he used a single simple shape for all cells. Brooks and Lozano-Pérez 1983 were the first introduced hierarchal approximate cell decomposition. Furthermore, they divided the configuration space into rectangloids cells with edges parallel to the axis of the space. Cells are classified as empty or full depending on whether they lie entirely outside or inside the obstacles. If there are interior points both inside and outside

of configuration obstacles, they are labeled mixed, for more knowledge leader can refer also to Donald 1984 and Zhu and Latombe 1990. In addition, Latombe 1991 stated that the approximate cell decomposition methods are resolution complete; they can find a path if one exists provided the resolution parameters are selected small enough, whereas exact cell decomposition methods are complete.

1.2.1.3. Potential Field

The Artificial Potential Fields general heuristic approach offers a metaphor based on the physical phenomenon of potential fields. The potential field metaphor has been employed throughout the field of artificial intelligence as a problem solving approach, enjoying particular success in neural networks.

The metaphor suggests that if a problem can be modeled by a function that assigns a value to each state configuration (position) in a continuous state space based on its usefulness, then the optimally useful state configuration can be found by minimizing the value of the function.

Potential field was originally developed by Khatib 1986 as an on-line collision avoidance approach, applicable when the robot does not have a prior model of the obstacles, but senses them during motion execution. The idea underlying potential field can be combined with graph searching techniques.

In this method, the robot represented as a point in configuration space; is a particle moving under the influence of an artificial potential produced by the goal configuration and the C-obstacles. If the robot is not a point, the total potential on the robot is computed by adding the potential values on a set of points sampled from the surface of the robot. Typically, the C-obstacles potential constructed

firstly, producing a repulsive force (has a high value on the obstacles and decreases monotonically as the distance from the obstacle increases) which pushes the robot away from them. While, the goal configuration potential generates an attractive force which pulls the robot toward the goal. The negated gradient of the total potential is treated as an artificial force applied to the robot. At every configuration, the direction of this force is considered the most promising direction of motion. The benefit of this method of being fast, but is incomplete because of the presence of local-minimum, which occurred when the attractive and repulsive forces are equals. As mentioned before, many authors designed different potential functions to lower the number and depth of the local-minimum, e.g., Chuang and Ahuja 1991 introduced the Newtonian potential function to plane a safe and smooth path with local-minimum-free of an object by minimizing the potential function locally for obstacle avoidance. In this algorithm, a global planner identifies narrow bottlenecks in the free space by computing minimum-distance links between obstacles. A collision-free path in each of these regions is computed using the potential field. These paths are connected to yield a solution. For a survey of related researches please see also Latombe 1991, Hwang and Ahuja 1992a.

1.2.2. Probabilistic Path Planning Approaches:

For high-dimensional path planning problems, it is computationally too expensive to calculate an explicit representation of the configuration space. Probabilistic path planning techniques have achieved substantial attention throughout the last decade, as they are capable of solving high-dimensional problems in acceptable execution times. As no explicit representation of configuration space exists, probabilistic methods invoke a binary collision checker to test whether a specific configuration is feasible. The three methods that attracted

most attention during the last years are Randomized Path Planner (RPP), Probabilistic Roadmap Method (PRM) and Rapidly-exploring Random Trees (RRT). All are probabilistically complete.

1.2.2.1. Randomized Path Planner (RPP)

As mentioned above, Randomized Path Planner (RPP) developed by Barraquand and Latombe 1990, 1991 considered as one of the first randomized path planning technique, that combines gradient descent on the potential with a random motion to escape local minimum in a potential field. The planner is probabilistically resolution complete, this means that the probability of finding a path (if there exists one) approaches 1.0 if the algorithm running time is not limited, Barraquand et al. 1992, Lamiriaux and Laumond 1996. RPP leaves the start configuration with gradient descent, and if it terminates at a spurious local minima rather than the intended goal configuration, a random walk of some length is started from the local minimum. Once a lower potential value is found or the length is attained, a new gradient descent towards the goal is attempted. If no lower potential can be found after a given number of descent and random walk iterations, a backtracking move to some previous configuration on a random walk segment of the current solution candidate is executed. The process is iterated from that configuration. RPP does not require any particular type of potential or any potential at all, but it can be guided by the distance to goal if the distance metric is defined to be infinite at configurations belonging to the non-free space. An analysis of this (RPP) planner is initiated by Lamiriaux and Laumond 1996.

1.2.2.2. The probabilistic Roadmap Method (RPM)

Probabilistic roadmap method consists of sampling the configuration space at random and connecting the samples in free space by free-collision local paths (usually straight paths), Kavraki et al. 1996. Unlike the roadmap method, the nodes of the PRM are free configurations, sampled randomly under a suitable probability distribution. PRM consist of two phases: a learning phase and a query phase. In the learning phase (also called construction phase or pre-processing phase in the literature), a roadmap is built by randomly sampling the configuration space. Those samples that correspond to collision-free configurations form the vertices of the roadmap. Neighboring vertices are then connected by edges if all states along these edges also are collision-free. In the query phase, the initial and the goal state are connected to two nodes in the random network, with paths that are feasible for the robot. Then it is searched for a sequence of path connecting these nodes. Concatenation of the successive path segments transforms this sequence, if one has been found, into a feasible path for the robot. Any standard smoothing algorithm can be used to improve the path, Kavraki 1995. Experiments with PRM planners have been quiet successful, showing that they are both fast and reliable with many degrees of freedom robots, Latombe 1999. In addition, it can handle high-dimensional configuration spaces efficiently. Path non-existence cannot be proven using PRM, which considered a weaker completeness result: if a path exists then the learning phase of PRM will eventually compute a roadmap that finds it, Kavraki et al. 1996, Kavraki et al. 1998a, Kavraki et al. 1998b.

Bohlin and Kavraki 2000 introduced a single query variant called Lazy PRM. In this approach, the roadmap validation is postponed. The roadmap is built not in the collision-free configuration space C_f , but in the whole configuration space C_s . First, after a path has been found in the query phase, this path is checked whether it is feasible or not. Thereby, the number of collision checks needed is

reduced drastically, making Lazy PRM favorable especially if collision checking is very costly. If no path could be found, the roadmap has to be extended.

For more information about probabilistic roadmap leaders should back to Kavraki and Latombe 1994a, Amato and Wu 1996, Barraquand et al. 1997, Bohlin and Kavraki 2000, Choset et al. 2005.

1.2.2.3. Obstacle-Based Probabilistic Roadmap Method (OBPRM)

Obstacle-Based Probabilistic Roadmap Method (OBPRM) firstly developed by Amato and Wu 1996, Wu 1996. The general approach of this algorithm follows traditional roadmap methods: during pre-processing a graph, or roadmap, is built in C_s . Planning consist of connecting the initial and goal configurations to the roadmap, and then finding a path in the roadmap between these two connection points. This approach generates candidate points randomly distributed in the surface of C_o . High quality roadmaps can be obtained using this approach even when the configuration space is crowded.

1.2.2.4. Rapidly-exploring Random Trees (RRT)

Another probabilistic algorithm is Rapidly-exploring Random Trees (RRT), which developed as a novelty by LaValle 1998. RRT is a data structure and algorithm that is designed for efficiently searching non-convex high-dimensional spaces. RRTs are constructed incrementally in a way that quickly reduces the expected distance of a randomly chosen point to the tree. RRTs are particularly suited for path planning problems that involve obstacles and differential constraints (nonholonomic or kinodynamic). RRTs can be considered as a technique for generating open-loop trajectories for nonlinear systems with state constraints. Usually, an RRT alone is insufficient to solve a planning problem.

Thus, it can be considered as a component that can be incorporated into the development of a variety of different planning algorithms.

1.3. TRAJECTORY PLANNING: STATE OF THE ART

Some authors use trajectory planning as a synonym for path planning, but this is not accurate. Path planning is restricted to the geometric aspects of the motion. The only constraints that can be taken into account are time-independent constraints such as stationary obstacles and kinematic constraints. From the other point of view, trajectory planning with its time dimension permits to take into account time dependent constraints such as moving obstacles and the dynamics constraints of the robot, i.e. the constraints imposed by the dynamics of the robot and the capabilities of its actuators. In other words, trajectory planning consists of creating a detailed specification of the motion of a manipulator that will cause it to proceed from an initial position to a goal position and usually involves some specification of the time parameters of the path (a sequence of positions, velocities and accelerations). As the trajectory is executed, the tip of the end effector traces a curve in space and changes its orientation. This curve is called the path of the trajectory. The curve traced by the sequence of orientations is sometimes called the rotation curve. However, since an infinite number of solutions exist to move from one point to another, a suitable minimum-time trajectory must be found to achieve high-productivity in a particular application.

The resolution of efficient trajectory planning problem with prevention of collisions for robots in complex environments requires computationally costly algorithms that prevent their industrial application. Mainly, these algorithms act as sequential form, so that in the first place the path is obtained and subsequently the

trajectory is adjusted, remaining this seriously conditioned by the result of the first phase where the criteria of optimality associated to dynamic parameters cannot be utilized.

Actually, there are two approaches dealing with trajectory planning problem for a dynamic system. The first one called decoupled or indirect approach, which includes first seeking a path in the configuration space and then finding a time optimal time scaling for the path subjected to the dynamic constraints of the manipulator. The second one named direct or global approach, where the search takes place in the system's state space. This approach involves optimal control, numerical optimization, and grid-based searches.

One of the most important issues in trajectory planning for industrial manipulators is increasing the productivity. Increase the productivity done in the way that instead of increasing actuator size and power, which leads to increase the inertia of the actuators themselves, cost, and power consumption of the lager actuators, minimize the trajectory time needed to perform a given task, Bobrow et al. 1985.

Generally, manipulator trajectories can be planned either in joint space which directly specifying the time evolution of the joint angles, or in Cartesian space which deals with the position and orientation of the end frame. In Cartesian space, calculated values must still be converted to joint values through inverse kinematic equations, which is a very expensive computational process, while in joint space, generated values relate directly to joint values. In joint space, the geometric problems with Cartesian space paths related to workspace and singularity can be avoided.

The first solution to the problem of minimum time planning between given end points for a manipulator introduced by Kahn and Roth 1971. An

approximation scheme based on the linearization of the robot dynamics was proposed to compute the optimal trajectories. The manipulator consisted of three-links serial mechanism with constant limits on the torques. This method is however effective only when the system motion is confined to a small region near the terminal configuration where the linearity assumption is valid.

The research did not attract much attention until early 80s. During the decade of 80s, many researchers have started to solve the time optimal control trajectories problem for serial chain robotic manipulators. The approaches to solve this problem can be classified into two groups: the standard optimal control theoretical approach and non-standard approximation approaches such as search techniques from artificial intelligence and nonlinear parameter optimization methods.

Hollerbach 1983 outlined an algorithm that finds a uniform time-scaling law of a trajectory to make it feasible given actuator torques. He also showed that it might be necessary to speed up a trajectory to make it dynamically feasible. Hollerbach led to a formulation where the time scaling factors are linear variables. With time-scaling algorithms in hand, the problem of finding a collision free trajectory for n-joint manipulator in its $2n$ -dimensional state space can be decoupled into the computationally simpler problem of planning paths in the n -dimensional configuration space followed by time optimal time scaling according to the manipulator dynamics. Shiller and Dubowsky 1988 used the idea of decoupling to find the global time optimal trajectories for a manipulator by considering the time optimal time scaling of a large set of paths. After the first set of paths is selected, each path is smoothed with cubic splines. Kieffer et al. 1997 presented a nearly time optimal path tracking control for non-redundant robotic manipulators using online trajectory time-scaling laws and dynamics. Akella and Peng 2004 exploited the time-scaling law identified by Hollerbach, which

decouples the path and timing along the path, to generate time-warped trajectories to coordinate multiple manipulators.

The first formalization of the problem of finding the optimal curve interpolating a sequence of nodes in the joint space, done by Lin et al. 1983. They proposed cubic (spline) polynomial functions for a trajectory planning where the total traveling time is minimized under kinematic constraints on joint velocities, accelerations, and jerks. Cubic polynomials are widely used for interpolation since they prevent the large oscillations of the trajectory, which can result with higher-order polynomials. Many years later, Angeles et al. 1988 proposed an alternative approach to trajectory planning which is also based on the concept of spline functions, but in these cases, no equation solving is required. The trajectory is synthesized from the scaling of a suitably normalized spline.

Thompson and Patel 1987 developed a procedure using B-splines for constructing robot trajectories. The robot motion was specified by a sequence of positions and orientations knots of the end-effector. B-splines were used to fit the sequence of joint displacements for each joint. Wang and Horng 1990 used the same algorithm presented by Lin et al. 1983, but the trajectories are expressed by means of cubic B-splines. Thompson algorithm and Wang and Horng algorithm had been used to generate the constrained minimum time joint trajectories for Puma 560. Bartels et al. 1987 stated in their book that B-spline polynomials provide local control of the joint trajectory. Chen 1991 had applied uniform cubic B-splines to compute point-to-point minimum time trajectories problem for robotic manipulators subject to state and control constraints. Jamhour and Andre 1996 modified Lin algorithm, so that it can deal with dynamic constraints and with general type objective functions. Steffen and Samarago 1996 used polynomial functions to represent the path between two adjacent trajectory points in the joint space. Continuity conditions to guarantee a smooth motion for the manipulator are

used to spline lower degree polynomial together. Angeles 1997 proposes trajectories with higher-order polynomials that allow the definition of intermediate coordinates in the Cartesian space, these intermediate coordinates lie between the initial and the final position, that are determinate in order to avoid collision.

The planners constructed to obtain optimum trajectories with respect to the execution time has been modeled in Kim and Shin 1985 developed a minimum time trajectory in joint space considering the manipulator dynamics and subjected to torque constraints. The trajectory was formed of a series of straight lines with specified path deviation at the corner points. By deriving bounds on the joint space acceleration from the manipulator torque limits based on a heuristic approximation, the problem was divided into a set of one-dimensional optimization problems, which could easily be solved. Bobrow et al. 1985, Shin and McKay 1985 independently derived similar, and much more efficient, algorithms for determining the time-optimal manipulator trajectory along a given path. The algorithms consider full arm dynamics and actuator torque limits. Subsequently, a computational enhancement to the algorithm was reported by Pfeiffer and Johanni 1987, Slotine and Yang 1989. Rajan 1985 characterized the trajectory using splines and computed the minimum time trajectory of two degrees of freedom manipulator arm based on the approach proposed by Bobrow et al. 1985. Their solution is found by applying a different algorithm based on dynamic programming. Shin and McKay 1986 employed a dynamic programming technique to find the minimum-time trajectories along a prescribed geometric path under the actuator constraints such as torques, assuming the robot full dynamics are available. Many years later, Kieffer et al. 1997 proposed two schemes for adapting time optimal trajectory planning algorithms for robots under computed torque control.

When obstacles are moving, the planner must compute a trajectory parameterized by time, instead of simply a geometric path. This problem has been proven to be computationally difficult even for robots with few DOFs by Reif and Sharir 1985. To coordinate the motion of multiple objects Erdmann and Lozano-Pérez 1986 introduced the notion of configuration \times time space, which is later extended to state \times time space by Fraichard 1993, where a state encodes a robot's configuration and velocity, to plan robot motions with both moving obstacles and kinodynamic constraints. Two months later, Fraichard and Laugier 1993 developed an approach addresses dynamic trajectory planning, which considered as an extension to the path-velocity decomposition proposed by Kant and Zucker 1986. Fraichard and Laugier introduced the concept of adjacent paths used within a novel planning schema operated in two stages: path planning, a set of collision free adjacent paths were computed considering kinematic constraints. Then, trajectory planning, determine the motion of the robot along and between these paths to avoid the moving obstacles considering dynamic constraints of the robot. The reader can refer to Fraichard 1993, Fiorini and Shiller 1995, 1996, Fraichard 1998, 1999, Kuffner and Latombe 2000, Hsu et al. 2002 for more details about the trajectory planning of robots moving in dynamic environments.

Fortune et al. 1986 described a global algorithm for finding collision-free trajectories for two planner manipulators, with one prismatic joint and one revolute joint, by characterizing the combinatorial structure of the configuration space of the two arms. In the same year, Erdmann and Lozano-Pérez 1986 constructed the configuration space-time for several planner manipulators, each with two revolute joints. The trajectories of the manipulators are planned one at a time, using the swept volume, in space/time, of the previous trajectories as obstacles. In the same time, also, Geering et al. 1986 proposed an algorithm to obtain time optimal trajectories for several two links robot arms by solving the

resultant nonlinear two point boundary value problem via the shooting and a parameter optimization method.

O'Dunlaing 1986 presented an exact polynomial time algorithm for planning the motion subjected to acceleration constraints. Canny et al. 1988 constructed a polynomial time algorithm to compute a near optimal trajectory on nonlinear grid in the phase space. Canny et al. 1990 developed an exact exponential-time algorithm for the time-optimal trajectory of a point robot, with velocity and acceleration bounds, in two dimensions.

Chen and Vidyasagar 1988 developed an optimal trajectory planner for planar n-link manipulators. A grid of points in the C-space is used to detect collisions with obstacles. Collision points are occurred in groups, and approximated by ellipses. The equations of these ellipses are then used as constraints in the optimal-control formulation, which is solved numerically. The main weakness of this algorithm is the large number of elliptical constraints needed to approximate configuration obstacles for a cluttered environment. A similar method is used to compute time-optimal trajectories of a manipulator that avoids the collision between the manipulator tip and obstacles introduced by Eltimsahy and Yang 1988. O'Donnell and Lozano-Pérez 1989 proposed a trajectory-scheduling algorithm for two manipulators synchronously operating in common workspace.

It has been shown by Chen and Desrochers 1990 that structure of the minimum time control (MTC) law for m-link robotic manipulators required that at least one of the actuators is always in saturation. Their numerical algorithm converts the original problem, possibly a partially singular one, into a totally nonsingular optimal control problem by introducing a perturbed energy term in the performance index.

McCarthy and Bobrow 1992 computed the number of actuators that must be saturated by calculating the acceleration bounds using linear programming. They formulated the equations for manipulators with arbitrary kinematic configuration and showed that the limits on the internal forces can be handled in the same way as the limits on the actuator torques.

Cao et al. 1994 optimized a piecewise cubic polynomial spline to obtain a smooth and time-optimal constrained motion.

Constantinescu and Croft 2000 proposed a method for calculating smooth and time optimal motion for path-constrained trajectories (SPCTOM) subjected to actuator torque and torque rate limits. This algorithm achieved an implicit jerk limitation by limiting the drive force rate, leading to reduced strain, improved tracking accuracy and speed. On the other hand, the algorithm proposed by Pietsch et al. 2003, Pietsch et al. 2005 limited the trajectory jerk explicitly while the drive force rate is implicitly limited.

Piazzzi and Visioli 1997a introduced a deterministic global optimization technique based on an interval algorithm to obtain a global minimum time trajectory subject to constraint on joint accelerations and jerks. In the same year, these authors, Piazzzi and Visioli 1997b proposed also a global algorithm to obtain minimum time trajectory planning of an m-joint industrial robot by means of a newly devised outer cutting plane algorithm. He used piecewise cubic polynomials in the joint space. Piazzzi and Visioli 1997c, 2000 developed an algorithm called interval analysis to globally minimize the maximum absolute value of the jerk along a trajectory using minimax approach. Abdel-Malek et al. 2006 used a minimum-jerk 3D model to obtain the desired trajectory in Cartesian coordinates. In addition, a direct optimization approach was used to predict each joint's profile (a spline curve). The optimization problem has four cost function terms: (1) Joint displacement function that evaluated displacement of each joint away from its

neutral position. (2) Inconsistency function, which is the joint rate change (first derivative) and its predicted overall trend from the initial to the final target point. (3) The non-smoothness function of the trajectory, which is the second derivative of the joint trajectory. (4) The non-continuity function consists of the amplitudes of joint angle rates at the initial and final target points, in order to emphasize smooth starting and ending conditions. They presented as an application example a high redundant upper-body modeling with 15 degrees of freedom.

Gasparetto and Zanotto 2007 stated that in the case of trajectory planning along a given path, all jerk-minimization algorithms that could be found consider an execution time set a priori and do not accept any kinematic constraint. On the other hand, the trajectory planning technique proposed by him did not require the execution time to be imposed; moreover, kinematic constraints are taken into account when generating the optimal trajectory, and they defined on the robot motion before running the algorithm. Such constraints are expressed as upper bounds on the absolute values of velocity, acceleration and jerk for all robot joints, so that any physical limitation of the real manipulator can be taken into account when planning its trajectory.

LaValle and Hutchinson 1996 considered multiple robots with independent goals. This problem was treated before by Buckley 1989 and Bien and Lee 1992. LaValle and Hutchinson developed performance measures parameters and proposed algorithms optimizing a scalarizing function, which is a weighted average of individual performance functions.

Saramago and Steffen 1998 had formulated off-line joint space trajectories to optimize traveling time and minimize mechanical energy of the actuators (as multi-objective optimization) using cubic spline function subjected to kinematic constraints on the maximum value of velocity, acceleration, and jerk. Saramago and Steffen 1999 proposed an approach to the solution of moving a robot

manipulator with minimum cost along a specified geometric path in the presence of obstacles. The optimal traveling time and the minimum mechanical energy of the actuators are considered together to build a multi-objective function. They applied that approach a two degrees of freedom manipulator arm. Saramago and Junior 2000 presented a general methodology for the off-line three-dimensional optimal trajectory planning of robot manipulators in the presence of moving obstacles. The obstacles are protected by spherical or hyper-spherical security zones, which are never penetrated by the end-effector. The end-effector is represented in the model as a single point. They also considered all second order terms were included in the dynamic equations of motion and friction. Saramago and Steffen 2001 introduced two different strategies to optimize the trajectory-planning problem of robot manipulators in the presence of static obstacles. The first strategy, the trajectory must pass through a given number of points. The second one, the trajectory passes directly from the initial point to the final one. The trajectories were defined using spline functions, and were obtained through off-line computation for on-line operation. Sequential unconstrained minimization techniques (SUMT) have been used for the optimization.

Choi et al. 2000 had discussed the problem of the minimum time trajectories and the control strategy to drive the robots along the trajectories when the exact dynamics equations of robots are unavailable (because of the difficulty for obtaining accurate dynamic equations in some cases and the kinematic approaches might be more appropriate than the dynamic ones). In each time interval, the trajectory is optimized by means of the use of evolution strategy so that the total traveling time is minimized under the kinematic constraints. The trajectory between the knot points, specified to describe the desired path, is built by cubic polynomials and parameterized by time intervals between the knot points.

Furukawa 2002 proposed an approach to search for a minimum time suboptimal trajectory for a general discrete nonlinear system. In his approach the relation between the input control and the time are partitioned into piecewise constant function. This function and time step are searched then by a general purpose nonlinear programming optimization method.

Valero et al. 2006 proposed a trajectory planner approach for industrial robots operating in the presence of obstacles. The dynamic constraints related to the characteristics of the robot when it evaluated the motion between configurations were considered. Valero and his research group presented a mixed planner (according to Tournassoud 1988 classifications) which avoids local minimum problems and considering the dynamics behavior of the robot, to generate trajectories in two stages: obtaining a discrete space of feasible configurations between two feasible ones (initial and final configurations), and then, obtain the optimal and feasible trajectory. The configuration space generation based on the concept of adjacent configuration developed by Valero et al. 1997, Valero et al. 2000 which enables to consider the generation of free-collision configurations as an optimization problem. They validated the functionality of the algorithm by applying it on robot Puma 560 with six degrees of freedom. The robot system and the workspace were modeled using Cartesian coordinates. Abu-Dakka et al. 2007 introduced an algorithm to optimize the trajectory between adjacent configurations constructing a discrete space of these configurations. This approach based on the one proposed by Valero et al. 1997, but the difference is that the robot system was modeled using joint space coordinates (generalized coordinates).

Rubio 2006 introduced in his thesis a simultaneous algorithm based on adjacent configurations for trajectory planning. Rubio et al. 2009b proposed a simultaneous direct approach for the trajectory-planning problem for industrial

robots in environments with obstacles, where the trajectory was created gradually as the robot moves. Their method deals with the uncertainties associated with lack of knowledge of kinematic properties of via points since they are generated as the algorithm evolves. One year later Rubio et al. 2010 tested the simultaneous approach with different interpolation functions.

1.4. TRAJECTORY AND PATH PLANNING USING GENETIC ALGORITHM: STATE OF THE ART

The growing interest for more flexible and autonomous industrial robots leads to the need for automatic path planning and robust obstacle avoidance algorithms. Several different procedures have been suggested as mentioned above. Here, a history of techniques for obstacle avoidance for path planning and trajectory planning based on Genetic Algorithm (GA) will be introduced.

The main difficulties with finding an optimum path arise from the fact that the complexity of the system means that analytical methods may be intractable, while enumerative search methods are overwhelmed by the size of the search space. Enter the genetic algorithm. GAs were first introduced by Holland 1975 based search and optimization techniques have recently found increasing use in machine learning, robot motion planning, scheduling, pattern recognition, image sensing and many other engineering applications.

In principle, GAs are stochastic search algorithms analogous to natural evolution based on mechanics of natural selection and natural genetics. They combine survival of the most fitting among the string structures with randomized yet structured information exchange to form a search algorithm with innovative flair of natural evolution. GAs have proven their robustness and usefulness over

other search techniques because of their unique procedures that differ from other normal search and optimization techniques in four distinct ways:

1. GAs work with coding of a parameter set, not the parameters themselves.
2. GAs search from a population of points, not a single point.
3. GAs use payoff (objective function) information, not derivative or other auxiliary knowledge.
4. GAs use probabilistic transition rules, not deterministic rules.

Numerous implementations of GAs in the field of robot path and trajectory planning have been carried out in the last decade.

Parker and Goldberg 1989 applied GAs to an inverse kinematics problem in which a redundant robot's maximum joint displacement in a point-to-point positioning task was minimized. The robot had four degrees of freedom, which allowed for an infinite number of joint solutions for arbitrary positioning of the end-effector within the three-dimensional workspace. The robot end-effector was assumed to be at some initial position with known initial joint angles. The world coordinates of the desired final position of the end-effector were specified. The fitness function combined two terms: world-positioning error at the achieved point and joint angle displacements from the initial position. The GA was applied to find the joint angles that would position a robot at the target location while minimizing the largest joint displacement from the initial position.

Davidor 1991 described a novel approach to the problem of the complexity of the optimization techniques typically used for redundancy resolution. He applied a GA to generate and optimize robot trajectories in two-dimensional space.

A 3-link planar (i.e. redundant) robot was used in his simulations. The start and goal points and the path between them (i.e. a straight line in two-dimensional space) were known. Given the triplet of joint positions $(\Theta_1, \Theta_2, \Theta_3)$ “gene” and length of each link, the end-effector’s position is uniquely determined. That is means; each gene represents one position or arm configuration on the movement path of the robot arm. The actual trajectories are formed by joining several arm configurations to yield the sequence of path knot points:

$$\{(\Theta_1, \Theta_2, \Theta_3)_1(\Theta_1, \Theta_2, \Theta_3)_2 \dots (\Theta_1, \Theta_2, \Theta_3)_i \dots (\Theta_1, \Theta_2, \Theta_3)_n\} \quad (1.1)$$

where $i = 1, 2, \dots, n$ designates the order of execution according to the ascending value. A trajectory could be found which minimized the sum of the position errors at each of the knot points along the path.

Khoogar and Parker 1991 also examined the path-planning problem in a two-dimensional space by developing an offline approach that used Cartesian space, which is simpler than configuration space and does not require complex, time-consuming mapping of the whole workspace. They also used a planar 3 degrees of freedom robot and introduced rectangular obstacles into the work envelope. The GA was used to plan a collision free trajectory of the robot from an arbitrary starting point to a desired goal point. The encoding method involved specifying N incremental moves, each of which had a small finite value. In an unusual coding scheme, the direction of the incremental joint moves for each joint were coded with ternary numbers $\Delta = (-1, 0, 1)$, where -1 represents a small rotation in the negative direction, 0 represents no move, and 1 represents a small rotation in the positive direction. Therefore, for a three degrees of freedom robot a set of $3*N$ ternary numbers can represent N successive moves all coded within a single string:

$$\{(\Delta_1, \Delta_2, \Delta_3)_1(\Delta_1, \Delta_2, \Delta_3)_2 \dots (\Delta_1, \Delta_2, \Delta_3)_i \dots (\Delta_1, \Delta_2, \Delta_3)_N\} \quad (1.2)$$

The fitness function incorporated the distance from the goal at the end of the N moves, and a penalty if any part of the trajectory involved a collision with the obstacle. This algorithm did not guarantee that the path would reach the goal point; if it did not, the GA would be restarted with the final set of joint angles as the new start point. In addition, a heuristic was involved to move the robot out of a trapped configuration.

Shiller and Dubowsky 1991 proposed a method to solve optimal trajectory with collision-free problem. Their method searched for a small number of candidates of optimal trajectory in a discretized workspace. Then the trajectory was improved using the gradient method. It is easy predicted that it takes too much time all over the workspace.

Ahuactzin et al. 1992 introduced a GA technique to solve the inverse kinematic problem. Moreover, they used a GA to search over a set of Manhattan paths to find collision-free paths for planner manipulators with multiple degrees of freedom. They apply a similar technique, coding the search space in terms of a list of “rotate” and “move” commands for the individual joints to plan paths for holonomic mobile robots. Many years later, this author with others extended this work through the development of the Ariadne’s Clew algorithm, Mazer et al. 1998, which utilizes both an explore function to build a representation of accessible space and a search function which looks for the target end state. This algorithm proved capable of planning collision-free paths for a six degree of freedom manipulator allowing it to avoid a separate six degrees of freedom manipulator driven by random trajectory commands.

Zhao et al. 1992 addressed a path-planning problem for a mobile manipulator system using genetic algorithms. Their simulation system was 3 degrees of freedom arm mounted on 2 degrees of freedom mobile base.

Shibata and Fukuda 1993 proposed an approach for multi-agent system coordinated motion planning by using GAs and fuzzy logic. In their approach, each mobile robot planned its motion while considering the known environment and using empirical knowledge for the unknown environment that included the other robots. Each robot had a starting point in the graph under the assumption that each robot passed a node only once or not at all. A path for a mobile robot was encoded based about traversed nodes. These points were selected randomly at first, while adjacent numbers must be connected with a link in the graph. Since order based strings were used, specialized operations of crossover and mutation were implemented.

In another interesting application, Ram et al. 1994 applied GAs to the learning of local robot navigation behaviors for a reactive control system. The method was applied to a mobile robot simulation in a two-dimensional world with stationary obstacles and known start and goal positions. They employed GAs to optimize the control parameters of the robot navigation in the system. Three motion primitives (move to goal, avoid obstacle, and noise) were embedded in the robot controller. A GA was used to determine optimum combinations of these primitives for three different global behaviors of the mobile robots (safe, fast, and direct) in three environments of varying degrees of obstacle ‘clutter’. A safe robot was optimized to avoid hitting obstacles. While both avoid collisions, fast robots prioritized speed whereas direct robots preferred shortest trips.

Toogood et al. 1995 a GA was used to find a collision-free trajectories for 3R (three degrees of freedom revolute manipulator) robot with specific start and goal joint configurations, among known stationary obstacles. A local XY -coordinate system was defined on each search plane with the origin located at the node point and the local X -axis parallel to the $\Theta_1 - \Theta_2$ plane. The parameters X and Y on each search plane are each coded as an M -bit binary number (typically, M

was in the range of 4 to 8). Parameters that describe the entire trajectory are then concatenated into a $2*N*M$ binary string for processing by the GA as:

$$\{(X_1, Y_1), (X_2, Y_2), \dots, (X_N, Y_N)\} \quad (1.3)$$

Thus, all three angles $(\Theta_1, \Theta_2, \Theta_3)$ at node i are able to be defined by any point generated on the search plane through the mapping $(X_i, Y_i) \rightarrow (\Theta_1, \Theta_2, \Theta_3)$, which reduce the number of variables from 3 to 2 to describe each knot point. In this way, the entire path was given by:

$$S\{(\Theta_1, \Theta_2, \Theta_3)_1, (\Theta_1, \Theta_2, \Theta_3)_2, \dots, (\Theta_1, \Theta_2, \Theta_3)_i, \dots, (\Theta_1, \Theta_2, \Theta_3)_N\}G \quad (1.4)$$

where S and G represent the start and goal points respectively.

Shibata et al. 1995 proposed a motion planning method using a GA in order to cut a three-dimensional work-piece using six degrees of freedom redundant manipulator. In this case, the rotational angles of end-effector along a path are used as the evaluation function.

Sugihara and Smith 1996 proposed a GA for three-dimensional path planning of a mobile robot in an environment possibly with unknown obstacles and moving obstacles, where the three-dimensional space was approximated with grid cells in a rectangular discrete space.

Yun and Xi 1996 used GAs for optimum motion planning problem in joint space. Yun and Xi algorithm incorporates kinematics, dynamics, and control constraints. They used a binary string as a way to represent the variables parameters. Each parameter is coded with l bits (genes), so the encoding form is as follows:

$$\left(\overbrace{b_{1,1}, b_{1,2}, \dots, b_{1,l}}^{x_1}, \overbrace{b_{2,1}, b_{2,2}, \dots, b_{2,l}}^{x_2}, \dots, \overbrace{b_{m,1}, b_{m,2}, \dots, b_{m,l}}^{x_m} \right) \quad (1.5)$$

where $x_i; i = 1, \dots, m$ are parameters, $b_{i,j}$ is the j^{th} bit of the i^{th} parameter.

This method works well only when the number of parameters is small. To verify their algorithm, a simulation results was carried out for two and three degrees of freedom robots.

Kubota et al. 1998 presented a hierarchical trajectory planning method for a redundant manipulator based on a virus evolutionary genetic algorithm. Firstly, they generate a set of configurations that are collision-free by using outputs of the learned neural network, and then apply their virus evolutionary genetic algorithm to refine the collision free trajectory.

Vadakkepat et al. 2000 combined GAs with the artificial potential field to derive optimal potential field functions, introducing a new methodology named Evolutionary Artificial Potential Field (EAPF). This is done to extend the basic artificial potential field approach, which is efficient at finding safe paths, but not typically optimal ones. Rather than adjusting the path explicitly, this technique adjusts the potential functions around the goal and obstacles in order to implicitly optimize the resulting path through the aggregate potential fields. The search space is represented by a set of tunable values parameterizing or “shaping” the various potential fields (multiplicative factors and powers). In this approach, the authors used genotype structures that represent local distance and direction in contrast to represent the whole path because of their simplicity to process and allow for faster real-time performance, while this way may not allow the robot to reach its target.

Tian and Collins 2005 analyzed the reachable workspace of two degrees of freedom robot and derived a condition for singularity avoidance. Afterwards, they

applied GA method (with the property of keeping the elitists results in the current generation to the next generation) to search for the optimal of the two degrees of freedom robot base. The robot end-effector moves in XY plane. They encoded the coordinates (x_b, y_b) of the location of the robot base into a chromosome, which is a binary string.

1.5. OBJECTIVES

The principle objective of this thesis is to provide efficient algorithms using genetic algorithms to solve the path planning and trajectory planning problems for industrial robots in complex environments and making clear the difference between them.

The proposed method has been built in a way such to be applicable to any robotic system working in an industrial environment. Particular examples have been developed on robot Puma 560.

In this work the kinematic and dynamics of the serial chain manipulators are worked out. In the formulation, it is assumed that the mechanical system is formed by rigid links interconnected with ideal revolution joints. The direct and inverse kinematic problems have been focused on, in addition to recursive relations for calculating the position, velocity and acceleration of each reference system contained as a function of the generalized coordinate. The recursive Newton-Euler for dynamic formulation has been addressed.

An efficient collision detection algorithm has been built to check the collisions between robot's arms and obstacles in the workspace. The key here is the way of building the obstacles by means of basic patterns.

1.6. ORGANIZATION OF THE THESIS

This thesis is organized as follows: in Chapter 2 the formulation of the kinematics and dynamics of the serial chain manipulator is performed for obtaining the equations of motion of the mechanical system, particular example Robot Puma 560. In addition, the workspace modeling is developed and the collision detection algorithms.

Chapter 3 introduces path planning problem and an optimization technique using genetic algorithm to find the shortest path between two given configurations of the robot.

Meanwhile, in Chapter 4 the trajectory planning optimization is addressed. Adjacent configuration concept has been treated and a detailed formulation has been produced. A genetic algorithm procedure to solve the adjacent configuration problem and trajectory-planning problem with new crossover and mutation operators has been discussed. Finally, a genetic algorithm procedure has been produced to solve the clamped cubic spline to obtain smooth trajectory with continuous derivatives.

Finally, Chapter 5 contains some conclusions about the most relevant aspects covered in this work. In addition, it provides some guidelines for future subjects that still need more investigation in the dynamic identification and simulation fields.

CHAPTER 2

PROBLEM MODELING

The strategy used to solve the problem requires the modeling of the robot as a function of generalized coordinates moving in a complex workspace discretized and constructed in Cartesian coordinates. This will facilitate the ramification process used by the genetic algorithm procedure to construct and find the best collision free path or trajectory in the discrete workspace between two given configurations of the robot.

In this chapter, the kinematics and dynamics formulation of robotic system, application example Puma 560, is defined as well as the formulation of obstacle avoidance process.

2.1. ROBOT MODELING

The robotic system has been modeled as function of generalized coordinates and considered as a wired model. This model consists of rigid links joined together by the corresponding kinematic joints. Although the robot configuration has been

modeled as a function of joint variables $C^j(q_i)$, the workspace and obstacles have been modeled in Cartesian coordinates (Section (2.4)) to facilitate the definition of the whole collision avoidance process. To achieve that, the robot configuration should be expressed in Cartesian coordinates.

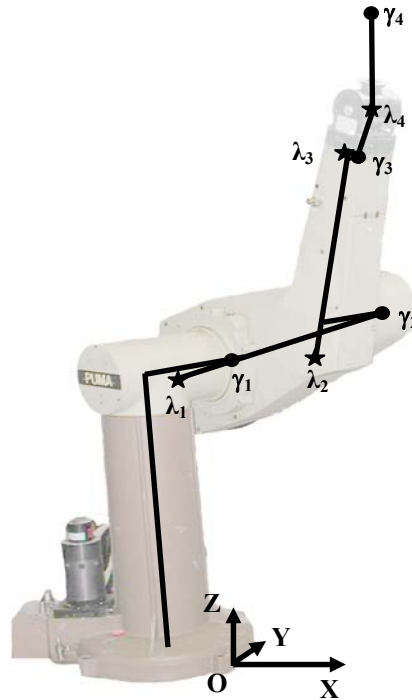


Figure 2.1: Robot Wired Model.

The robot configuration can be expressed in Cartesian coordinates as a set of points called significant points $\gamma_m^j(q_i)$ and interesting points $\lambda_k^j(q_i)$, see Figure (2.1). Significant points have been modeled as a function of joint coordinates and expressed in Cartesian coordinates to facilitate the formulation of the collision avoidance process. The selection of these points is made based on the degrees of freedom of the robot. These points should be as minimum as possible to define sin ambiguity the configuration of the robot. It is important to emphasize that they do

not constitute an independent set of coordinates. To improve the efficiency of the collision detection algorithm, interesting points $\lambda_k^j(q_i)$ have been modeled as function of joint coordinates and expressed in Cartesian coordinates. The interesting points' coordinates are obtained from the significant points and the geometric characteristics of the robot. The robot configuration $C^j(q_i)$ has been converted to the Cartesian coordinates $C^j(\gamma_m^j, \lambda_k^j)$ to facilitate the collision avoidance technique.

In the Figure (2.1), an application example is shown for Puma 560 robotic system with four significant points $\gamma_m^j(q_i) \Rightarrow (\gamma_1^j, \gamma_2^j, \gamma_3^j, \gamma_4^j)$ and four interesting points $\lambda_k^j(q_i) \Rightarrow (\lambda_1^j, \lambda_2^j, \lambda_3^j, \lambda_4^j)$.

2.2. KINEMATIC PROBLEM

The scope of this Section is about the kinematic position analysis of an open chain mechanical system in a recursive way. Kinematics is part of the science of motion that treats motion regardless of the forces that cause it. For instance, and depending on the geometric description of the manipulator, it is necessary to find the mathematical relations between the positions in coordinates of the workspace and the joint variables that conform the configuration space, Lozano-Pérez 1983. These relations denominate forward (or direct) and inverse kinematics respectively, depending on the transformation sense. Within the science of kinematics, the position, the velocity, the acceleration, and all higher order derivatives of the position variables (with respect to time or any other variable(s)) can be studied.

In this thesis, the positioning problem of the manipulator linkages will be considered. Forward kinematics is defined as the geometrical problem to obtain the Cartesian position and orientation of a robot's end-effector given its joint coordinates, see sub-Section (2.2.2). However, inverse kinematics is the opposite problem, where a set of joint angles should be found for a given position and orientation of the end-effector. Later, in Chapter 4, the velocities and accelerations will be derived from the interpolating polynomial (represents the moving curve of the end-effector) to use them in the inverse dynamic solver.

In the following sub-sections, the analytical relationship between the joint angles and the end-effector position and orientation will be described. In order to study them, the structure of the *kinematic chain* has to be considered first.

2.2.1. Coordinate System

A kinematic chain maybe thought of as a set of rigid bodies connected by joints. These bodies are called *links*. The joints are usually rotational, but may also be prismatic. The rotation maybe performed in three orthogonal directions depends on the type of joint. This is called the degree of freedom (DOF) of the joint. Any joint with n degrees of freedom may be modeled as n joints of one degree of freedom connected with $n - 1$ links of zero length. Therefore, without loss of generality, we only have to consider kinematic chain consisting entirely of joints each having just one degree of freedom. The two ends of the kinematic chain are called the *base* and the *end-effector* respectively. The base of the chain is fixed at one position while the end-effector can move freely around the space.

In order to describe the kinematic chain accurately and effectively, a convention is required. Denavit and Hartenberg 1955 proposed a matrix method that systematically establishes coordinate systems attached to the rigid body for

each element in the articulated chain. D-H has a 4*4 homogenous transformation matrix representation, which represents the coordinate systems of each link/body of the articulated chain with respect to the coordinate system of the previous one. Thus, through a sequential transformation, the end-effector expressed in its local coordinate system can be transformed and expressed in the global coordinate system. In this thesis, the Modified Denavit-Hartenberg (Modified-DH) notation, presented by Craig 2005, that defines the geometry of each link by means of four independent parameters and defines the location of the corresponding reference frame is used. These parameters allow the calculation of the vector between the origin of the coordinate systems for different links and the rotation matrix between them. Figure (2.2) shows the assignment of these parameters considering revolute joints type. As can be seen, each link has four parameters, namely a_i , α_i , d_i , and θ_i .

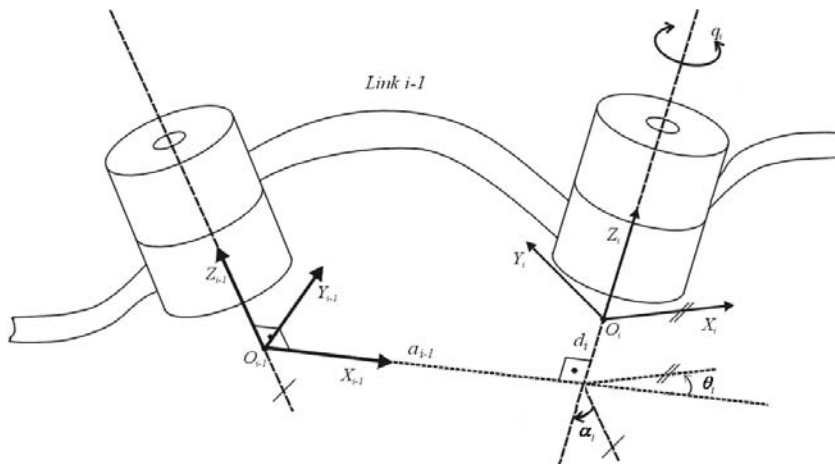


Figure 2.2: Modified Denavit-Hartenberg Assignment Criteria for Link with Revolute Joint.

Depending on the type of the joint, one of them is the joint variable, or generalized coordinate, and the other three are constants. If the joint is revolute, as shown in the figure, then its variable is θ_i , whereas, for a prismatic one it is d_i . Here, the generalized coordinates “ θ_i ” will be denoted by the symbol q_i .

2.2.2. Forward Kinematics

Forward kinematics is the issue to find the position and orientation of the end-effector relative to some coordinate system given a set of joint angles. Using the link parameters defined in the previous section, the transformation matrix ${}^{i-1}T_i$ that transforms a vector in frame $i - 1$ to frame i can be defined.

$${}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_i \\ \sin(\theta_i) \cos(\alpha_{i-1}) & \cos(\theta_i) \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & d_i \sin \alpha_i \\ \sin(\theta_i) \sin(\alpha_{i-1}) & \cos(\theta_i) \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & -d_i \cos \alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

From the transformation matrix, the position vector ${}^{i-1}\vec{r}_{O_{i-1}, O_i}$ and the rotation matrix ${}^{i-1}R_i$ that describe the relative position and orientation, respectively between any two consecutive local reference frames can be extracted as following,

$${}^{i-1}\vec{r}_{O_{i-1}, O_i} = \begin{bmatrix} a_i \\ d_i \sin \alpha_i \\ -d_i \cos \alpha_i \end{bmatrix} \quad (2.2)$$

$${}^{i-1}R_i = \begin{bmatrix} \cos(q_i) & -\sin(q_i) & 0 \\ \sin(q_i) \cos(\alpha_{i-1}) & \cos(q_i) \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) \\ \sin(q_i) \sin(\alpha_{i-1}) & \cos(q_i) \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) \end{bmatrix} \quad (2.3)$$

2.2.2.1. Application: Robot Puma 560

The methods presented in this thesis have been verified and tested over Puma 560 robotic system. Puma 560 is a robot with six degrees of freedom and all are rotational joints. It is shown in Figure (2.3) with link frame assignments in the position corresponding to all joint angles equal to zero.

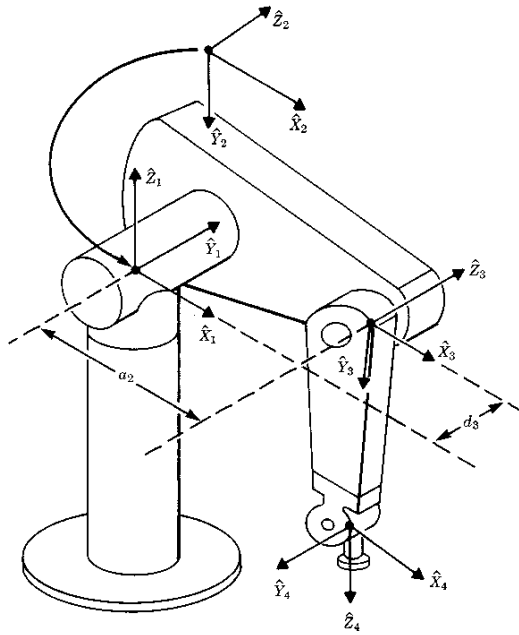


Figure 2.3: Some Kinematic Parameters and Frame Assignments for the Puma 560 Manipulator.

After the calculation of ${}^{i-1}\vec{r}_{O_{i-1},O_i}$ and ${}^{i-1}R_i$, Equations (2.2) and (2.3) respectively, the vector ${}^i\vec{r}_{O_i,O_j}$ denotes the position vector from the origin of the i^{th} reference frame to the j^{th} reference frame expressed in i^{th} reference frame. Hence, the significant points and the interesting points can be found from the following equation

$${}^0\vec{r}_{O_0,p} = {}^0\vec{r}_{O_0,O_i} + {}^0R_i {}^i\vec{r}_{O_i,p} \quad (2.4)$$

where ${}^0\vec{r}_{O_0,p}$ is the vector from the origin of the reference system attached to the base of the robot to the significant or interesting point located in the link i , and p is one of the points $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \lambda_1, \lambda_2, \lambda_3$, and λ_4 .

The partial derivatives of the previous positions relative the generalized coordinates could be obtained according to the following equation:

$$\frac{dp}{dq_i} = \frac{\partial {}^0\vec{r}_{O_0,p}}{\partial q_i} = {}^0\vec{Z}_i \times {}^0\vec{r}_{O_i,p} \quad (2.5)$$

where ${}^0\vec{Z}_i$ is a unit vector in the z -axis direction of the reference frame i expressed in the base reference system 0, Yoshikawa 1990.

With the definitions of these points and their derivatives, it will be easy to obtain and derive the minimum distances between each obstacle in the workspace and the robot's links needed for the prevention of collisions constraints. This will be explained in details in Section (2.5).

2.2.3. Inverse Kinematics

The inverse kinematic problem is about finding the generalized coordinates of a kinematic chain that give rise to a particular end-effector position and orientation. This problem has been extensively studied in robotics. Since computer based, robots are usually driven in joint space, though the objects to be manipulated are expressed in the global coordinate system; the inverse kinematic solution is essential in controlling the position and orientation of the end-effector of the robot arm to reach its goals. The inverse kinematic problem is much more difficult due to the existence of multiple algebraic solutions. There are two classes

of solution methods for the inverse kinematics problem: *closed* form and *numerical*. In robotics, a closed form solution is usually desired for the kinematic chain of a robot arm rather than a numerical solution. Numerical solutions are generally much slower than the corresponding closed form solution. The closed form solution of a kinematic chain can be obtained by one or both of the two solution methods: *algebraic* and *geometric*. In this thesis the algebraic method explained in Craig 2005 will be used to solve the inverse kinematic problem for an application Puma 560.

2.2.3.1. Application: Robot Puma 560

In this sub-section, the inverse kinematic problem for an industrial Puma 560 robot will be formulated. All the following relations are extracted from Craig 2005. Considering the joint variables (q_1, q_2, \dots, q_6) , the transformation matrix of the end-effector $\{T\}$ with respect to the global reference system (the base) is represented

$${}^0_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_X \\ r_{21} & r_{22} & r_{23} & P_Y \\ r_{31} & r_{32} & r_{33} & P_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0_1T(\theta_1) {}^1_2T(\theta_2) {}^2_3T(\theta_3) {}^3_4T(\theta_4) {}^4_5T(\theta_5) {}^5_6T(\theta_6) \quad (2.6)$$

where $\begin{bmatrix} P_X \\ P_Y \\ P_Z \end{bmatrix}$ is the position vector of the end-effector of the robot with

respect to frame 0,

and $\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ is the orientation of the end-effector, where

$$r_{11} = c_1 [c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] + s_1(s_4c_5c_6 + c_4s_6) \quad (2.7)$$

$$r_{21} = s_1 [c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] + c_1(s_4c_5c_6 + c_4s_6) \quad (2.8)$$

$$r_{31} = -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 \quad (2.9)$$

$$r_{12} = c_1 [c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] + s_1(c_4c_6 - s_4c_5s_6) \quad (2.10)$$

$$r_{22} = s_1 [c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] - c_1(c_4c_6 - s_4c_5s_6) \quad (2.11)$$

$$r_{32} = -s_{23}(-c_4c_5s_6 - s_4c_6) + c_{23}s_5s_6 \quad (2.12)$$

$$r_{13} = -c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 \quad (2.13)$$

$$r_{23} = -s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5 \quad (2.14)$$

$$r_{33} = s_{23}c_4s_5 - c_{23}c_5 \quad (2.15)$$

$$P_X = c_1 [a_2c_2 + a_3c_{23} - d_4s_{23}] - d_3s_1 \quad (2.16)$$

$$P_Y = s_1 [a_2c_2 + a_3c_{23} - d_4s_{23}] + d_3c_1 \quad (2.17)$$

$$P_Z = -a_3s_{23} - a_2s_2 - d_4c_{23} \quad (2.18)$$

where $s_i = \sin(q_i)$ & $c_i = \cos(q_i)$, $i = 1, 2, \dots, 6$

$s_{23} = \sin(q_2+q_3)$ & $c_{23} = \cos(q_2+q_3)$ and so on.

Then, the joint variables can be calculated using the following equations, as presented by Craig 2005.

$$q_1 = \text{Atan2}(P_Y, P_X) - \text{Atan2}\left(d_3 \pm \sqrt{P_X^2 + P_Y^2 - d_3^2}\right) \quad (2.19)$$

Note that there are two possible solutions for q_1 corresponding to the plus-or-minus sign in Equation (2.19).

$$K = \frac{P_X^2 + P_Y^2 + P_Z^2 - a_2^2 - a_3^2 - d_3^2 - d_4^2}{2a_2} \quad (2.20)$$

$$q_3 = \text{Atan2}(a_3, d_4) - \text{Atan2}\left(K \pm \sqrt{a_3^2 + d_4^2 - K^2}\right) \quad (2.21)$$

The plus-or-minus sign in Equation (2.21) leads to two different solutions for q_3 .

$$q_{23} = \text{Atan2}\left[\begin{aligned} &(-a_3 - a_2 c_3) P_Z - (c_1 P_X + s_1 P_Y)(d_4 - a_2 s_3), \\ &(a_2 s_3 - d_4) P_Z - (a_3 + a_2 c_3)(c_1 P_X + s_1 P_Y) \end{aligned}\right] \quad (2.22)$$

Equation (2.22) computes four values of q_{23} according to the four possible combinations of solutions for q_1 and q_3 . Then, four possible solutions for q_2 are computed as follows

$$q_2 = q_{23} - q_3 \quad (2.23)$$

where the appropriate solution for q_3 is used when forming the difference.

As long as $s_5 \neq 0$, we can solve for q_4 as

$$q_4 = \text{Atan2}\left(-r_{13} s_1 + r_{23} c_1, -r_{13} c_1 c_{23} - r_{23} s_1 c_{23} + r_{33} s_{23}\right) \quad (2.24)$$

$$\begin{aligned} r_{13}(c_1 c_{23} c_4 + s_1 s_4) + r_{23}(s_1 c_{23} c_4 - c_1 s_4) - r_{33}(s_{23} c_4) &= -s_5 \\ r_{13}(-c_1 s_{23}) + r_{23}(-s_1 s_{23}) - r_{33}(-c_{23}) &= c_5 \end{aligned} \quad (2.25)$$

Hence, q_5 can be solved as

$$q_5 = \text{Atan2}(s_5, c_5) \quad (2.26)$$

Finally, q_6 can be solved as follows:

$$\begin{aligned} s_6 &= -r_{11}(c_1 c_{23} s_4 - s_1 c_4) - r_{21}(s_1 c_{23} s_4 + c_1 c_4) + r_{31}(s_{23} s_4) \\ c_6 &= r_{11}[(c_1 c_{23} c_4 + s_1 s_4) c_5 - c_1 s_{23} s_5] \\ &\quad + r_{21}[(s_1 c_{23} c_4 - c_1 s_4) c_5 - s_1 s_{23} s_5] \\ &\quad - r_{31}(s_{23} c_4 c_5 + c_{23} s_5) \end{aligned} \quad (2.27)$$

$$q_6 = \text{Atan2}(s_6, c_6) \quad (2.28)$$

Because of the plus-or-minus signs appearing in Equations (2.19) and (2.21), these equations compute four solutions. Additionally, there are four more solutions obtained by flipping the wrist of the manipulator. For each of the four solutions computed above, the flipped solution can be obtained by

$$\begin{aligned}q'_4 &= q_4 + 180^\circ, \\q'_5 &= -q_5, \\q'_6 &= q_6 + 180^\circ.\end{aligned}\tag{2.29}$$

After all eight solutions have been computed, some (or even all) of them might have to be discarded due to joint-limit violations. Of any remaining valid solutions, usually the one closest to the present manipulator configuration is chosen after applying the check collision algorithm over these valid solutions, Craig 2005.

2.3. THE DYNAMIC MODEL

The scope of this section deals with the dynamics of robot manipulators. Whereas the kinematic equations describe the motion of the robot without consideration of the forces and torques producing the motion, the dynamic equations clearly describe the relationship between motion and the force. The equations of motion are important to consider in the design of robots, in simulation and animation of robot motion, and in the design of control algorithms. The equations of motion provide the basis for a number of computational algorithms that are useful in mechanical design, control, and simulation. There are two main problems in robot dynamics:

- Forward dynamics problem: consist in finding the characteristics of motion that the robot acquire as a consequence of given actions (the forces are given and the motion is the result). It is used mainly in simulation.
- Inverse dynamics problem (IDP): consist in computing the generalized forces from a specification of the manipulator's trajectory (position,

velocity, acceleration). It has a variety of uses, such as motion control systems, mechanical design and trajectory planning. Several researchers developed $O(n)$ algorithms for inverse dynamics for robotics used a Newton-Euler (NE) formulation of the problem. Stepanenko and Vukobratovic 1976 developed a recursive NE method for human limb dynamics, and Orin et al. 1979 made the recursive method more efficient by expressing forces and moments to local link coordinates for real-time control of a leg of a walking machine. Luh et al. 1980 developed a very efficient Recursive NE Algorithm (RNEA) by expressing most quantities to link coordinates. The RNEA is the most cited method. Hollerbach 1980 developed an $O(n)$ recursive Lagrangian formulation, but found that it was much less efficient than the RNEA in terms of the number of multiplications and additions/subtractions required in the algorithm. Provenzano 2001 introduced an algorithm using Gibbs-Appell equations leads to computationally efficient direct and inverse dynamic problem algorithms. Concerning the formulation that rewrite the inverse dynamic problem in its linear form Benimeli 2006 presented an analytical algorithms for this purpose. In these algorithms the equation of motion are provided in their linear matrix form. Mata et al. 2002 introduced an algorithm for the inverse and direct dynamic problem constructed based on the formulation of Gibbs-Appell. Links only were considered and the inertia matrices were assumed to be given with respect to the center of gravity.

In this thesis, as the IDP is not the main concerns, the recursive Newton-Euler formulation proposed by Luh et al. 1980 will be used because of its intuitive and efficiency.

2.3.1. Inverse Dynamics Problem

As mentioned above, the inverse dynamic is the problem of determining the forces required to produce a prescribed motion, as well as the constraint moments and forces, i.e., the reactions at the joints. In this thesis, the dynamic model of the manipulator is obtained by solving the recursive Newton-Euler formulation to obtain the joint torques required for a given set of positions, velocities, and accelerations (q, \dot{q}, \ddot{q}) (see sub-Section 4.1.2)) of the joint angles for Puma 560 robot.

The iterative Newton-Euler dynamic formulation has two-step processes consisting of an outward loop and an inward loop. The forward recursion or outward iteration propagates kinematic information — such as angular velocities, linear and angular accelerations— from the base reference frame (inertial frame) to the end-effector. The backward recursion or inward iteration propagates the forces and moments exerted on each link from the end-effector of the manipulator to the base reference frame.

2.3.1.1. Outward Loop:

To calculate the inertial forces acting on each link of the model we have to calculate the angular velocity and linear and angular acceleration of the centre of masses of each link. This is done by the outward loop starting from link 1 and going up to link n (last link).

Angular velocity propagation from link i to link $i + 1$ expressed in reference frame $i + 1$ can be determined using the following equation,

$${}^{i+1}\omega_{i+1} = {}^{i+1}R^i \omega_i + \dot{q}_{i+1} {}^{i+1}\hat{Z}_{i+1} \quad (2.30)$$

where: ${}^{i+1}\omega_{i+1}$ = Angular velocity of the link $i + 1$ expressed in the reference frame $i + 1$.

${}^{i+1}\hat{Z}_{i+1}$ = Unit vector in Z direction in frame $i + 1$.

${}^{i+1}R_i$ = Rotation matrix describing orientation of frame i in frame $i + 1$.

\dot{q}_{i+1} = First time-derivative of joint angle $i + 1$.

The angular acceleration can be transformed from one link to the next by,

$${}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}R_i \left({}^i\dot{\omega}_i + {}^{i+1}R_i \left({}^i\omega_i \times \dot{q}_{i+1} \right) \right) + {}^{i+1}\hat{Z}_{i+1} \ddot{q}_{i+1} + {}^{i+1}\dot{\hat{Z}}_{i+1} \quad (2.31)$$

where:

${}^{i+1}\dot{\omega}_{i+1}$ = Angular acceleration of the link $i + 1$ expressed in the reference frame $i + 1$

\ddot{q}_{i+1} = Second time-derivative of joint angle $i + 1$.

The linear acceleration for point ${}^iP_{i+1}$ is computed by the following equation,

$${}^{i+1}\dot{v}_{i+1} = {}^{i+1}R_i \left({}^i\dot{\omega}_i \times {}^iP_{i+1} + {}^i\omega_i \times \left({}^i\omega_i \times {}^iP_{i+1} \right) + {}^i\dot{v}_i \right) \quad (2.32)$$

Linear acceleration for the centre of mass $P_{C_{i+1}}$ in link $i + 1$ is calculated as follows,

$${}^{i+1}\dot{v}_{C_{i+1}} = {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}} + {}^{i+1}\omega_{i+1} \times \left({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C_{i+1}} \right) + {}^{i+1}\dot{v}_{i+1} \quad (2.33)$$

where: \dot{v}_C is the acceleration of centre of mass

Having obtained the linear and angular acceleration of each link, the next step is to find the inertial force and torque acting at the centre of mass of each link.

$${}^{i+1}F_{i+1} = m_{i+1} {}^{i+1}\dot{v}_{C_{i+1}} \quad (2.34)$$

$${}^{i+1}N_{i+1} = {}^{C_{i+1}}I_{i+1} {}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1} {}^{i+1}\omega_{i+1} \quad (2.35)$$

where: m_i is the total mass of link i .

${}^C I$ is the inertia tensor of the link written in a frame, $\{C\}$, about the centre of mass, Craig 2005.

2.3.1.2. Inward Loop:

In this section, the joint torques required for the motion will be calculated. The iteration in this step are inward due to the fact that calculations now start at the terminal link and work backwards toward the base of the robot. The equations adopted are based on the force and moment dynamic equilibrium equations of a link. All the following equations are extracted from Craig 2005.

From the force balance equation, the following iterative relationship can be deduced:

$${}^i f_i = {}^i R_{i+1} {}^{i+1} f_{i+1} + {}^i F_i \quad (2.36)$$

while from the moment balance equation, the following iterative relationship can be deduced:

$${}^i n_i = {}^i N_{i+1} + {}^i R^{i+1} n_{i+1} + {}^i P_{C_i} \times {}^i F_i + {}^i P_{i+1} \times {}^i R^{i+1} f_{i+1} \quad (2.37)$$

Finally, the joint torques for revolute joints are calculated using the following relationship:

$$\tau_i = {}^i n_i^T {}^i \hat{Z}_i \quad (2.38)$$

where n_i is the torque exerted on link i by link $i - 1$.

The inverse dynamic problem for an industrial Puma 560 robot will be developed by solving the iterative Newton-Euler dynamic formulation, Equations from (2.30) to (2.38). The point ${}^i P_{i+1}$ in Equation (2.32), is the position vector declared in Equation (2.2). In the outward iterations, $i = 0 \rightarrow 5$. In the inward iterations, $i = 6 \rightarrow 1$. For Puma 560 robot, the number of links $n = 6$.

2.4. ENVIRONMENT MODELLING

The workspace and obstacles have been modeled in Cartesian coordinates. The details of the modeling strategy will be found in the next two sub-Sections.

2.4.1. Workspace Modeling

The workspace of a given manipulator has been defined by Craig 2005 as the existence or nonexistence of a kinematic solution. The workspace in this thesis is a subset of Craig definition and is defined as the space that contains at least a set of robot configurations obtained based on a discrete set of end-effector's positions. To achieve that definition, let's consider a rectangular prism between the initial C^i

and goal C^f robot configurations. The end points of the prism's diagonal (represented by γ_4^i and γ_4^f in Figure (2.4)) are corresponding to the positions in Cartesian coordinates of the end-effector of the initial C^i and final C^f respectively. The prism edges are parallel to the global Cartesian reference system.

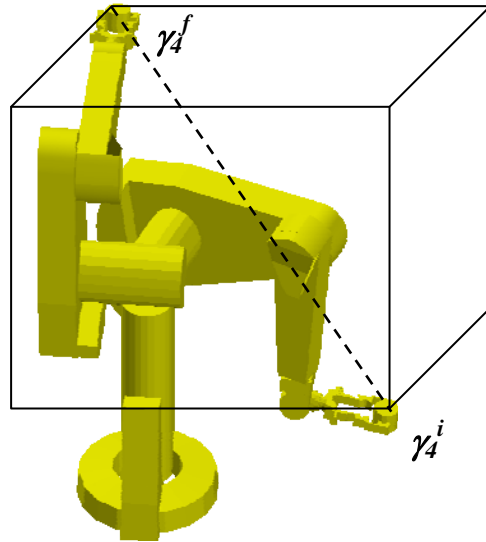


Figure 2.4: Workspace Modeling.

A uniform grid of points is considered inside the prism. These points are far a magnitude small enough ($\Delta_x, \Delta_y, \Delta_z$) to prevent the existence of obstacles between two adjacent points in the grid. Thus, the workspace contains a discrete set of configurations such that the position of the end-effector for each configuration must belong to the previously defined grid. This means that the robot configurations must keep the end-effector inside the prism. The set of positions that can be occupied by the robot's end-effector inside the prism are restricted finite number of points provided by discretizing the prism according to the following increments:

$$\Delta_x = \frac{\gamma_{4x}^f - \gamma_{4x}^i}{Pts_x - 1}; \quad \text{where } Pts_x = 1 + \text{ceil}\left(\frac{|\gamma_{4x}^f - \gamma_{4x}^i|}{D}\right) \quad (2.39)$$

$$\Delta_y = \frac{\gamma_{4y}^f - \gamma_{4y}^i}{Pts_y - 1}; \quad \text{where } Pts_y = 1 + \text{ceil}\left(\frac{|\gamma_{4y}^f - \gamma_{4y}^i|}{D}\right) \quad (2.40)$$

$$\Delta_z = \frac{\gamma_{4z}^f - \gamma_{4z}^i}{Pts_z - 1}; \quad \text{where } Pts_z = 1 + \text{ceil}\left(\frac{|\gamma_{4z}^f - \gamma_{4z}^i|}{D}\right) \quad (2.41)$$

where $\text{ceil}(\text{number})$ returns the smallest integer value that is not less than that *number*. D is less than the size of the smallest obstacle in the workspace or less than the smallest robot's link diameter (depends which is smaller). $(Pts_x - 1, Pts_y - 1, Pts_z - 1)$ are the number of points steps that discretize the prism. The points $(\gamma_{4x}^f, \gamma_{4y}^f, \gamma_{4z}^f)$ and $(\gamma_{4x}^i, \gamma_{4y}^i, \gamma_{4z}^i)$ are the coordinates of the end-effector positions of the initial and final configurations of the robot.

2.4.2. Obstacle Modeling

One of the objectives of path and trajectory planning algorithms is to generate collision-free configurations. To facilitate and systematize the calculation of the distances between the robot links and the obstacles, a generic obstacle models have been constructed in terms of a combination of three basic patterns: Spheres, cylisphere, and quadrilateral planes since they are computationally simple. Very little information has to be stored in order to fully define such elements. Any type of obstacle can be modeled using one or set of these elements.

A sphere is the most basic element that can be used to model an object since it is defined by its centre position and the radius. On the other hand, the cylisphere is a cylinder with hemispheres on each end. The position and orientation of the cylisphere can be defined by locating the position of the end points of the cylinder axis and its radius. Finally, the quadrilateral plane is a basic building block for a wide variety of shapes. It is defined by three points and thickness, Table (2.1).

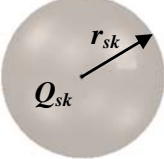
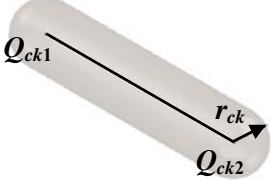
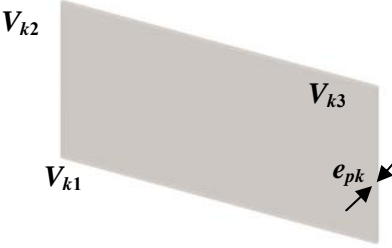
<p>a) Sphere</p> 	<p>Q_{sk} → Centre</p> <p>r_{sk} → Radius</p>
<p>b) Cylisphere</p> 	<p>Q_{ck1} → Centre 1</p> <p>Q_{ck2} → Centre 2</p> <p>r_{ck} → Radius</p>
<p>c) Quadri-lateral plane</p> 	<p>V_{pk1} → Vertex 1</p> <p>V_{pk2} → Vertex 2</p> <p>V_{pk3} → Vertex 3</p> <p>e_{pk} → Height</p>

Table 2.1: The Obstacle Three Basic Elements.

The minimum distance is obtained among these basic elements and the robot's links. In this thesis, obstacles are considered to be static, which means, their positions and orientations do not change with time. The three basic elements can be defined in the space as follows:

According to Lozano-Pérez and Wesley 1979 a process of growing obstacles has been used in order to obtain the actual dimensions of the robot.

2.5. COLLISION AVOIDANCE FORMULATION

As mentioned previously, one of the objectives of path and trajectory planning algorithm is generating free-collision paths or trajectories. For the purpose of preventing collisions between the robot and the obstacles, the distances calculated between the robot's links and the obstacles are considered as constraints in the optimization problem. A method to facilitate the formulation of the shortest distance between any obstacle and robot links, is by shrinking robot links, Section (2.1), and expanding the obstacles, Lozano-Pérez and Wesley 1979.

In the next three sub-Sections, the shortest distance between the three obstacle patterns (Spheres, Cylispheres, Quadri-lateral plane) and the robot links will be calculated. Before starting with the distances derivations, some terminologies should be specified.

\vec{v}_1 is a vector from the global reference system to the point v_1 , its length is $\|\vec{v}_1\|$.

$\vec{v}_1 v_2 = v_2 - v_1$ is a vector from point v_2 to point v_1 , and its length is $\|\vec{v}_1 v_2\|$.

" \cdot " will be used for multiply a scalar " a " with vector " \vec{v}_1 "; i.e. $a \cdot \vec{v}_1$.

" \circ " will be used for Vectors Dot Product; i.e. $\vec{v}_1 \circ \vec{v}_2 = \|\vec{v}_1\| \cdot \|\vec{v}_2\| \cos \theta$.

" \times " will be used for Vectors Cross Product; i.e. $\vec{v}_1 \times \vec{v}_2 = \|\vec{v}_1\| \cdot \|\vec{v}_2\| \sin \theta \cdot \vec{n}$, where

\vec{n} is unit vector normal to \vec{v}_1 and \vec{v}_2 .

$$\overrightarrow{proj}_{v_2}^{v_1} = \frac{\vec{v}_1 \circ \vec{v}_2}{\vec{v}_2 \circ \vec{v}_2} \vec{v}_2 \text{ is the projection of } \vec{v}_1 \text{ on } \vec{v}_2 \text{ and in the direction of } \vec{v}_2.$$

In all cases, the robot link is considered as a cylisphere with radius r_m . This link will then shrinking it to line segment defined by two points: Significant point γ_m and Interesting point λ_m , where $m = 1 \rightarrow 4$ for Puma 560 robot.

2.5.1. Spherical Obstacles

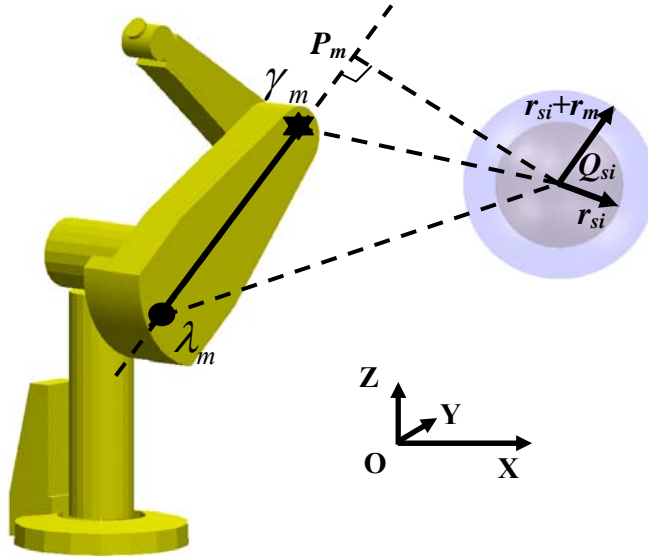


Figure 2.5: Minimum Distance Derivation Between Robot Link and Sphere.

An algorithm has been built to find the shortest distance between sphere and robot links (considered as segments), taking into account the growing obstacle technique. Let's consider the case of the i^{th} spherical obstacle denoted by $S_i(Q_{si}, r_{si})$ and the link segment of the robot defined by γ_m and λ_m .

Consider the Figure (2.5), the minimum distance between the sphere S_i and the robot link will be $\|\lambda_m Q_{si}\|$, $\|\gamma_m Q_{si}\|$, or $\|P_m Q_{si}\|$. The key to know the answer is by calculating the projection of vector $\overrightarrow{\lambda_m Q_{si}}$ on vector $\overrightarrow{\lambda_m \gamma_m}$.

$$\overrightarrow{P_m \lambda_m} = \overrightarrow{proj_{\lambda_m \gamma_m}^{\lambda_m Q_{si}}} \quad (2.42)$$

If the $\|P_m \lambda_m\|$ more than $\|\lambda_m \gamma_m\|$ then $\|\gamma_m Q_{si}\|$ should be compared with $r_{si} + r_m$, otherwise $\|P_m \lambda_m\|$ should be checked if it has a value less than zero. In

this case $\|\lambda_m Q_{si}\|$ should be compared with $r_{si} + r_m$, otherwise $\|P_m Q_{si}\|$ should be compared with $r_{si} + r_m$. So the collision prevention with spherical obstacles can be achieved.

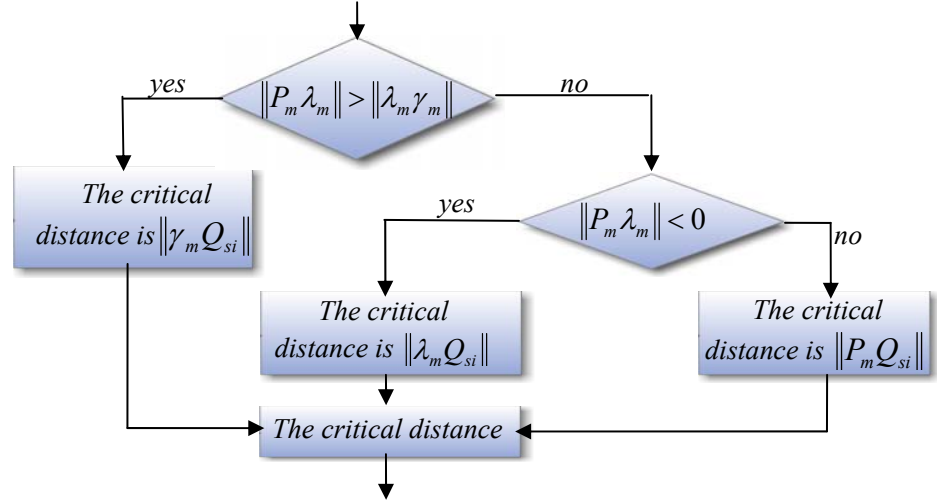


Figure 2.6: The Flow-Chart of the Algorithm to find the Shortest Distance Between Robot Link and Sphere.

The derivatives of those distances with respect to the generalized coordinates are obtained as follow: First the derivative of the length $\|\gamma_m Q_{si}\|$ with respect to the problem variables (generalized coordinates of the robot q_j).

$$\frac{d(\|\gamma_m Q_{si}\|)}{dq_j} = \left(\frac{1}{\|\gamma_m Q_{si}\|} \right) \cdot \begin{pmatrix} (\gamma_{m_x} - Q_{si_x}) \cdot \frac{d\gamma_{m_x}}{dq_j} + \\ (\gamma_{m_y} - Q_{si_y}) \cdot \frac{d\gamma_{m_y}}{dq_j} + \\ (\gamma_{m_z} - Q_{si_z}) \cdot \frac{d\gamma_{m_z}}{dq_j} \end{pmatrix} \quad (2.43)$$

where $j = 1 \rightarrow 6$ for robot Puma 560, i index relates to the number of spheres and j relates to joint variables. $\frac{d\gamma_m}{dq_j}$ can be found using Equation (2.5)

The derivative of $\|\lambda_m Q_{si}\|$ and $\|P_m Q_{si}\|$ with respect to the problem variables is:

$$\frac{d(\|\lambda_m Q_{si}\|)}{dq_j} = \left(\frac{1}{\|\lambda_m Q_{si}\|} \right) \cdot \begin{pmatrix} (\lambda_{m_x} - Q_{si_x}) \cdot \frac{d\lambda_{m_x}}{dq_j} + \\ (\lambda_{m_y} - Q_{si_y}) \cdot \frac{d\lambda_{m_y}}{dq_j} + \\ (\lambda_{m_z} - Q_{si_z}) \cdot \frac{d\lambda_{m_z}}{dq_j} \end{pmatrix} \quad (2.44)$$

$$\frac{d(\|P_m Q_{si}\|)}{dq_j} = \left(\frac{1}{\|P_m Q_{si}\|} \right) \cdot \begin{pmatrix} (P_{m_x} - Q_{si_x}) \cdot \frac{dP_{m_x}}{dq_j} + \\ (P_{m_y} - Q_{si_y}) \cdot \frac{dP_{m_y}}{dq_j} + \\ (P_{m_z} - Q_{si_z}) \cdot \frac{dP_{m_z}}{dq_j} \end{pmatrix} \quad (2.45)$$

where $\frac{d\lambda_m}{dq_j}$ can be found using Equation (2.5).

Consider :

$$A = \frac{d(\overrightarrow{\lambda_m Q_{si}})}{dq_j} \circ \overrightarrow{\lambda_m \gamma_m} \qquad B = \overrightarrow{\lambda_m Q_{si}} \circ \frac{d(\overrightarrow{\lambda_m \gamma_m})}{dq_j}$$

$$C = 2 \cdot \left(\overrightarrow{\lambda_m Q_{si}} \circ \overrightarrow{\lambda_m \gamma_m} \right) \cdot \frac{d(\|\lambda_m \gamma_m\|)}{dq_j} \quad D = \frac{\left(\overrightarrow{\lambda_m Q_{si}} \circ \overrightarrow{\lambda_m \gamma_m} \right) \cdot \frac{d(\overrightarrow{\lambda_m \gamma_m})}{dq_j}}{\left(\|\lambda_m \gamma_m\| \right)^2}$$

$$\frac{dP_m \lambda_m}{dq_j} = \frac{d\left(\overrightarrow{proj_{\lambda_m \gamma_m}^{\lambda_m Q_{si}}} \right)}{dq_j} = \frac{(A+B) \cdot \|\lambda_m \gamma_m\| - C}{\left(\|\lambda_m \gamma_m\| \right)^3} \cdot \overrightarrow{\lambda_m \gamma_m} + D \cdot \overrightarrow{\lambda_m \gamma_m}$$

$$= \frac{d\lambda_m}{dq_j} - \frac{dP_m}{dq_j}$$
(2.46)

$$\frac{d(\|P_m \lambda_m\|)}{dq_j} = \frac{(A+B) \cdot \|\lambda_m \gamma_m\| - (D)}{2 \cdot \|P_m \lambda_m\| \cdot \left(\|\lambda_m \gamma_m\| \right)^2}$$
(2.47)

The derivative of the length $\|\lambda_m \gamma_m\|$ with respect to the problem variables is:

$$\frac{d(\|\lambda_m \gamma_m\|)}{dq_j} = \left(\frac{1}{\|\lambda_m \gamma_m\|} \right) \cdot \left(\begin{array}{l} \left[\left(\lambda_m - \gamma_m \right) \cdot \left(\frac{d\lambda_m}{dq_j} - \frac{d\gamma_m}{dq_j} \right) \right]_x + \\ \left[\left(\lambda_m - \gamma_m \right) \cdot \left(\frac{d\lambda_m}{dq_j} - \frac{d\gamma_m}{dq_j} \right) \right]_y + \\ \left[\left(\lambda_m - \gamma_m \right) \cdot \left(\frac{d\lambda_m}{dq_j} - \frac{d\gamma_m}{dq_j} \right) \right]_z \end{array} \right)$$
(2.48)

2.5.2. Cylispherical Obstacles

Cylisphere is a cylinder with hemispheres on each end. A cylisphere is symmetrical about its ‘long’ axis. The collision avoidance algorithm between the

robot links and cylispheres has been built considering robot links and cylispheres as line segments. Then the minimum distance between two segments has been determined as following. Let's consider the case of the k^{th} cylisphere obstacle denoted by $Cy_k(Q_{ck1}, Q_{ck2}, r_{ck})$ and the robot's link segment is defined by γ_m and λ_m .

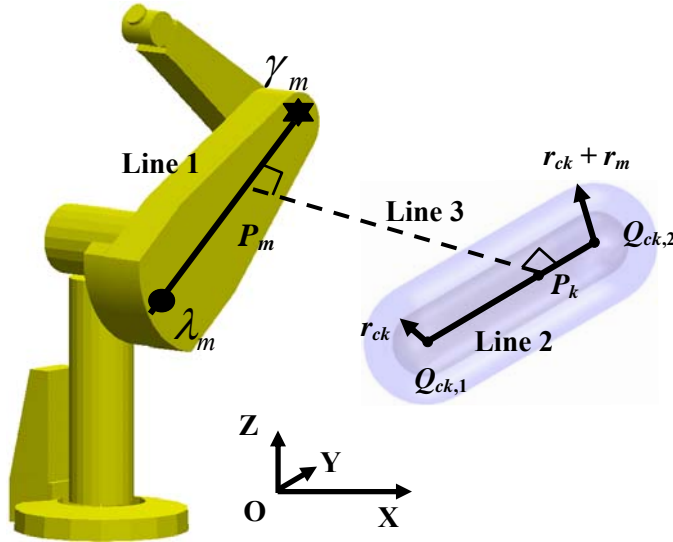


Figure 2.7: Minimum Distance Derivation Between Robot Links and Cylispheres.

In Figure (2.7), Line 1 represents the robot link while Line 2 represents a zero-radius cylisphere. Points Q_{ck1} , Q_{ck2} with the radius r_{ck} represents the cylisphere obstacle. Points P_m and P_k are the ends of the shortest line between the two segments. P_m is located on Line 1 and Line 3, and P_k is located on Line 2 and Line 3. By representing each line parametrically and utilizing what is known about Line 3 and points P_m and P_k , a system of equations can be solved to determine the coordinates of points P_m and P_k in terms of the coordinates of points λ_m , γ_m , successively and Q_{ck2} .

Any point P_{Line1} on Line 1 can be represented parametrically as

$$P_{Line1} = \lambda_m + (\gamma_m - \lambda_m) \cdot x_1 \quad (2.49)$$

$$\frac{dP_{Line1}}{dq_j} = \frac{d\lambda_m}{dq_j} + (\gamma_m - \lambda_m) \cdot \frac{dx_1}{dq_j} + \left(\frac{d\gamma_m}{dq_j} - \frac{d\lambda_m}{dq_j} \right) \cdot x_1 \quad (2.50)$$

Any point P_{Line2} on Line 2 can be represented parametrically as

$$P_{Line2} = Q_{ck1} + (Q_{ck2} - Q_{ck1}) \cdot x_2 \quad (2.51)$$

$$\frac{dP_{Line2}}{dq_j} = 0 \quad (2.52)$$

Although the locations of points P_m and P_k are unknown, any point P_{Line3} on Line 3 can be represented in parametric form as

$$P_{Line3} = P_m + (P_k - P_m) \cdot x_3 \quad (2.53)$$

$$\frac{dP_{Line3}}{dq_j} = \frac{dP_m}{dq_j} + (P_k - P_m) \cdot \frac{dt_k}{dq_j} + \left(-\frac{dP_m}{dq_j} \right) \cdot x_3 \quad (2.54)$$

Since Line 3 must be simultaneously perpendicular to Line 1 and Line 2,

$$\overrightarrow{\lambda_m \gamma_m} \circ \overrightarrow{P_m P_k} = 0 \quad (2.55)$$

$$\text{and } \overrightarrow{Q_{ck1} Q_{ck2}} \circ \overrightarrow{P_m P_k} = 0 \quad (2.56)$$

where $\overrightarrow{\lambda_m \gamma_m} = \gamma_m - \lambda_m$, $\overrightarrow{Q_{ck1} Q_{ck2}} = Q_{ck2} - Q_{ck1}$ and $\overrightarrow{P_m P_k} = P_k - P_m$. Since P_m is a point on Line 1, and P_k is a point on Line 2, P_m can be expressed as

$$P_m = \lambda_m + (\gamma_m - \lambda_m) \cdot x_1 \quad (2.57)$$

And its derivative can be determined as Equation (2.50)

and P_k can be expressed as

$$P_k = Q_{ck1} + (Q_{ck2} - Q_{ck1}) \cdot x_2 \quad (2.58)$$

And its derivative can be determined as Equation (2.52)

In Equations (2.57) and (2.58), x_1 and x_2 represent the parameter values for P_m and P_k , respectively. Taking the difference of Equations (2.58) and (2.57) results in

$$P_k - P_m = Q_{ck1} + (Q_{ck2} - Q_{ck1}) \cdot x_2 - \lambda_m - (\gamma_m - \lambda_m) \cdot x_1 \quad (2.59)$$

Now, Equation (2.59) can be substituted into Equations (2.55) and (2.56) such that

$$\overrightarrow{\lambda_m \gamma_m} \circ (\overrightarrow{Q_{ck1} + (Q_{ck2} - Q_{ck1}) \cdot x_2 - \lambda_m - (\gamma_m - \lambda_m) \cdot x_1}) = 0 \quad (2.60)$$

And

$$\overrightarrow{Q_{ck1} Q_{ck2}} \circ (\overrightarrow{Q_{ck1} + (Q_{ck2} - Q_{ck1}) \cdot x_2 - \lambda_m - (P_m - \lambda_m) \cdot x_1}) = 0 \quad (2.61)$$

Collecting terms and putting Equations (2.60) and (2.61) into matrix form leads to

$$\begin{bmatrix} -\overrightarrow{\lambda_m \gamma_m} \circ \overrightarrow{\lambda_m \gamma_m} & \overrightarrow{\lambda_m \gamma_m} \circ \overrightarrow{Q_{ck1} Q_{ck2}} \\ -\overrightarrow{\lambda_m \gamma_m} \circ \overrightarrow{Q_{ck1} Q_{ck2}} & \overrightarrow{Q_{ck1} Q_{ck2}} \circ \overrightarrow{Q_{ck1} Q_{ck2}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \overrightarrow{\lambda_m \gamma_m} \circ (\overrightarrow{\lambda_m} \circ \overrightarrow{Q_{ck1}}) \\ \overrightarrow{Q_{ck1} Q_{ck2}} \circ (\overrightarrow{\lambda_m} \circ \overrightarrow{Q_{ck1}}) \end{bmatrix} \quad (2.62)$$

Or

$$[m] \cdot x = \begin{bmatrix} A \\ B \end{bmatrix} \quad (2.63)$$

Solving Equation (2.63) via Cramer's rule for t leads to

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{\overrightarrow{Q_{ck1}Q_{ck2}} \circ \overrightarrow{Q_{ck1}Q_{ck2}}}{|m|} A - \frac{\overrightarrow{\lambda_m \gamma_m} \circ \overrightarrow{Q_{ck1}Q_{ck2}}}{|m|} B \\ \frac{\overrightarrow{\lambda_m \gamma_m} \circ \overrightarrow{Q_{ck1}Q_{ck2}}}{|m|} A - \frac{\overrightarrow{\lambda_m \gamma_m} \circ \overrightarrow{\lambda_m \gamma_m}}{|m|} B \end{bmatrix} \quad (2.64)$$

Since the parametric equations for Line 1 and Line 2 represent any point on segments through the given points while the cylispheres represented are constrained to the line segments connecting the given points, new parameters must be defined such that:

$$x_{11} = \begin{cases} 0, & \text{if } x_1 < 0 \\ 1, & \text{if } x_1 > 1 \\ x_1, & \text{if } 0 \leq x_1 \leq 1 \end{cases} \quad (2.65)$$

and

$$x_{22} = \begin{cases} 0, & \text{if } x_2 < 0 \\ 1, & \text{if } x_2 > 1 \\ x_2, & \text{if } 0 \leq x_2 \leq 1 \end{cases} \quad (2.66)$$

Now potential coordinates for P_m and P_k can be calculated such that:

$$P_m = \lambda_m + (\gamma_m - \lambda_m) \cdot x_{11} \quad (2.67)$$

$$P_k = Q_{ck1} + (Q_{ck2} - Q_{ck1}) \cdot x_{22} \quad (2.68)$$

A potential minimum distance, d_k , between the robot link and cylisphere is

$$\vec{d}_{k,1} = \vec{P_m P_k} \tag{2.69}$$

and the magnitude of this potential minimum distance, $\|d_{k,1}\|$, is

$$\|d_{k,1}\| = \|\vec{P_m P_k}\| = \sqrt{\vec{P_m P_k} \circ \vec{P_m P_k}} \tag{2.70}$$

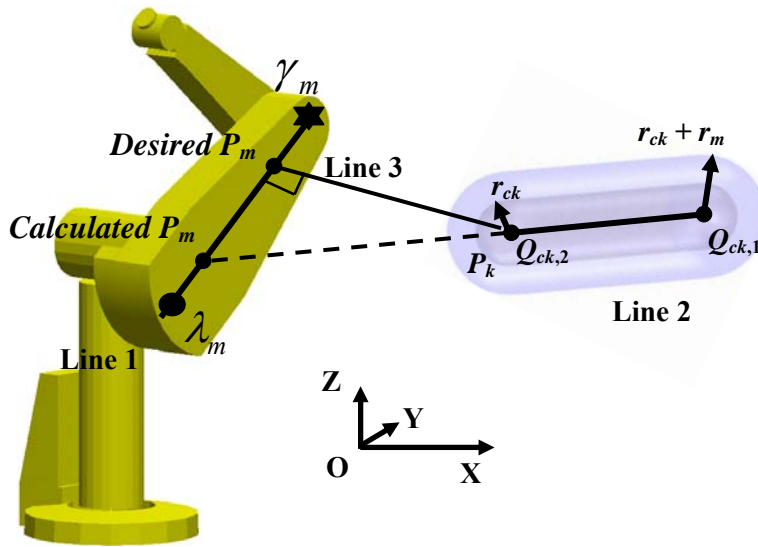


Figure 2.8: Minimum Distance WHEN $x_1 \neq x_{11}$ OR $x_2 \neq x_{22}$.

If $x_1 = x_{11}$ and $x_2 = x_{22}$, the values calculated using Equations (2.69) and (2.70) are correct. However, if $x_1 \neq x_{11}$ or $x_2 \neq x_{22}$, further checks need to be done. Figure (2.8) shows a sample case of when the calculated minimum distance is incorrect. In order to find the coordinates for the desired P_m , the algorithm for calculating the minimum distance between a line segment and a sphere will be used, with Line 1 represents the robot link and point Q_{ck2} represents the sphere. Overall, if $x_1 \neq x_{11}$ or $x_2 \neq x_{22}$, the algorithm for calculating the minimum distance between a line

segment and a sphere must be used with each of the Lines endpoints. Figure (2.9) shows the steps of finding the minimum distance in such cases.

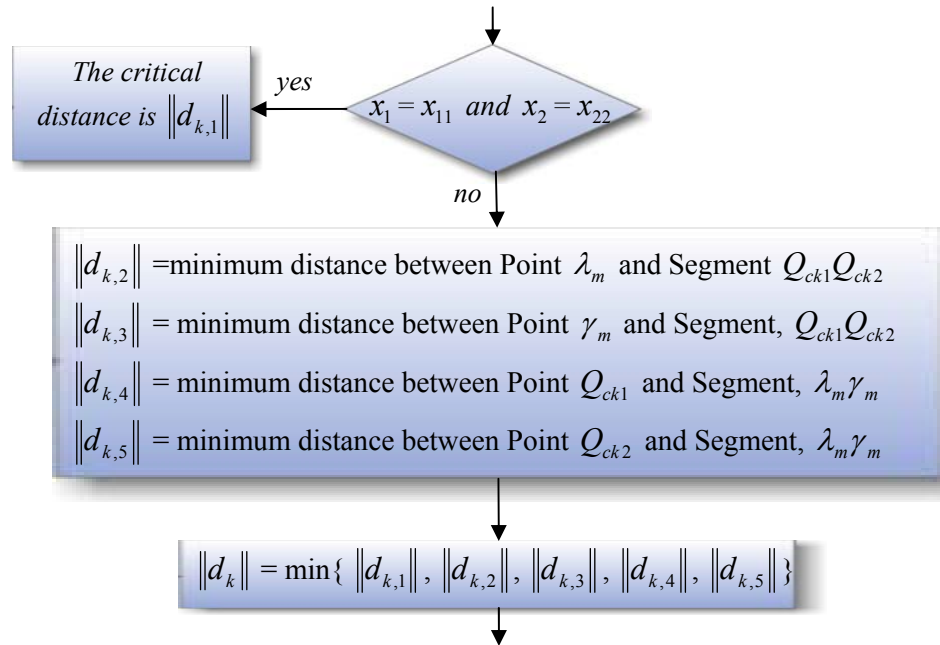


Figure 2.9: The Flow-Chart to find the Shortest Distance Between Robot Link and Cylisphere.

The smallest of the five minimum distance magnitudes calculated $\|d_k\|$ is chosen along with its respective P_m and P_k coordinates. Finally, the radii of the robot link and cylisphere $r_{ck} + r_j$ are subtracted from $\|d_k\|$ to get the true minimum distance magnitude value, Harden 2002.

2.5.3. Quadri-lateral Plane Obstacles

A quadrilateral plane is a basic building block for a wide variety of shapes. It is defined by three points $P_1, P_2,$ and $P_3,$ and a half thickness, $e_p.$ The fourth point of the quadrilateral plane is calculated as

$$P_4 = P_2 + P_3 - P_1 \tag{2.71}$$

The minimum distance between quadrilateral plane and robot links has been calculated as the minimum distance between segment and plane as following:

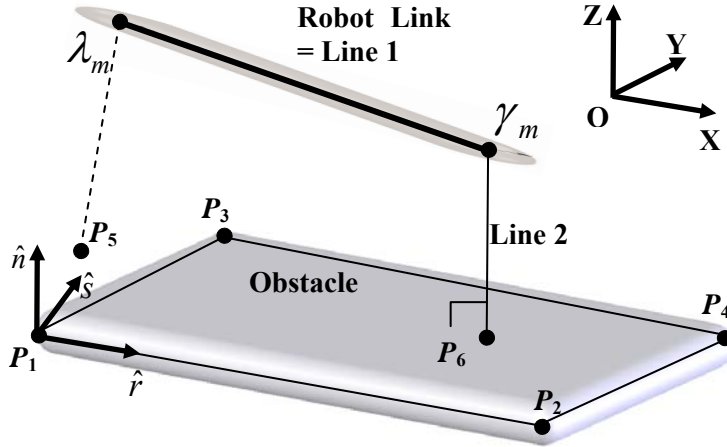


Figure 2.10: Minimum Distance Derivation Between Robot Link and a Quadri-Lateral Plane.

Figure (2.10) shows the picture used to derive the minimum distance between a robot link represented by a line segment and a quadri-lateral plane. This derivation process can also be used for an infinite plane. In the figure, Line 1 represents the robot link. Let's consider the case that the link line segment of the robot is defined by λ_m and γ_m . Points P_{n1} , P_{n2} , P_{n3} and P_{n4} represent the n^{th} zero-thickness quadri-lateral planar surface $QP_n(P_{n1}, P_{n2}, P_{n3}, e_{pn})$. Points λ_m , γ_m , P_{n5} , and P_{n6} are potential observation points, and Line 2 is the desired minimum distance line. The symbols \vec{r} , \vec{s} , and \vec{n} represent a mutually orthogonal set of unit vectors. Here, \vec{r} and \vec{s} are both located in the plane, and \vec{n} is normal to the plane. For the derivation, it is assumed that Line 1 cannot intersect with the Planar Surface because such an intersection indicates a collision, which we are trying to avoid.

The minimum distance between a line segment and an infinite plane is always the difference between one of the line endpoints and the corresponding projection of the same endpoint onto the surface of the plane. If the plane is a quadri-lateral, then checks must be performed to ensure the line endpoint projection is inside the quadri-lateral and that both line endpoints are on the same side of the plane.

For the example shown in Figure (2.10), the first step in determining the minimum distance is to calculate the unit vectors \vec{r} , \vec{s} , and \vec{n} .

The unit vector \vec{r} is calculated as

$$\vec{r} = \frac{P_{n2} - P_{n1}}{\|P_{n2} - P_{n1}\|} \quad (2.72)$$

Then, \vec{n} can be calculated as

$$\vec{n} = \vec{r} \times \frac{P_{n3} - P_{n1}}{\|P_{n3} - P_{n1}\|} \quad (2.73)$$

Finally, \vec{s} can be calculated as

$$\vec{s} = \vec{n} \times \vec{r} \quad (2.74)$$

Once these unit vectors are known, the projections of points λ_m and γ_m onto the plane can be calculated as

$$P_{n5} = (\lambda_m - P_{n1}) \circ \vec{r} + (\lambda_m - P_{n1}) \circ \vec{s} + P_{n1} \quad (2.75)$$

$$\frac{dP_{n5}}{dq_j} = \left(\frac{d\lambda_m}{dq_j} \circ \vec{r} \right) \cdot \vec{r} + \left(\frac{d\lambda_m}{dq_j} \circ \vec{s} \right) \cdot \vec{s} \quad (2.76)$$

and

$$P_{n6} = (\gamma_m - P_{n1}) \circ \vec{r} + (\gamma_m - P_{n1}) \circ \vec{s} + P_{n1} \quad (2.77)$$

$$\frac{dP_6}{dq_j} = \left(\frac{d\gamma_m}{dq_j} \circ \vec{r} \right) \cdot \vec{r} + \left(\frac{d\gamma_m}{dq_j} \circ \vec{s} \right) \cdot \vec{s} \quad (2.78)$$

A potential minimum distance magnitude, $\|d_n\|$, can then be calculated as

$$\|d_n\| = \min \{ \|\lambda_m - P_{n5}\|, \|\gamma_m - P_{n6}\| \} \quad (2.79)$$

$$\frac{d\|\lambda_m - P_{n5}\|}{dq_j} = \left(\frac{1}{\|\lambda_m - P_{n5}\|} \right) \cdot \left(\begin{array}{c} \left[\left(\frac{d\lambda_m}{dq_j} - \frac{dP_{n5}}{dq_j} \right) \cdot (\lambda_m - P_{n5}) \right]_x \\ + \left[\left(\frac{d\lambda_m}{dq_j} - \frac{dP_{n5}}{dq_j} \right) \cdot (\lambda_m - P_{n5}) \right]_y \\ + \left[\left(\frac{d\lambda_m}{dq_j} - \frac{dP_{n5}}{dq_j} \right) \cdot (\lambda_m - P_{n5}) \right]_z \end{array} \right) \quad (2.80)$$

$$\frac{d\|\gamma_m - P_{n6}\|}{dq_j} = \left(\frac{1}{\|\gamma_m - P_{n6}\|} \right) \cdot \left(\begin{array}{c} \left[\left(\frac{d\gamma_m}{dq_j} - \frac{dP_{n6}}{dq_j} \right) \cdot (\gamma_m - P_{n6}) \right]_x \\ + \left[\left(\frac{d\gamma_m}{dq_j} - \frac{dP_{n6}}{dq_j} \right) \cdot (\gamma_m - P_{n6}) \right]_y \\ + \left[\left(\frac{d\gamma_m}{dq_j} - \frac{dP_{n6}}{dq_j} \right) \cdot (\gamma_m - P_{n6}) \right]_z \end{array} \right) \quad (2.81)$$

If the obstacle is an infinite plane, then the result in Equation (2.79) is correct and no further calculation is needed. For the example shown in Figure (2.10), Equation (2.79) gives the result that $\|d_n\| = \|\gamma_m - P_{n6}\|$. Therefore, a check must be made to ensure that the potential obstacle witness point, P_{n6} , is inside the quadri-lateral, $P_{n1}P_{n2}P_{n4}P_{n3}$. The obstacle observation point, P_{n6} , is inside the quadri-lateral if all of the following equations are true,

$$\begin{aligned} ((P_{n2} - P_{n1}) \times (P_{n6} - P_{n1})) \circ \vec{n} &> 0, \\ ((P_{n4} - P_{n2}) \times (P_{n6} - P_{n2})) \circ \vec{n} &> 0, \\ ((P_{n3} - P_{n4}) \times (P_{n6} - P_{n4})) \circ \vec{n} &> 0, \\ ((P_{n1} - P_{n3}) \times (P_{n6} - P_{n3})) \circ \vec{n} &> 0. \end{aligned} \quad (2.82)$$

$$\begin{aligned} ((P_{n2} - P_{n1}) \times (P_{n5} - P_{n1})) \circ \vec{n} &> 0, \\ ((P_{n4} - P_{n2}) \times (P_{n5} - P_{n2})) \circ \vec{n} &> 0, \\ ((P_{n3} - P_{n4}) \times (P_{n5} - P_{n4})) \circ \vec{n} &> 0, \\ ((P_{n1} - P_{n3}) \times (P_{n5} - P_{n3})) \circ \vec{n} &> 0. \end{aligned} \quad (2.83)$$

If all of the Equations (2.83) are satisfied, the obstacle observation point P_{n5} is inside the quadri-lateral. If all of the Equations (2.82) are satisfied, a final check

must be made to ensure that the line endpoints λ_m and γ_m are located on the same side of the plane. This is the case if

$$\frac{(\lambda_m - P_{n5})}{\|\lambda_m - P_{n5}\|} = \frac{(\gamma_m - P_{n6})}{\|\gamma_m - P_{n6}\|} \tag{2.84}$$

If any of the Equations (2.82) are not satisfied, then the minimum distance magnitude calculated using Equation (2.79) is incorrect, and the true minimum distance must be determined using the process for calculating the minimum distance between two line segments, sub-Section (2.5.2). If Equation (2.84) is not satisfied, then the minimum distance calculated may be correct; but must be compared with the four potential minimum distances that can be calculated using the distance calculation algorithm for two line segments, see the block diagram in Figure (2.12). An illustration of this situation is shown in Figure (2.11).

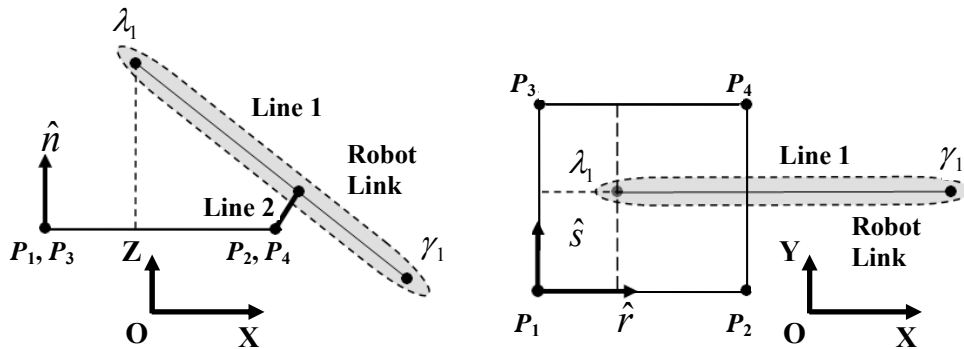


Figure 2.11: Minimum Distance When Line Endpoints Outside Quadri-Lateral.

In the figure, dotted lines represent potential minimum distances that are considered. Line 2 is the actual minimum distance because points λ_m and γ_m are on opposite sides of the plane. Calculation proceeds by treating each edge of the quadri-lateral plane as a line segment and computing four new potential minimum distances. Of all the valid potential minimum distances, the one with the smallest

magnitude is chosen along with its respective witness points. Finally, the radius of the robot link and the thickness of the plane are subtracted from the chosen minimum distance magnitude to get the true minimum distance magnitude value, Harden 2002.

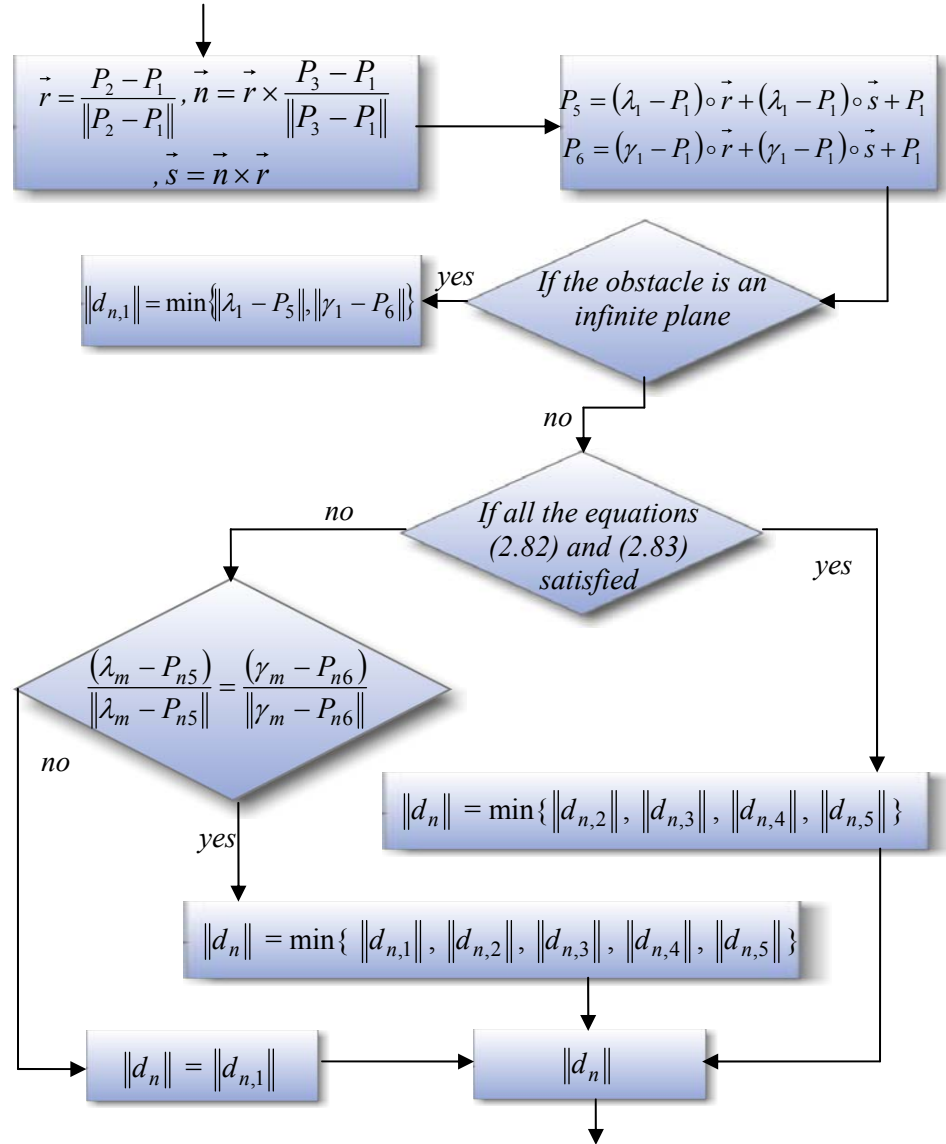


Figure 2.12: The Algorithm to Find the Distance Between Robot Link and Quadri-Lateral.

where:

$\|d_{n,2}\|$ = minimum distance between segments $\lambda_m \gamma_m$ and $P_{n1}P_{n2}$,

$\|d_{n,3}\|$ = minimum distance between segments $\lambda_m \gamma_m$ and, $P_{n2}P_{n3}$,

$\|d_{n,4}\|$ = minimum distance between segments $\lambda_m \gamma_m$ and, $P_{n3}P_{n4}$,

$\|d_{n,5}\|$ = minimum distance between segments $\lambda_m \gamma_m$ and, $P_{n4}P_{n1}$.

CHAPTER 3

PATH PLANNING

A principle problem in robotics, which will concern us in this chapter of this dissertation, is the path planning. We want to devise algorithms that will enable a robot to move from one position to another without any collisions. Path planning is becoming increasingly important in many areas, for example, industrial robotics, autonomous systems, assembly planning and virtual prototyping, Chang and Li 1995, computer graphics simulations, Kuffner and Latombe 2000, and computer-aided drug design, Finn et al. 1997. This chapter deals with the basic path-planning problem for industrial robot moving in a well-defined static environment using genetic algorithms. Path planning deals with the problem of finding motion strategies for movable objects or articulated structures. An articulated structure can be used to model things like, e.g., the motion of a computer-animated character, a robotic manipulator or a complex protein molecule.

The path planning problem; trying to solve in this thesis; is to find a sequence of configurations in which the robot moves from an initial configuration C^i to a goal configuration C^f without colliding with obstacles in the environment.

In path planning problems, the number of feasible paths between the initial and final position of a robot are often very large, and the goal is not necessarily to determine the best solution, but to obtain an acceptable one according to certain requirements and constraints. Various search methods have been developed (e.g., calculus-based methods, enumerative schemes, random search algorithms, etc.) for the robot path-planning problem. In this work, genetic algorithm has been used.

Genetic Algorithm (GA) based search and optimization techniques have recently found an increasing use in machine learning, robot motion planning, scheduling, pattern recognition, image sensing and many other engineering applications. In principle, GAs are search algorithms based on mechanics of natural selection and natural genetics. They combine survival of the most fitting among the string structures with randomized yet structured information exchange to form a search algorithm with innovative flair of natural evolution.

In the proposed method, generating such path is used to minimize the distance between its initial and final configurations. The genetic algorithm (GA) appears here to solve such problem by minimizing the traveling distance of the end-effector and the significant points (Section 2.1)) between the initial and final point avoiding obstacles. The workspace will be modeled in such way to provide a discrete configuration space based on the positions of the end-effector between the initial and final configurations of the robot.

In this procedure, two optimization processes using genetic algorithms are involved. The first one, an optimization process for the obtaining of the adjacent configurations (detailed in Section (3.1)). The order in which the adjacent configurations are generated will condition the Space of Configurations generated and, therefore, the path to be obtained. Second optimization process is used for the obtaining of the path, which consist of a set of adjacent configurations. This

algorithm will be applied on an industrial robot Puma 560 modeled with six degree of freedom.

3.1. ADJACENT CONFIGURATIONS FOR PATH PLANNING

In this section, the process of generating a discrete space of configuration is presented. This space of configurations is based on the obtaining of adjacent configurations concerning kinematics compatibility and feasibility with collision avoidance regardless the dynamics concerns.

3.1.1. Adjacent Configurations Definition

The configuration C^k is adjacent to a given configuration C^P , if they are feasible and the three following conditions are satisfied:

1. The end-effector position γ_4 (see Figure (2.1)) corresponds to a point of the discrete workspace. In addition, it is one increment far from the point corresponding to the C^P configuration, so it is said that, the two configurations are neighboring and there must be a given increment between them less than the smallest obstacle size in the workspace.
2. Verification of the absence of obstacles between adjacent configurations C^k and C^P . Also, to verify that the distance between significant points meet the following condition,

$$\left| \overrightarrow{\gamma_i^p \gamma_i^k} \right| \leq 2 \cdot \min(r_j); \quad i = 1, 2, 3; \quad j = 1, 2, \dots \quad (3.1)$$

where r_j is the minimum characteristic dimension of the obstacles in the workspace.

3. C^k should be such as to minimize the function:

$$\|C^k - C^p\| = A \cdot \sum_{j=1}^4 \left(\begin{array}{c} (\gamma_j^k - \gamma_j^p)_x^2 + \\ (\gamma_j^k - \gamma_j^p)_y^2 + \\ (\gamma_j^k - \gamma_j^p)_z^2 \end{array} \right) + B \cdot \sum_{i=1}^6 (q_i^f - q_i^k)^2 \quad (3.2)$$

where A , B are coefficient and the expression is expressed in Cartesian coordinates, which aims to minimize the distance between significant points and the distance between the joints values of the current configuration and the final global one.

Adjacent configuration for path planning concern in finding a set of via points (intermediate points) that constructed the path. This path can be tracked after that to find an optimal time scaling subjected to the dynamic constraints of the manipulator.

3.1.2. Workspace Discretization

The first step of the optimization process is generating a discreet space. See Section (2.4.1) for more details about the workspace discretization.

3.1.3. Obtaining The Configuration C^k

In the building process of the path, a random search procedure will be applied to search from the C^i for the next adjacent configuration and so on until it reaches the C^f . The main concern in this part is finding a sequence of robot configurations between the initial and final configurations that fulfils the early listed three conditions. A methodology of two distinct routines has been constructed to obtain a robot configuration C^k adjacent to C^P . In first place, the inverse kinematic problem; explained in 2.2.3); will be used to find the C^k for a given γ_4 . If the new configuration C^k doesn't fulfill the condition, a genetic algorithm procedure will be used to solve the problem.

Genetic algorithm maintains a population of solutions or individuals throughout the search. It initializes the population with a pool of potential solutions to the problem and seeks to produce better solutions, by combining the better of the existing ones through the use of genetic operators. Individuals are selected at each iteration through a selection scheme depends on the fitness or the objective function value for each individual.

A Steady State Genetic Algorithm (SSGA) procedure is used to obtain a robot configuration C^k adjacent to a given one C^P considering the three conditions mentioned previously. A SSGA uses overlapping populations. This means, the ability to specify how much of the population should be replaced in each generation. Newly generated offspring are added to the population, and then the worst individuals are destroyed (so the new offspring may or may not make it into the population, depending on whether they are better than the worst in the population).

- Chromosome

The individual or the chromosome represents the robot configuration. Each chromosome consists of six genes; the robot generalized coordinates ($q_i; i = 1, 2, \dots, 6$).

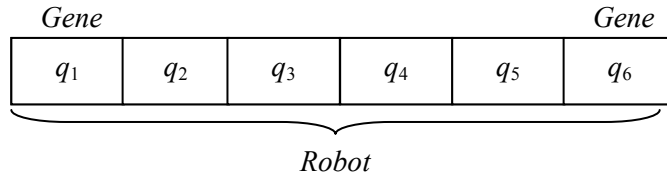


Figure 3.1: Adjacent Configuration for Path Planning GA Chromosome.

The initial population consists of a defined number of chromosomes. The initial values of each gene in the chromosome are selected randomly between the two limits of the generalized coordinates for that gene. For example:

$$gene(i) = RV(q_{i,\min}, q_{i,\max}); \quad i = 1 \rightarrow 6 \tag{3.3}$$

where RV = Random Value (between low and high).

In fact, this way of generating the genes value and then checking the validity of the resulting chromosome is computationally expensive. To improve that, by looking at the workspace modeling and the conditions to produce adjacent configurations, it will be concluded that the movement between the two configurations is small (less then the robot width). Because of that, the previous Equation can be modified. Consider that q_i^p is the given configuration and Δq is a small increment. Therefore, the new interval for each q can be calculated, as the next flow chart indicates:

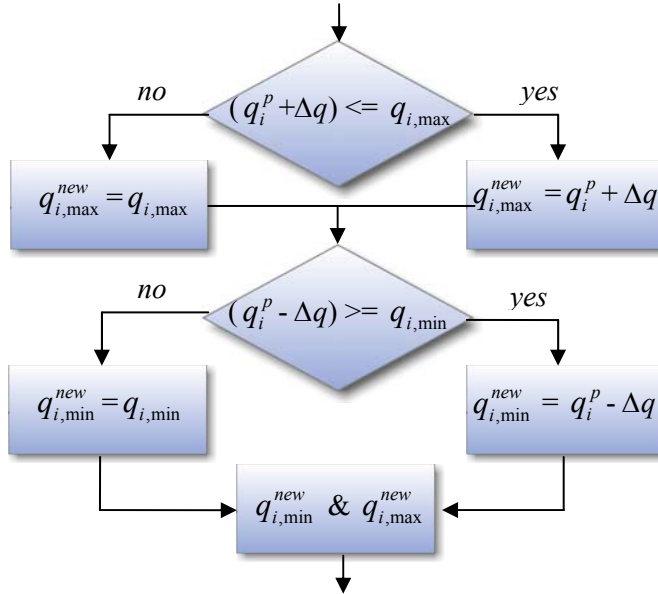


Figure 3.2: Flow-Chart Indicates How to calculate the q_i Intervals.

This means that the Equation (3.3) will be as follows:

$$gene(i) = RV(q_{i,\min}^{new}, q_{i,\max}^{new}); \quad i = 1 \rightarrow 6 \quad (3.4)$$

where RV = Random Value (between low and high).

- Selection:

A roulette-wheel selection method is applied to select individuals for crossover and mutation. This method is based on the magnitude of the fitness score of an individual relative to the rest of the population. The higher score, the more likely an individual will be selected.

- Crossover:

The crossover operator defines the procedure for generating a child from two selected parents. A single point crossover used in this procedure, see Figure (3.3).

- Mutation:

The mutation operator defines the procedure for mutating each genome. In this procedure, an offspring will be selected randomly, then a gene will be selected randomly from that offspring. This gene will be mutated with respect to the following equation.

$$gene(i) = gene(i) + RV(q_{i,\min}, q_{i,\max}) \times [RV(q_{i,\min}, q_{i,\max}) - RV(q_{i,\min}, q_{i,\max})] \quad (3.5)$$

where RV = Random Value (between low and high), $i = 1 \rightarrow 6$.

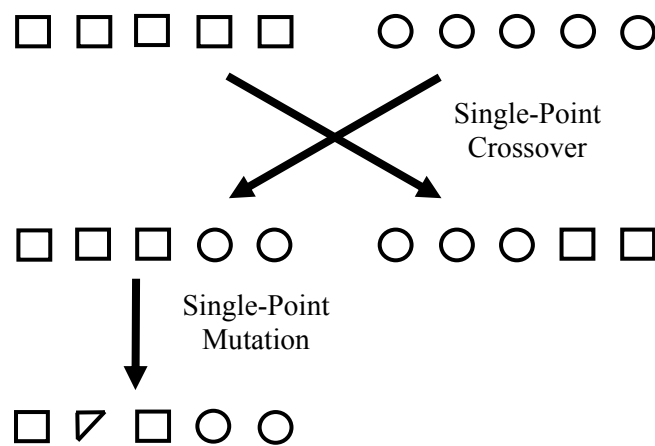


Figure 3.3: Adjacent Configuration Crossover and Mutation.

- Objective:

Minimize Equation (3.2).

3.2. GA PROCEDURE FOR PATH PLANNING

The search technique consists of generating an initial population of strings at random. Each solution is assigned a numerical evaluation of its fitness by an

objective function, which is a mathematical function that maps a particular solution on a single positive number, that is a measure of the solution's worth. During each iteration (generation), each individual string in the current population is evaluated using this measure of fitness. New strings (children) for the next generation are selected from the current population of strings (parents) by a process known as "selection". A random selection process is used with a higher probability given for strings with higher fitness values. Such selection scheme systematically eliminates low-fitness individuals from the population of one generation to the next. New generations can be produced either synchronously, so that the old generation is completely replaced, or asynchronously, in which the generations overlap.

The genetic algorithm for path planning uses parallel populations with migration technique. The genetic algorithm has multiple, independent populations. It creates the populations by cloning the genome or population that you pass when you create it. Each population evolves using steady-state genetic algorithm, but at each generation, some individuals migrate from one population to another. The migration algorithm is deterministic stepping-stone; each population migrates a fixed number of its best individuals to its neighbor. The master population is updated each generation with best individual from each population.

Two genetic operators, crossover and mutation, are probabilistically applied to create a new population of individuals. Parent individuals are selected as candidates for crossover or mutation using the roulette-wheel selection method. Genetic algorithms are domain independent because they require no explicit notion of a neighborhood. Hence, crossover and mutation may not always produce feasible solutions. Therefore, the feasibility of a newly created individual is ascertained before inserting it in the population to replace a parent string.

In the GA based solution procedure, a number of new individuals are created at each iteration. The remaining individuals are obtained by deterministically copying the individuals with the top fitness from the previous generation.

3.2.1. Genetic Algorithms Operators and Parameters

The main operators and characteristics in the exposed GA are:

- Individual:

The individual or the chromosome is composed of set of intermediate points (end-effector positions) including end points (initial and final position of the end effector). This means that each chromosome represent a complete path between initial and final configurations. Each triplet cells comprising one point (the Cartesian coordinates of the end-effector) in the chromosome and considered as a gene, Figure (3.4).

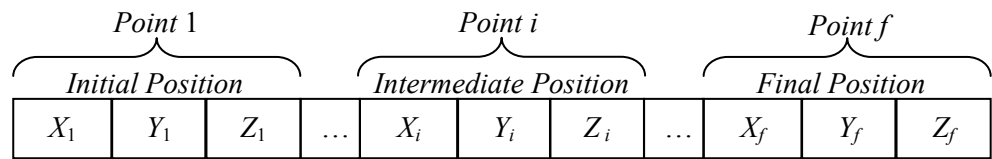


Figure 3.4: Path Planning GA Chromosome.

The first point of each individual is the initial position of the end-effector of C^i . The second point will be selected randomly from the discretized workspace without repetition in one of seven directions: X -direction, Y -direction, Z -direction, XY -direction, XZ -direction, YZ -direction, and XYZ -direction. This strategy will be repeated for the next point and so on until the goal position is achieved. Note that this definition is based on the number of intermediate points that constitute the

path, which means that paths do not have equal lengths, which leads to more complexity in crossover and mutation.

- Objective Function:

The objective of this optimization problem is to find the optimal path between initial and final positions of a robot end-effector. Because of the possibility of existing obstacles, and geometric constraints, the algorithm will try to find the shortest possible path. The shortest path will be calculated by minimizing the sum of the straight-line segments of the corresponding significant points of the robot, from the initial to the final point. In this case, the objective of GA is to minimize the following equation:

$$\text{Minimize} \left\{ \sum_{i=1}^{n-1} \sum_{j=1}^m \sqrt{(\gamma_j^{i+1} - \gamma_j^i)_x^2 + (\gamma_j^{i+1} - \gamma_j^i)_y^2 + (\gamma_j^{i+1} - \gamma_j^i)_z^2} \right\} \quad (3.6)$$

where: j is the number of the significant points of the robot, and $m = 4$ for Puma 560 robot; the case demonstrated in this thesis. $i = 1, 2, \dots, n$ is the number of robot configurations included in the path.

- Selection:

The selection operation is made using the roulette-wheel method.

- Crossover:

The crossover is made through the exchange of a part of the path (chromosome) between two selected paths through the selection operation mentioned earlier; being that, it is executed only if the probability of the crossover is satisfied. This is done by searching groups of individuals that have been selected for crossover, and then, select pair of individuals randomly. In each pair, the algorithm searches the genes of each individual for the intersection configurations.

The intersection in this case is to find a (C_j^p) configuration p in path i that can be adjacent to a (C_j^k) configuration k in the path j . The search for the adjacent configuration occurs in the positive direction. The algorithm looks for all possible intersections between two selected chromosomes (paths) for crossover. I.e., given two paths: *Dad* with length n and *Mom* with length m .

$$Dad = C_{Dad}^1 \cup C_{Dad}^2 \cup \dots \cup C_{Dad}^i \cup \dots \cup C_{Dad}^n \quad (3.7)$$

$$Mom = C_{Mom}^1 \cup C_{Mom}^2 \cup \dots \cup C_{Mom}^j \cup \dots \cup C_{Mom}^m \quad (3.8)$$

$$Dad \cap Mom = \left\{ (C_{Dad}^k, C_{Mom}^p)_1, (C_{Dad}^k, C_{Mom}^p)_2, \dots, (C_{Dad}^k, C_{Mom}^p)_{l_1} \right\} \quad (3.9)$$

$$Mom \cap Dad = \left\{ (C_{Mom}^k, C_{Dad}^p)_1, (C_{Mom}^k, C_{Dad}^p)_2, \dots, (C_{Mom}^k, C_{Dad}^p)_{l_2} \right\} \quad (3.10)$$

where (C_{Dad}^k, C_{Mom}^p) are adjacent configurations, $l_1 = 0, 1, 2, \dots, n-2$ in case of *Dad*. $l_2 = 0, 1, 2, \dots, m-2$ in case of *Mom* number of adjacent configurations found.

This way of intersection means that $Mom \cap Dad$ and $Dad \cap Mom$ are not necessary to be equal, which leads to the possibility to produce only one offspring rather than two in some cases.

The algorithm then will select one intersection randomly in case of many are found satisfying these criteria. The new offspring (path) will be:

$$Offspring1 = sis = C_{Dad}^1 \cup C_{Dad}^2 \cup \dots \cup C_{Dad}^i \cup C_{Mom}^j \cup \dots \cup C_{Mom}^m \quad (3.11)$$

$$Offspring2 = bro = C_{Mom}^1 \cup C_{Mom}^2 \cup \dots \cup C_{Mom}^j \cup C_{Dad}^i \cup \dots \cup C_{Dad}^n \quad (3.12)$$

Note: If there are no such points, the crossover will be cancelled.

This means that the resulting path (offspring) will consist of two parts: a part from *Dad* (from the initial configuration until the selected Configuration C^i), and a part from *Mom* (from C^j until the final configuration). This crossover method doesn't need equal chromosomes lengths. This process is illustrated in 2-D in Figure (3.5).

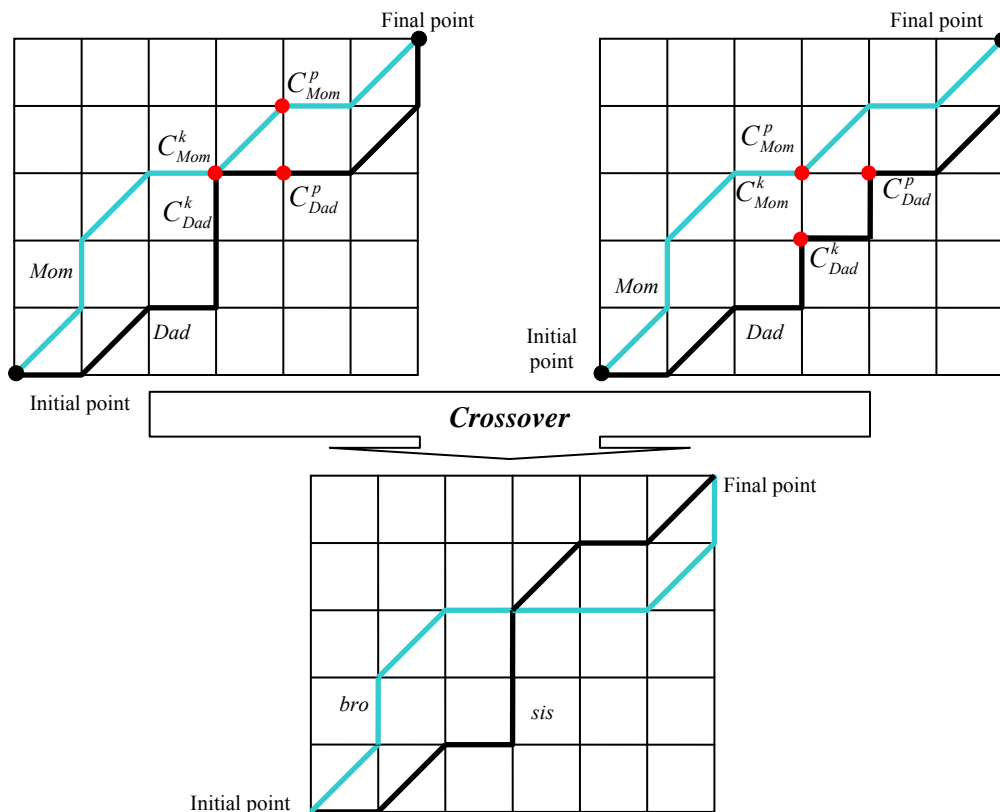


Figure 3.5: Crossover Between Two Robot Paths.

- Mutation:

Mutation is done by selecting a configuration (gene) randomly from a selected path (chromosome). The first and the final configurations are not

considered for mutation. The configuration is then compared to the previous and next configurations in the path. All the possible changes with which the path will remain incremental and quantum are applied to the configuration. To illustrate, let's consider three consecutive robot configurations C^{i-1} , C^i , C^{i+1} (three consecutive genes) in which their end-effector have the positions $(0, 0, 0)$, $(1, 0, 1)$, $(1, 1, 2)$ with a step value of 1 in the x , y and z -coordinates. If mutation is to be applied on the C^i , where its end-effector position lies at $(1, 0, 1)$, the algorithm will consider how each of the coordinates changed. The x -coordinate changed from 0 (previous position) to 1 and remained 1 in the next position. It is clear that changing the x -coordinate from 1 to 0 will not affect the validity of the path since the positions will become $(0, 0, 0)$, $(0, 0, 1)$, $(1, 1, 2)$; i.e. x -coordinate changed from current position to next, while remains the same when going from the previous position to the current one. The same thing can be said about the y -coordinate, since it has not changed when going from the previous position to the current one, while changed when going to the next position. The mutation will cause the y -coordinate to change from 0 to 1. Finally, the z -coordinate cannot be modified since it changed from 0 to 1 to 2. If the mutation would change the z -coordinate to 0 or 2, the step would be greater than the predefined step. The mutation will not affect the coordinates that has not changed at all. For example, the x -coordinate in $(0,0,0),(0,0,1),(0,1,1)$ since any changes will result invalid path. For this new position, the adjacent configuration algorithm will take places to move the robot from the position $(0, 0, 0)$ to $(0, 0, 1)$ and then to $(1, 1, 2)$. This process is illustrated in Figure (3.6).

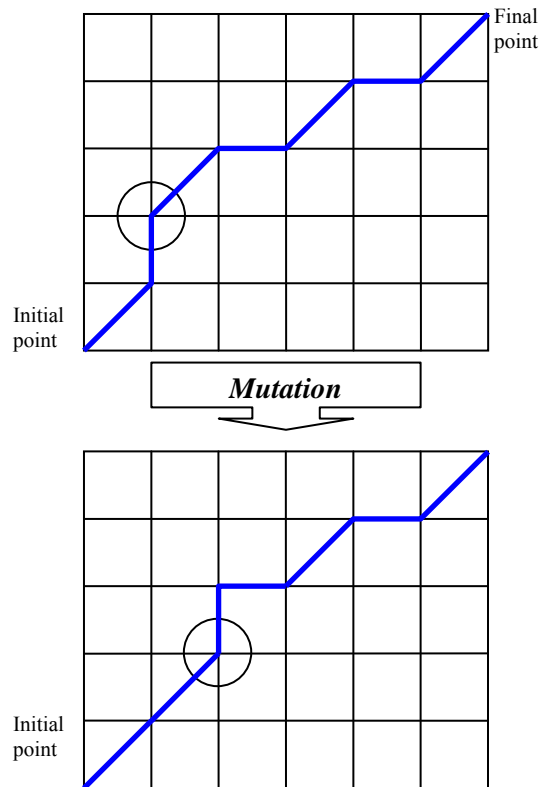


Figure 3.6: Path GA Mutation.

3.3. TRAJECTORY PLANNING: INDIRECT METHOD

As mentioned before, in the introduction, one of the approaches deals with trajectory planning is the indirect or decoupled approach. Indirect approaches firstly seek for a path in the configuration space, and then the trajectory adjusts; subjected to the dynamic constraints of the manipulator, see Saramago and Steffen 2001, Valero et al. 2006 for more details. Indirect approaches are the most widely used in path planning (For depth knowledge you should refer to Piazzzi and Visioli 1997a, 2000, Saramago and Steffen 2001, Plessis and Snyman 2003, Behzadipour

and Khajepour 2006, Valero et al. 2006, Bertolazzi et al. 2007, Gasparetto and Zanotto 2007).

In next Chapter 4 of this thesis, the trajectory-planning problem will be discussed in details. However, in this section, after the path-planning problem has been solved by the mentioned procedure in the previous section, the trajectory can be adjusted by finding an optimal time scaling for the path subjected to the dynamic constraints of the manipulator. To achieve that, the clamped cubic spline (the time optimizer algorithm) explained in next section can be used.

3.4. TIME OPTIMIZER

A genetic algorithm procedure is fed by a path (sequence of configurations) obtained (from previous section), its aim is to schedule the time intervals between two adjacent configurations such that the total traveling time is minimized using Clamped Cubic Spline subjected to: (1) Physical constraints on joint velocities, accelerations, and jerks. (2) Dynamic constraints on actuators torques, powers, and energies.

3.4.1. Formulation of Cubic Polynomial Joint Trajectory

The philosophy of splining is to use low order polynomials to interpolate from grid point to grid point. This is ideally suited when one has control of the grid locations and the values of data being interpolated. As this control is dominated, the relative accuracy can be controlled by changing the overall space between the grid points.

Cubic splines are the lowest order polynomial endowed with inflection points. If one would think about interpolating a set of data points using parabolic (quadratic) functions without inflection points, the interpolation would be meaningless.

The formulation of the cubic spline is based on the n joint vectors (n configurations) that construct the joint trajectory. Joint vectors are denoted as q_i^j which represents the position of the joint i with respect to configuration j . The cubic polynomial trajectory is then constructed for each joint to fit the joint sequence $q_i^0, q_i^1, \dots, q_i^n$. Let $t_0 < t_1 < \dots < t_{n-2} < t_{n-1} < t_n$ be an ordered time sequence, at time $t = t_j$ the joint position will be q_i^j . Let $q_i^j(t)$ be a cubic polynomial function defined on the time interval $[t_j, t_{j+1}]$; $0 \leq j \leq n-1$. The problem of trajectory interpolation is to spline $q_i^j(t)$, for $j = 0, 1, 2, \dots, n-1$, together such that the required displacement, velocity and acceleration are satisfied; and the displacement, velocity and acceleration are continuous on the entire time interval $[t_0, t_n]$.

Consider the cubic spline function as follows:

$$S(t) = \begin{cases} q_i^0(t) & \text{if } t_0 \leq t < t_1 \\ q_i^1(t) & \text{if } t_1 \leq t < t_2 \\ \vdots & \\ q_i^{n-1}(t) & \text{if } t_{n-1} \leq t < t_n \end{cases} \quad (3.13)$$

The cubic spline function $S(t)$ satisfies these properties:

- 1) $S(t)$ will interpolate all data points.

$$2) \begin{cases} S(t) & \text{continuous on } [t_0, t_n] \\ \dot{S}(t) & \text{continuous on } [t_0, t_n] \\ \ddot{S}(t) & \text{continuous on } [t_0, t_n] \end{cases} \quad (3.14)$$

$$3) \begin{cases} q_i^j(t_{j+1}) = q_i^{j+1}(t_{j+1}) \\ \dot{q}_i^j(t_{j+1}) = \dot{q}_i^{j+1}(t_{j+1}) \\ \ddot{q}_i^j(t_{j+1}) = \ddot{q}_i^{j+1}(t_{j+1}) \end{cases} \quad j = 0, 1, \dots, n-2 \quad (3.15)$$

This means that $S(t)$ (represents robot trajectory) is presented by cubic polynomials, each one has 4 coefficients, and all its derivatives (represents robot joint velocities and accelerations) are continuous for any time t in the open interval (t_j, t_{j+1}) .

This results in a matrix of $n - 1$ equations and $n + 1$ unknowns. The two remaining equations are based on the border conditions for the starting point $s_0(t_0)$, and end point $s_{n-1}(t_n)$. One of the following border conditions can be used.

- a) Free or Natural splines: The second order derivatives of the splines at the end points are zero.

$$\ddot{q}_i^0(t_0) = \ddot{q}_i^n(t_n) = 0 \quad (3.16)$$

- b) Parabolic runout splines: The second order derivatives of the splines at the end points are the same as at the adjacent points. The result is that the curve becomes a parabolic curve at the end points.

$$\begin{aligned} \ddot{q}_i^0(t_0) &= \ddot{q}_i^1(t_1) \\ \ddot{q}_i^n(t_n) &= \ddot{q}_i^{n-1}(t_{n-1}) \end{aligned} \quad (3.17)$$

- c) Cubic runout splines: The curve degrades to a single cubic curve over the last two intervals by setting the second order derivative of the splines at the end points to:

$$\begin{aligned}\ddot{q}_i^0(t_0) &= 2 \cdot \ddot{q}_i^1(t_1) - \ddot{q}_i^2(t_2) \\ \ddot{q}_i^{n-1}(t_n) &= 2 \cdot \ddot{q}_i^{n-2}(t_{n-1}) - \ddot{q}_i^{n-3}(t_{n-2})\end{aligned}\quad (3.18)$$

- d) Clamped spline: The first order derivative of the splines at the end points are set to known values.

$$\begin{cases} \dot{q}_i^0(t_0) = \dot{q}_0 \\ \dot{q}_i^{n-1}(t_n) = \dot{q}_i^n \end{cases} \text{ in this case } \begin{cases} \dot{q}_i^0(t_0) = \dot{q}_i^{n-1}(t_n) = 0 \end{cases}\quad (3.19)$$

This algorithm will be applied to an industrial robot (In this case Puma 560) which means that the starting and ending velocities in the application examples will be *ZERO*. So the border condition used is clamped spline, from more details see Henrici 1982, Press et al. 1992.

Construction: Apply above conditions:

$$\begin{aligned}q_i^j(t) &= a_i^j + b_i^j(t - t_j) + c_i^j(t - t_j)^2 + d_i^j(t - t_j)^3 \\ &; j = 0, 1, \dots, n - 1\end{aligned}\quad (3.20)$$

$$q_i^j(t_j) = a_i^j = q_i^j \quad (3.21)$$

$$\begin{aligned}a_i^{j+1} &= q_i^{j+1} = q_i^j(t_{j+1}) \\ &= a_i^{j+1} + b_i^j(t_{j+1} - t_j) + c_i^j(t_{j+1} - t_j)^2 + d_i^j(t_{j+1} - t_j)^3 \\ &; j = 0, 1, \dots, n - 2\end{aligned}\quad (3.22)$$

$$\text{Let } \begin{cases} h_j = t_{j+1} - t_j & j = 0, 1, \dots, n-1 \\ a_i^n = q_i^{n-1}(t_n) = q_i^n \\ b_i^n = \dot{q}_i^{n-1}(t_n) \\ c_i^n = \frac{\ddot{q}_i^{n-1}(t_n)}{2} \end{cases} \quad (3.23)$$

$$a_i^{j+1} = a_i^j + b_i^j h_j + c_i^j h_j^2 + d_i^j h_j^3 \quad j = 0, 1, \dots, n-1 \quad (3.24)$$

$$\dot{q}_i^j(t) = b_i^j + 2 \cdot c_i^j (t - t_j) + 3 \cdot d_i^j (t - t_j)^2 \quad \Rightarrow \quad \dot{q}_i^j(t_j) = b_i^j \quad (3.25)$$

$$b_i^{j+1} = b_i^j + 2 \cdot c_i^j h_j + 3 \cdot d_i^j h_j^2 \quad j = 0, 1, \dots, n-1 \quad (3.26)$$

$$\ddot{q}_i^j(t) = 2 \cdot c_i^j + 6 \cdot d_i^j (t - t_j) \quad \Rightarrow \quad \ddot{q}_i^j(t_j) = 2 \cdot c_i^j \quad (3.27)$$

$$c_i^{j+1} = c_i^j + 3 \cdot d_i^j h_j ; j = 0, 1, \dots, n-1 \quad (3.28)$$

Solve for d_j in Equation (3.28) and substitute into Equations (3.24) and (3.26) to get:

$$a_i^{j+1} = a_i^j + b_i^j h_j + \frac{h_j^2}{3} (2 \cdot c_i^j + c_i^{j+1}) \quad j = 0, 1, \dots, n-1 \quad (3.29)$$

$$b_i^{j+1} = b_i^j + h_j (c_i^j + c_i^{j+1}) \quad ; \quad j = 0, 1, \dots, n-1 \quad (3.30)$$

Then solve for b_j in Equation (3.29)

$$b_i^j = \frac{1}{h_j}(a_i^{j+1} - a_i^j) - \frac{h_j}{3}(2 \cdot c_i^j + c_i^{j+1}) \quad (3.31)$$

$$\therefore b_i^{j-1} = \frac{1}{h_{j-1}}(a_i^j - a_i^{j-1}) - \frac{h_{j-1}}{3}(2 \cdot c_i^{j-1} + c_i^j) \quad (3.32)$$

Substitute b_j and b_{j-1} into Equation (3.30): gives

$$\begin{aligned} h_{j-1}c_i^{j-1} + 2 \cdot (h_{j-1} + h_j)c_i^j + h_jc_i^{j+1} \\ = \frac{3}{h_j}(a_i^{j+1} - a_i^j) - \frac{3}{h_{j-1}}(a_i^j - a_i^{j-1}) \end{aligned} \quad (3.33)$$

$j = 0, 1, \dots, n-1$

Since h_j and $a_j = q_i(t_j)$ (the robot configuration at each intermediate point of the trajectory) are known. Moreover, the first order derivatives of the splines at the end points (represent the initial configuration C^i of the robot and the final one C^f) are set to zero, Equation (3.19). The system of equation will be in matrix form like:

$$Ax = b \quad (3.34)$$

where is A is $(n+1) \times (n+1)$ matrix.

$$A = \begin{bmatrix} 2 \cdot h_0 & h_0 & 0 & 0 & 0 & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & 0 & 0 & 0 \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & 0 & 0 & 0 & h_{n-1} & 2 \cdot h_{n-1} \end{bmatrix} \quad (3.35)$$

$$x = \begin{bmatrix} c_i^0 \\ c_i^1 \\ \vdots \\ c_i^n \end{bmatrix} \quad b = \begin{bmatrix} \frac{3}{h_0}(a_i^1 - a_i^0) - 3 \cdot \dot{q}_i^0(t_0) \\ \frac{3}{h_1}(a_i^2 - a_i^1) - \frac{3}{h_0}(a_i^1 - a_i^0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_i^n - a_i^{n-1}) - \frac{3}{h_{n-2}}(a_i^{n-1} - a_i^{n-2}) \\ 3 \cdot \dot{q}_i^{n-1}(t_n) - \frac{3}{h_{n-1}}(a_i^n - a_i^{n-1}) \end{bmatrix} \quad (3.36)$$

$$\text{where } \begin{cases} \dot{q}_i^0(t_0) = 0 \\ \dot{q}_i^{n-1}(t_n) = 0 \end{cases}$$

3.4.2. Optimization of Cubic Polynomial Joint Trajectory

For industrial applications, the speed of operation affects the productivity. To maximize the speed of operation, the traveling time for the robot should be minimized. Thus, the optimization problem is to adjust the time intervals between each pair of adjacent configurations such that the total traveling time is minimum. That is, the problem is to determine a set of optimum values for time intervals t_1, t_2, \dots, t_{n-1} . Note that there are N joints that must be considered simultaneously. A GA procedure with parallel populations with migration technique has been implemented to optimize the time intervals needed to move the robot between adjacent configurations in the pursued trajectory. The GA operators for this procedure are as follows:

- Chromosome:

The individual or the chromosome consists of set of genes. Each gene contains a real number represents the time interval. Number of genes in each chromosome is varied, depends on the length of the fed trajectory.

The value of each gene is selected randomly from an interval. The interval limits are 0 and $t_{j,\max}$. $t_{j,\max}$ is set by the user or obtained from the adjacent configuration algorithm for trajectory planning explained in next Chapter, Section (4.1). The value of $t_{j,\max}$ will change in each generation depending on the new generated offsprings.

- Selection:

A roulette-wheel selection method is applied to select individuals for crossover and mutation.

- Crossover:

The crossover operator defines the procedure for generating a child from two selected parents. In this procedure, an arbitrary number “*arr*” should be calculated.

$$arr_j = RV(0, t_{j,\max}) * [RV(0, t_{j,\max}) - RV(0, t_{j,\max})] \quad (3.37)$$

where RV = Random Value (between low and high).

After that, the genes of the new offspring will be calculated by mixing the parents genes.

$$gene_j^{Bro} = arr_j \times gene_j^{Dad} + (1 - arr_j) \times gene_j^{Mom} \quad (3.38)$$

$$gene_j^{Sis} = arr_j \times gene_j^{Mom} + (1 - arr_j) \times gene_j^{Dad} \quad (3.39)$$

where $j = 1, 2, \dots, n - 1$.

- Mutation:

The mutation operator defines the procedure for mutating each genome. In this procedure, an offspring will be selected randomly then a gene j will be selected randomly from that offspring. The mutation will occur with respect to the following equation.

$$gene_j = gene_j + RV(0, t_{j,max}) \times [RV(0, t_{j,max}) - RV(0, t_{j,max})] \quad (3.40)$$

where RV = Random Value (between low and high).

3.5. APPLICATION EXAMPLES

In this Section, application examples “particularized” for robot Puma 560 have been implemented and analyzed to validate the efficiency of the proposed algorithms. Two groups of examples will be demonstrated; one to verify the path-planning algorithm, while the other one to verify the time optimizer algorithm.

The introduced procedure has been executed using a computer with Intel Xeon CPU E5440 @ 2.83 GHz, 8 GB of RAM.

3.5.1. Path Planning Procedure Examples

Many examples have been executed to verify the path-planning algorithm. These examples have different initial and final configurations with different types and quantities of obstacles in the workspace.

Four operational parameters have been studied when the procedure was applied to a numerous different examples. The parameters are:

- a) The objective function: minimize the summation of significant points traveling distance, Equation (3.6), denoted by “ d_s ”.
- b) End-effector traveling distance, denoted by “ d_e ”.
- c) Computational time, denoted by “ t_{c1} ” for path planning algorithm and “ t_{c2} ” for time optimizer algorithm:
- d) Finally, the execution time: The minimum time produced by the Time Optimizer Algorithm to adjust a trajectory on the produced path, denoted by “ t_e ”.

3.5.1.1. Example 1: Comparison with Rubio et al. 2009a

This example demonstrates the effectiveness of the mentioned algorithm. This example has been solved by Rubio et al. 2009a. Thus, a comparison results will be done.

The robot initial and final configurations are shown in Table (3.1). Obstacles are shown in Table (3.2), these obstacles are used to create 10 different environments, starting with the case without obstacles and then the cases of 1, 2, 3 obstacles for each obstacle type.

Joint No.	1	2	3	4	5	6
Initial configuration	59.09°	-145.38°	13.03°	1.13°	31.68°	0.00°
Final configuration	-34.65°	- 169.14°	58.56°	0.00°	15.78°	0.00°

Table 3.1: Initial and Final Configurations for Example 1.

	1st Spherical obstacle	2nd Spherical obstacle	3rd Spherical obstacle
Centre	$C_1^{SO} =$ (-0.85, -0.40, 0.50)	$C_2^{SO} =$ (-0.75, 0.00, 0.50)	$C_3^{SO} =$ (-0.60, 0.20, 0.30)
Radius	$r_1^{SO} = 0.15$	$r_2^{SO} = 0.15$	$r_3^{SO} = 0.15$
	1st Cylindrical obstacle	2nd Cylindrical obstacle	3rd Cylindrical obstacle
Centre 1	$C_1^{Cyl,1} =$ (-0.85, -0.5, 0.0)	$C_2^{Cyl,1} =$ (-0.75, 0.0, 0.0)	$C_3^{Cyl,1} =$ (-0.7, 0.2, 0.0)
Centre 2	$C_1^{Cyl,2} =$ (-0.85, -0.5, 2.0)	$C_2^{Cyl,2} =$ (-0.75, 0.0, 2.0)	$C_3^{Cyl,2} =$ (-0.7, 0.2, 2.0)
Radius	$r_1^{Cyl} = 0.15$	$r_2^{Cyl} = 0.15$	$r_3^{Cyl} = 0.15$
	1st Prismatic obstacle	2nd Prismatic obstacle	3rd Prismatic obstacle
Point 1	$P_{11} =$ (-0.7, -0.35, 0.0)	$P_{21} =$ (-0.5, 0.0, 0.0)	$P_{31} = (-0.5, 0.3, 0.0)$
Point 2	$P_{12} =$ (-0.7, -0.35, 2.0)	$P_{22} =$ (-0.5, 0.0, 2.0)	$P_{32} = (-0.5, 0.3, 2.0)$
Point 3	$P_{13} =$ (-1.5, -0.35, 2.0)	$P_{23} =$ (-1.3, 0.0, 2.0)	$P_{33} = (-1.3, 0.3, 2.0)$
Point 4	$P_{14} =$ (-1.5, -0.35, 0.0)	$P_{24} =$ (-1.3, 0.0, 0.0)	$P_{34} = (-1.3, 0.3, 0.0)$

Table 3.2: Obstacles Locations (in m) for Example 1.

The next Figures (3.7), (3.8), and (3.9) show the robot in the initial configuration for three different runs for the same example with different environments. It's shown clearly in the figures that the workspace dimensions can be modified as needed. This is considered as one of the effectiveness of this algorithm.

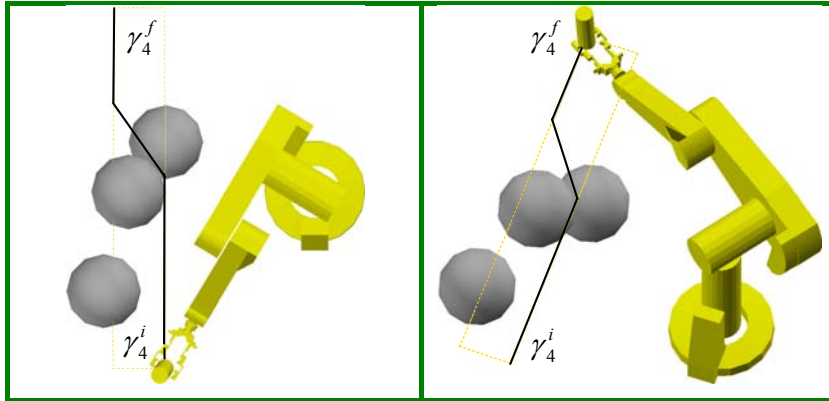


Figure 3.7: Example 1 – The Case of Three Spherical Obstacles.

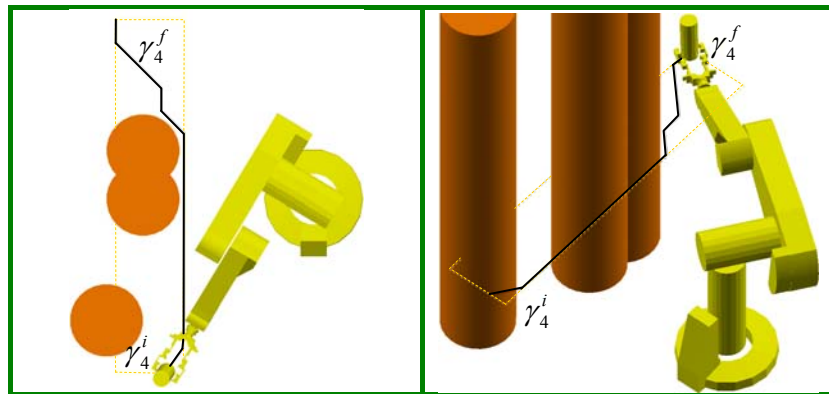


Figure 3.8: Example 1– The Case of Three Cylindrical Obstacles.

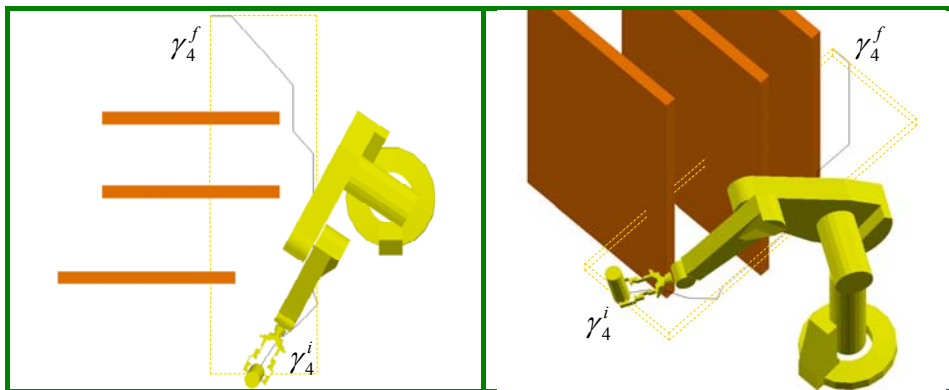


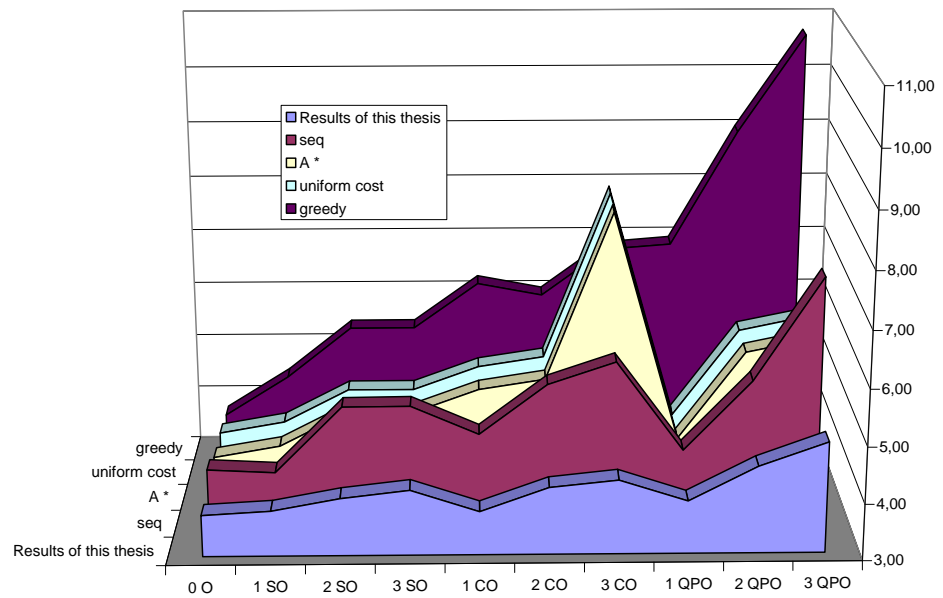
Figure 3.9: Example 1– The Case of Three Quadri-Lateral Plane Obstacles.

The numerical results of this example and the comparison with Rubio et al. 2009a results are tabulated in the next Table (3.3). The cases of the three basic obstacle elements with different quantities are presented here. The column titled as “Results of this thesis” contains the results of the proposed GA procedure. The column titled by Rubio et al. 2009a contains 4 sub-columns (correspond to the results of 4 different approaches used by him). These approaches are: (1) In-direct algorithm: seq, (2) Simultaneous algorithm: A*, (3) Simultaneous algorithm: uniform cost, and (4) Simultaneous algorithm: greedy. For more details about these approaches please refer to their article; Rubio et al. 2009a.

		Results of this thesis	Rubio et al. 2009a Results			
			seq	A *	unifor m cost	greedy
0 Obstacles	d_s (m)	3.7358	4.07	3.83	3.83	3.73
	d_e (m)	1.5199				
	t_{c1} (s)	5532	66.31	112.14	651.25	4.94
	t_{c2} (s)	7881				
	t_e (s)	2.00408				
1 Spherical Obstacle	d_s (m)	3.8029	4.01	4.03	4.03	4.46
	d_e (m)	1.5199				
	t_{c1} (s)	10567	201.11	257.99	1267.8	10.48
	t_{c2} (s)	9238				
	t_e (s)	2.11442				
2 Spherical Obstacle	d_s (m)	4.0187	5.18	4.63	4.63	5.38
	d_e (m)	1.5346				
	t_{c1} (s)	3806	211.85	484.03	1685.8	44.86
	t_{c2} (s)	12486				
	t_e (s)	2.37418				
3 Spherical Obstacle	d_s (m)	4.1585	5.19	4.63	4.63	5.38
	d_e (m)	1.5199				
	t_{c1} (s)	4013	193.44	485.36	1682.4	44.98
	t_{c2} (s)	17395				
	t_e (s)	3.36064				
1 Cylispherical Obstacle	d_s (m)	3.7692	4.68	5.05	5.05	6.21
	d_e (m)	1.5199				
	t_{c1} (s)	3932	122.97	149.02	744.30	28.78
	t_{c2} (s)	6946				
	t_e (s)	2.24456				
2 Cylispherical Obstacle	d_s (m)	4.1915	5.56	5.23	5.23	5.99
	d_e (m)	1.6149				
	t_{c1} (s)	8139	260.63	270.52	987.85	15.03
	t_{c2} (s)	14505				
	t_e (s)	2.62379				
3 Cylispherical Obstacle	d_s (m)	4.3138	5.95	8.20	8.20	6.84
	d_e (m)	1.5917				
	t_{c1} (s)	8366	230.48	869.02	1457.4	23.80
	t_{c2} (s)	18371				
	t_e (s)	3.15649				

1 Quadri-lateral Plane Obstacle	d_s (m)	3.9348	4.37	4.13	4.13	6.93
	d_e (m)	1.5364				
	t_{c1} (s)	10272	76.17	110.36	529.52	16.14
	t_{c2} (s)	14019				
	t_e (s)	3.03276				
2 Quadri-lateral Plane Obstacle	d_s (m)	4.5412	5.60	5.70	5.70	8.98
	d_e (m)	2.1387				
	t_{c1} (s)	19523	198.45	346.43	869.47	67.80
	t_{c2} (s)	20833				
	t_e (s)	3.29023				
3 Quadri-lateral Plane Obstacle	d_s (m)	4.9628	7.42	5.94	5.94	10.71
	d_e (m)	1.8353				
	t_{c1} (s)	24322	1676.1	602.91	1407	82.27
	t_{c2} (s)	25710				
	t_e (s)	4.19822				

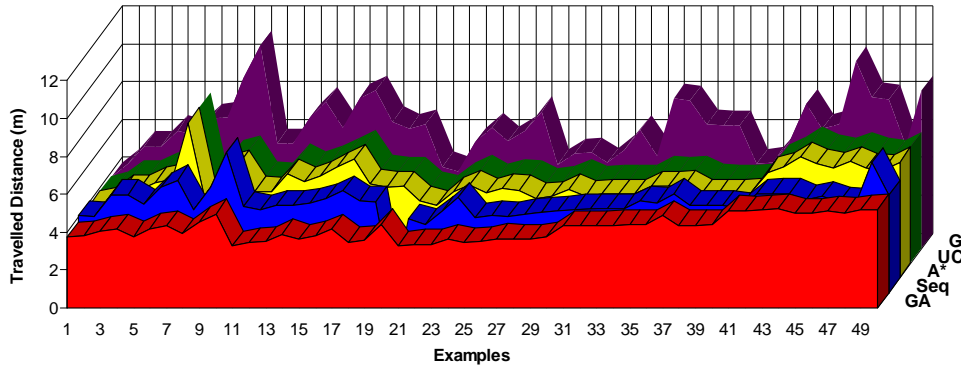
Table 3.3: Example 1 Results.



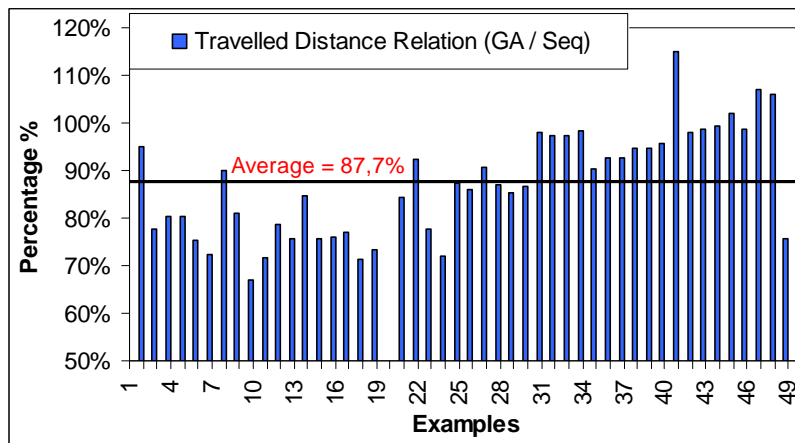
Graph 3.1: Example 1 – A Comparison Results of d_s (m) Between GA Procedure and Rubio Algorithms.

3.5.1.2. Example 2:

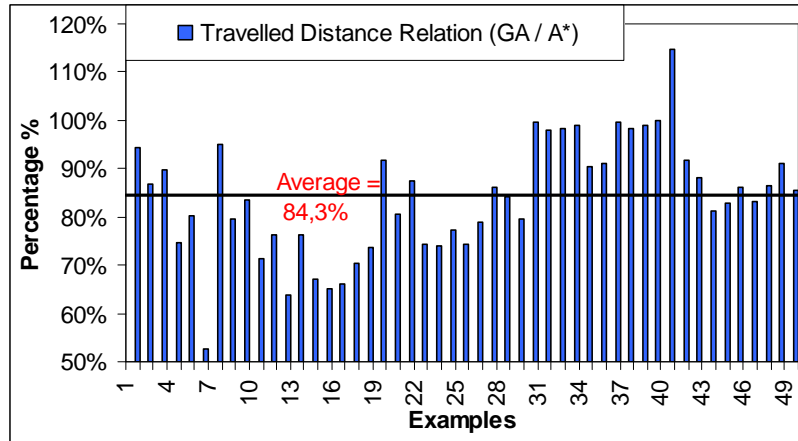
In this example, a group of 50 examples with different initial and final configurations and different obstacles will be discussed. These examples have been solved by Rubio et al. 2009a.



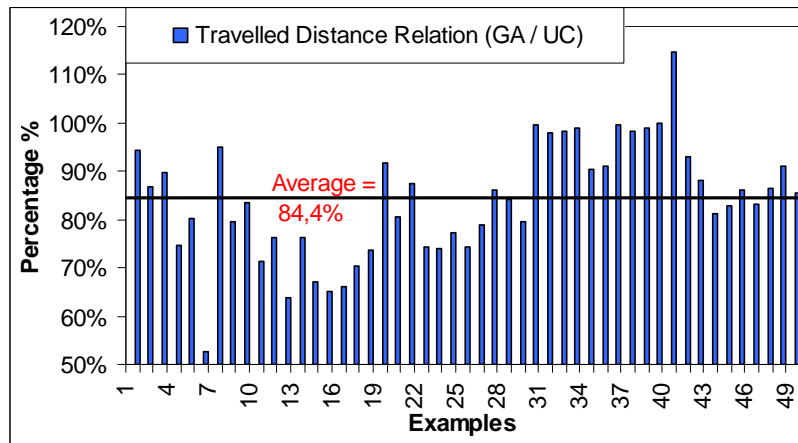
Graph 3.2: Traveled Distance Comparison Between GA Procedure and Rubio Procedures.



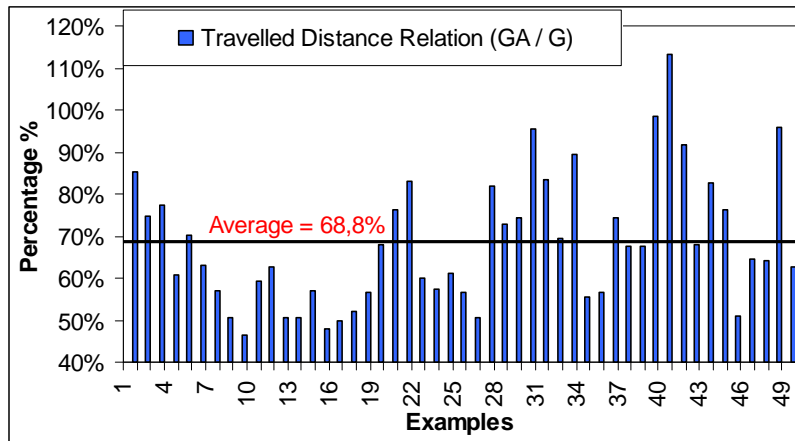
Graph 3.3: Traveled Distance Comparison Between GA Procedure & Seq Procedure Produced by Rubio et al. 2009a.



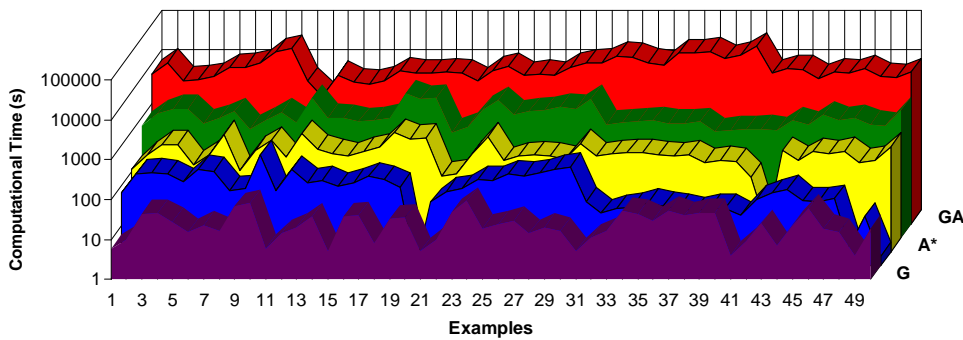
Graph 3.4: Travelled Distance Comparison Between GA Procedure & A* Procedure Produced by Rubio et al. 2009a.



Graph 3.5: Travelled Distance Comparison Between GA Procedure & UC Procedure Produced by Rubio et al. 2009a.



Graph 3.6: Travelled Distance Between GA Procedure & G Procedure Produced by Rubio et al. 2009a.



Graph 3.7: Computational Time Comparison Between GA and Rubio Procedures.

3.5.1.3. Example 3: Industrial Application – Comparison Results

This example demonstrates the effectiveness of the mentioned algorithm. The robot initial and final configurations are shown in Table (3.4). Obstacles are shown in Table (3.5).

Joint No.	1	2	3	4	5	6
Initial configuration	-7.50°	-174.80°	46.40°	4.30°	16.50°	-6.50°

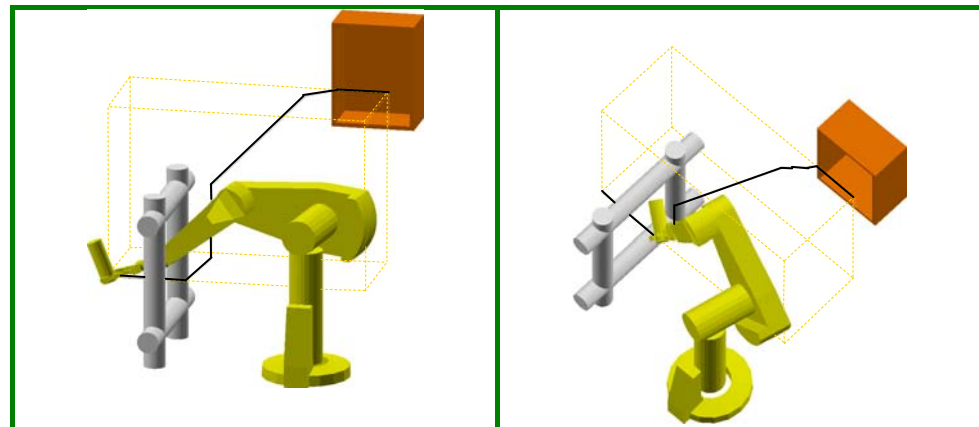
Joint No.	1	2	3	4	5	6
Final configuration	-95.10°	-101.20°	15.59°	0.00°	0.00°	0.00°

Table 3.4: Initial and Final Configurations for Example 3.

1 st Cylindrical obstacle	2 nd Cylindrical obstacle	3 rd Cylindrical obstacle	4 th Cylindrical obstacle
$C_1^{Cyl,1} = (-0.7, 0.5, 0.0)$	$C_2^{Cyl,1} = (-0.7, 0.0, 0.0)$	$C_3^{Cyl,1} = (-0.7, -0.15, 0.7)$	$C_4^{Cyl,1} = (-0.7, -0.15, 0.15)$
$C_1^{Cyl,2} = (-0.7, 0.5, 0.8)$	$C_2^{Cyl,2} = (-0.7, 0.0, 0.8)$	$C_3^{Cyl,2} = (-0.7, 0.65, 0.7)$	$C_4^{Cyl,2} = (-0.7, 0.65, 0.15)$
$r_1^{Cyl} = 0.15$	$r_2^{Cyl} = 0.15$	$r_3^{Cyl} = 0.15$	$r_4^{Cyl} = 0.15$
1 st Prismatic obstacle	2 nd Prismatic obstacle	3 rd Prismatic obstacle	4 th Prismatic obstacle
$P_{11} = (0.31, 0.79, 1.42)$	$P_{21} = (0.31, 0.79, 1.42)$	$P_{31} = (-0.03, 0.79, 1.42)$	$P_{41} = (-0.03, 0.79, 0.97)$
$P_{12} = (0.31, 0.99, 1.42)$	$P_{22} = (0.31, 0.99, 1.42)$	$P_{32} = (-0.03, 0.99, 1.42)$	$P_{42} = (-0.03, 0.99, 0.97)$
$P_{13} = (0.31, 0.79, 0.97)$	$P_{23} = (-0.03, 0.99, 1.42)$	$P_{33} = (-0.03, 0.99, 0.97)$	$P_{43} = (0.31, 0.99, 0.97)$
$P_{14} = (0.31, 0.99, 0.97)$	$P_{24} = (-0.03, 0.79, 1.42)$	$P_{34} = (-0.03, 0.79, 0.97)$	$P_{44} = (0.31, 0.79, 0.97)$

Table 3.5: Obstacles Locations (in m) for Example 3.

The next Figure (3.10) shows the path evolution from the initial robot configuration to the final configuration in a complex environment.



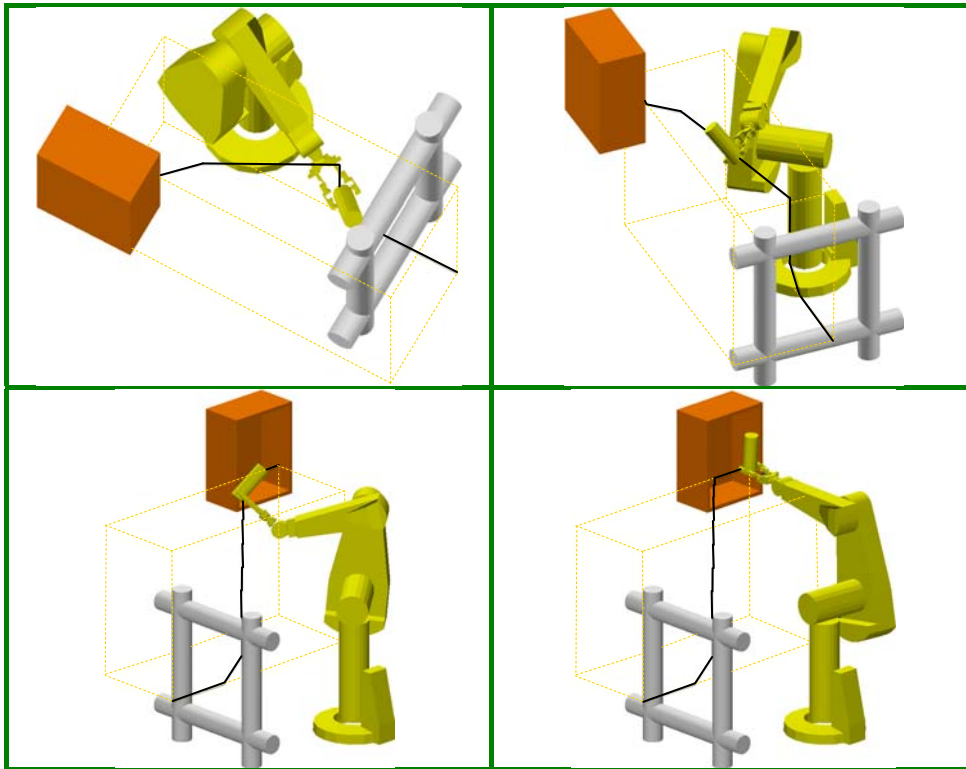


Figure 3.10: Example 3.

In this example, the path-planning problem has been solved in first place, and then the trajectory has been adjusted. This in-direct method of obtaining the trajectory has been compared with the direct method developed by Rubio et al. 2009b. The comparison of results for this example is shown in Table (3.6):

		d_s (m)	d_e (m)	t_c (s)	t_e (s)
Rubio et al. 2009b	A *	5.82		17049.94	35.61
	uniform cost	5.41		16233.08	29.23
	greedy	5.43		2674.69	45.70
Thesis Results		4.3181	1.7858	17782	1.63415

Table 3.6: Example 3 Results.

3.5.2. Time Optimizer Examples

In this approach, many examples have been solved to show the efficiency of the time optimizer algorithm. The first example is a comparison example, while the other examples are illustrated by calculated the minimum time needed to adjust trajectories on paths resulting from path planning procedure examples group.

3.5.2.1. Example 1: Comparison Results with Tse and Wang 1998

For illustration, consider a Puma 560 type robot with six revolute joints. Eight intermediate configurations from a Cartesian path of the hand are selected, Table (3.7). The robot is at rest initially, and comes to a full stop at the end of the minimum time interval. In this example, only the physical constraints will be considered. The velocity, acceleration, and jerk constraints are given in Table (3.8). This example published by Tse and Wang 1998, so a comparison between results will be made. Tse and Wang tested their algorithm using combinations of crossover rate (0.35, 0.65, 0.95) and mutation rate (0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4).

N° of Configuration	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
1	10	15	45	5	10	6
2	35	20	112.5	12.5	20	23
3	60	25	180	20	30	40
4	75	30	200	60	-40	80
5	130	-45	120	110	-60	70
6	110	-55	15	20	10	-10
7	100	-70	-10	60	50	10
8	-10	-10	100	-100	-40	30
9	-30	0	75	-65	-15	25
10	-50	10	50	-30	10	20

Table 3.7: Sequence of Configurations [degree].

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Velocity (deg/s)	100	95	100	150	130	110
Acceleration (deg/s²)	45	40	75	70	90	80
Jerk (deg/s³)	60	60	55	70	75	70

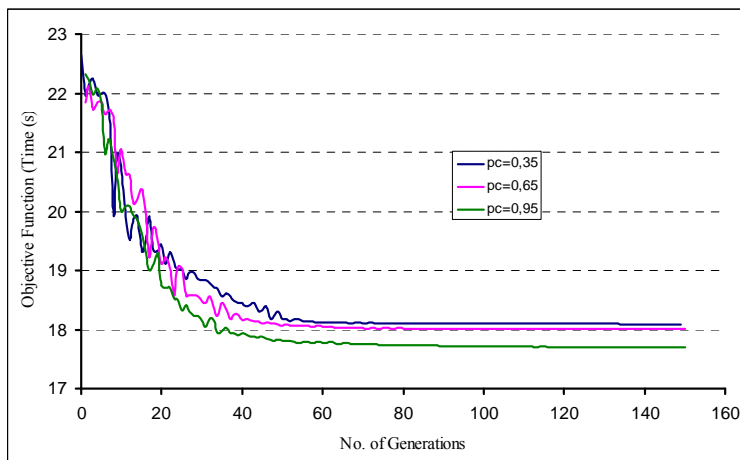
Table 3.8: Velocity, Acceleration, and Jerk Constraints.

After exploring the experiments, the GA parameters that give the best solution are listed in the next table.

Description	Parameter	Value
Population size	popsiz	30
N° of populations	numpop	3
Generation number	ngen	150
Crossover rate	pcross	0.95
Mutation rate	pmut	0.05
Number of migration	nmig	7
Number of solutions replaced by new generation	nReplacement	5

Table 3.9: Parameter Values for the Genetic Algorithm Procedure.

The next graph demonstrates the evolution of the time over generations in different crossover rates (0.35, 0.65, 0.95) combined with mutation rate 0.05.



Graph 3.8: Objective Function (Time in seconds) vs. No. of Generations, Example 1.

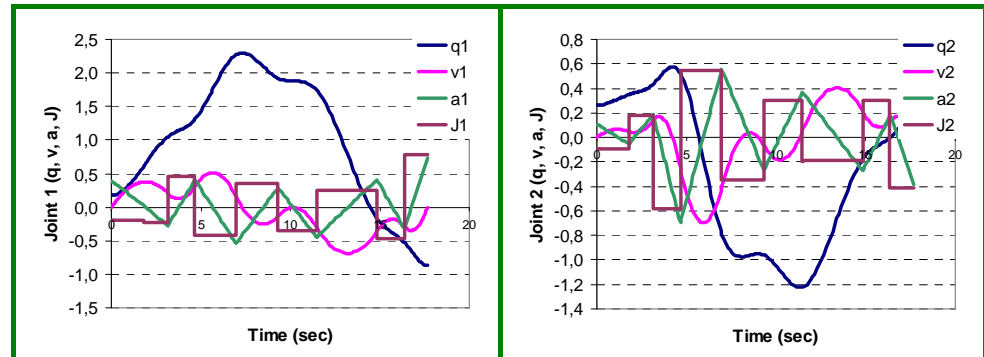
The next table demonstrates the comparison of results between this GA procedure and Tse and Wang procedure. The results are about different runs with crossover rate 0.95 combined with mutation rates 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4.

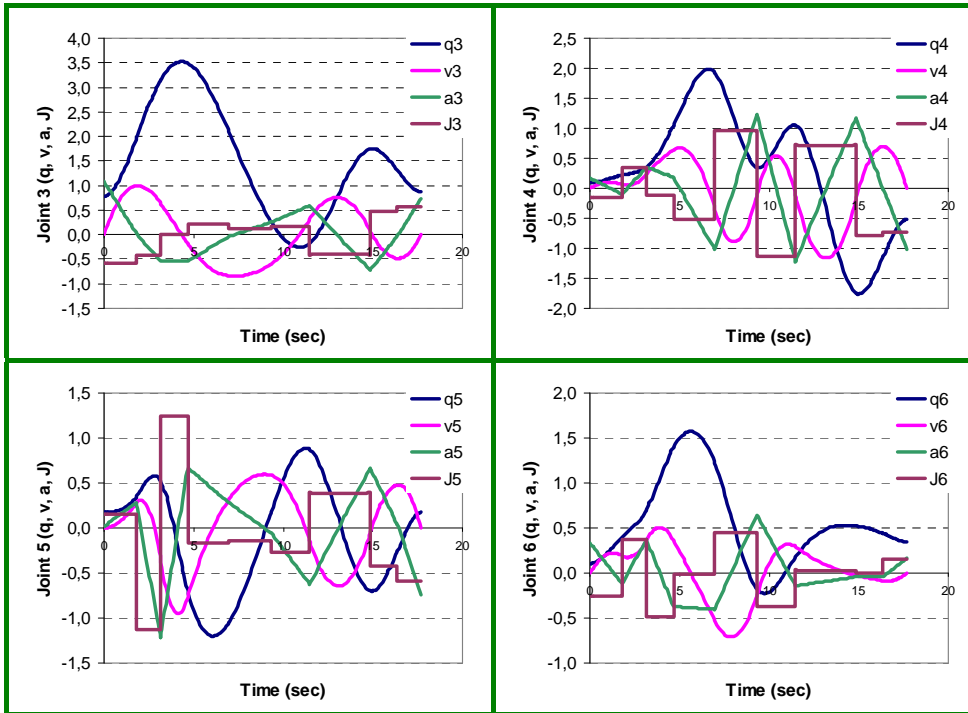
	0.001	0.01	0.05	0.1	0.2	0.3	0.4
Tse and Wang 1998 Thesis algorithm	20.156	19.880	18.211	18.226	18.929	18.957	19.062
Thesis algorithm	18.091	17.726	17.706	17.971	17.896	17.897	17.931

Table 3.10: Best Minimum Time with Crossover Rate = 0.95, Example 1.

From the above results, the best minimum time found is 17.706 seconds obtained from the GA search with crossover rate = 0.95 and mutation rate = 0.05. Besides, the rest of results in Table (3.10) are better than the Tse and Wang results. In addition, the results obtained from the combinations of crossover rate = 0.35, 0.65 and mutation rate = 0.01 and 0.05 are 18.087 and 18.009 respectively, are better than the Tse and Wang results 18.356 and 18.258.

The next graphs show the optimum joint trajectories.

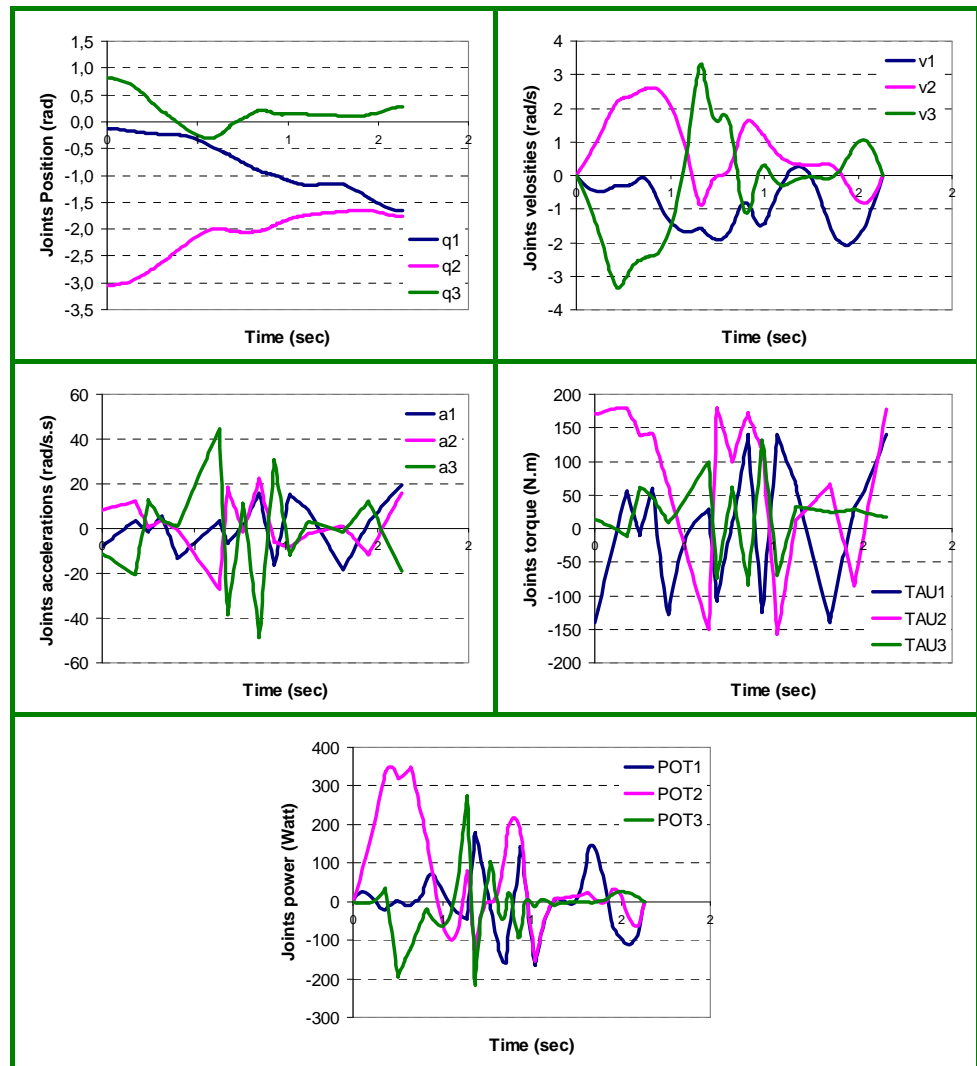




Graph 3.9: Joint Variables and Derivatives vs. Time.

3.5.2.2. Example 2:

In graph (3.10), the kinematic parameters (position, velocity and acceleration), the actuators torques, and the power developed in the first three actuators are shown. The graph corresponds to example sub-Section (3.5.1.2). The torques in the actuators are limited due to the following values: $\tau_1 \leq |140| \text{ N.m}$, $\tau_2 \leq |180| \text{ N.m}$, $\tau_3 \leq |140| \text{ N.m}$, $\tau_4 \leq |80| \text{ N.m}$, $\tau_5 \leq |80| \text{ N.m}$, $\tau_6 \leq |40| \text{ N.m}$. The power limits are $Pot_1 \leq |275| \text{ Watt}$, $Pot_2 \leq |350| \text{ Watt}$, $Pot_3 \leq |275| \text{ Watt}$, $Pot_4 \leq |150| \text{ Watt}$, $Pot_5 \leq |150| \text{ Watt}$, $Pot_6 \leq |75| \text{ Watt}$.



Graph 3.10: Joints (Coordinates, Velocities, Accelerations, Torques, and Power) vs. Execution Time.

3.6. DISCUSSION OF RESULTS

The examples illustrated in previous section prove the ability of the presented procedure to solve the Path planning problem for industrial robots.

The traveled distance by the significant points of the robot, are very acceptable, and by comparing the results with other works, in the most of cases, the traveled distance of the presented procedure is more desirable than the ones of the other works.

In the example (1), sub-Section (3.5.1.1), the comparison of results shows the efficiency of the proposed GA procedure over the four procedures (Seq, A*, UC, G) provided by Rubio et al. 2009a. The GA procedure improved the results of the traveled distance for path planning by an average of percentage 87.7%, 84.3%, 84.4%, and 68.6%, respectively.

The computation time in all examples is high which may considered as the main disadvantage of the genetic algorithm in general. Referring to In the example (1), sub-Section (3.5.1.1), the computational time for GA procedure is higher than the one obtained by the four procedures (Seq, A*, UC, G) provided by Rubio et al. 2009a by an average of percentage 28821%, 7316%, 1044% and 50872%.

The presented procedure shows a significant ability to adapt the robot and its path to any workspace characteristics.

CHAPTER 4

TRAJECTORY PLANNING

For industrial robots, the problem of minimum time trajectory planning has been addressed by numerous researchers motivated by the direct relation between the tasks executed in minimum time and the productivity in manufacturing systems.

The trajectory-planning problem aims at finding a relationship between two elements belonging to two different domains: time and space. Accordingly, the trajectory is usually expressed as a parametric function of the time, which provides at each instant the corresponding desired position. Obviously, after having defined this function, other aspects related to its implementation must be considered, such as time discretization, saturation of the actuation system, and so on.

The main distinction among the various categories of trajectories consists in the fact that they can be one- or multi-dimensional. In the first case, they define a position for a one DOF system, while in the latter case a multidimensional working space is considered. From a formal point of view, the difference between

these two classes of trajectories consists in the fact that they are defined by a scalar ($p = p(t)$) or a vectorial ($\vec{q} = \overrightarrow{q(t)}$) function.

The working scope of this thesis deals with the multidimensional trajectories. The Merriam-Webster dictionary defines the trajectory as “the curve that a body describes in space”. Although in the case of a machine composed by several motors each of them can be independently programmed and controlled (control in the joint space), many applications require coordination among the different axes of motion with the purpose of obtaining a desired multidimensional trajectory in the operational space of the machine. This is the case of tool machines used to cut, mill, drill, grind, or polish a given workpiece, or of robots, which must perform tasks in the three-dimensional space, such as spot, welding, arc welding, handling, gluing, etc.

Actually, as mentioned before in Section (1.3), trajectory-planning problem for multidimensional trajectories has been analyzed using two different approaches: direct or global approaches and decoupled or indirect approaches. Indirect approaches firstly seek for a path in the configuration space, and then the trajectory adjusts; subjected to the dynamic constraints of the manipulator, see Saramago and Steffen 2001 and Valero et al. 2006. On the other hand, the search takes place in the system’s state space in the direct approaches. These approaches involve optimal control and numerical optimization (see Saramago and Steffen 2001, Plessis and Snyman 2003, Gasparetto and Zanotto 2007).

Most of the existing methods belong to one or other of these types, although the indirect methods are the most widely used. For depth knowledge you should refer to Piazzzi and Visioli 1997a, 2000, Saramago and Steffen 2001, Plessis and Snyman 2003, Behzadipour and Khajepour 2006, Valero et al. 2006, Bertolazzi et al. 2007, Gasparetto and Zanotto 2007.

A characteristic of indirect methods is that the path is known previously; either because it depends on the activity to be done by the industrial robot or because it has been generated by a path planner. Generally speaking, the indirect methods combine the path planning with the obtaining of the time history of motion usually in a sequential way.

By contrast, direct methods are characterized primarily because they do not separate the path planning from the time history of motion rather they directly solve the problem in the state space of the robot. They try to solve the trajectory directly based on the evolution of dynamic variables, taking into account geometrical constraints and setting out an optimization problem to optimize some cost function. Some examples of direct methods are presented in Constantinescu and Croft 2000, Chettibi et al. 2004, Abdel-Malek et al. 2006.

The basic trajectory can be analytically expressed by polynomials, harmonics, exponential, etc. In this thesis, a polynomial presentation is used for simplicity. The degree n of the polynomial depends on the number of conditions to be satisfied and on the desired “smoothness” of the resulting motion. Since the number of boundary conditions is usually even, the degree n of the polynomial function is odd, i.e. three, five, seven, and so on. In our case, a third degree polynomial will be used.

In this thesis, the trajectory planning will be obtained in means of adjacent configurations concepts, these adjacent configurations have a new definition a slightly differs from the definition used in the path planning explained in Section (3.1).

4.1. ADJACENT CONFIGURATIONS FOR TRAJECTORY PLANNING

In this section, the process of generating a discrete space of configuration is presented. This space of configurations is based on the obtaining of adjacent configurations developed by Valero 1990, Valero et al. 1997, Valero et al. 2006, and redefined by Abu-Dakka et al. 2007, Abu-Dakka et al. 2008.

Adjacent configurations are useful in two means. Firstly, it can be used to generate a space of adjacent configurations between the initial and goal configurations. After that, by applying a search algorithm (such as A*, etc.), the pursued trajectory between the initial and final configurations can be found. This strategy is not the goal of this thesis, but a test has been done to ensure the capability of the algorithm to construct a space of adjacent configurations, see Section (4.1.6). The second functionality of the adjacent configurations generation is that it can be used to construct a pursued trajectory directly without the need of a complete space of adjacent configuration. The only need is to find the adjacent configurations necessary to build the pursued trajectory gradually.

4.1.1. Adjacent Configurations Formulation

The adjacent configurations can be defined as follows: The configuration C^k is adjacent to a given configuration C^p , if they are feasible and the three following conditions are satisfied, Valero et al. 2005, Valero et al. 2000:

1. The end-effector position γ_4 (see Figure (2.1)) corresponds to a point of the discrete workspace. In addition, it is one increment far from the point

corresponding to the C^p configuration, so it is said that, the two configurations are neighboring and there must be a given increment between them less than the smallest obstacle size in the workspace.

2. Verify the absence of obstacles between adjacent configurations C^k and C^p . Also, to verify that the distance between significant points satisfy the following condition,

$$\left| \overrightarrow{\gamma_i^p \gamma_i^k} \right| \leq 2 \cdot \min(r_j); \quad i = 1, 2, 3; \quad j = 1, 2, \dots \quad (4.1)$$

where r_j is the minimum characteristic dimension of the obstacles in the workspace.

3. C^k should be such as to minimize the function:

$$\|C^k - C^p\| = A \cdot t^2 + B \cdot \sum_{i=1}^6 (q_i^f - q_i^k)^2 + C \cdot \left(\begin{array}{l} (\gamma_j^k - \gamma_j^p)_x^2 + \\ \sum_{j=1}^4 (\gamma_j^k - \gamma_j^p)_y^2 + \\ (\gamma_j^k - \gamma_j^p)_z^2 \end{array} \right) \quad (4.2)$$

where A, B , and C are coefficients.

The first term t^2 is the time needed to move the end effector between adjacent configurations through a third degree polynomial trajectory expressed in the next Section (4.1.2).

The second term $(q_i^f - q_i^k)^2$ is the joint coordinates, which aims to minimize the difference between the joint coordinates of the generated configuration C^k , and the joint coordinates of the goal configuration of the robot, where q_i is the joint value.

$$\text{The last term} \left(\begin{array}{l} (\gamma_j^k - \gamma_j^p)_x^2 + \\ \sum_{j=1}^4 (\gamma_j^k - \gamma_j^p)_y^2 + \\ (\gamma_j^k - \gamma_j^p)_z^2 \end{array} \right)$$

expressed in Cartesian coordinates, aims to minimize the distance between significant points.

4.1.2. Third-Order Polynomial Trajectory Planning Between Adjacent Configurations

The motion of the robot between adjacent configurations $C^k(q_i^k)$ and $C^p(q_i^p)$ has been presented by a third-order polynomial function as following:

$$q_i^{pk} = a_i^{pk} + b_i^{pk} t_{pk} + c_i^{pk} t_{pk}^2 + d_i^{pk} t_{pk}^3 \quad (4.3)$$

$$\dot{q}_i^{pk} = b_i^{pk} + 2 * c_i^{pk} t_{pk} + 3 * d_i^{pk} t_{pk}^2 \quad (4.4)$$

$$\ddot{q}_i^{pk} = 2c_i^{pk} + 6 * d_i^{pk} t_{pk} \quad (4.5)$$

where $a_i^{pk}, b_i^{pk}, c_i^{pk}, d_i^{pk}$ are the polynomial coefficients, q_i^{pk} is the generalized coordinates, and t_{pk} is the minimum time necessary to go from C^p to C^k , satisfying the robot's torque constraints τ_{\max}^i and τ_{\min}^i . The verification of the maximum and minimum torque in each actuator is done by dividing the interval t_{pk} into intermediate points, and then, solving the corresponding inverse dynamic problem, using the recursive Newton-Euler formulation (Section (2.3)) to obtain the joint torques required for a given set of joint angles, velocities, and accelerations.

For a solution of t_{pk} , the coefficients of the polynomial function can be determined from the following four equations:

$$\text{For } t=0 \quad \Rightarrow \quad \begin{cases} q_i^{pk}(0) = q_i^p \\ \dot{q}_i^{pk}(0) = 0 \end{cases} \quad (4.6)$$

$$\text{For } t=t_{pk} \quad \Rightarrow \quad \begin{cases} q_i^{pk}(t_{pk}) = q_i^k \\ \dot{q}_i^{pk}(t_{pk}) = 0 \end{cases} \quad (4.7)$$

Requiring zero velocity at the ends does not fit in the motion conditions between the configurations C^p and C^k as if they were part of the pursued trajectory. However, it facilitates the comparison between the configurations that constitute the discrete space since common initial and goal velocity requirements are imposed.

The coefficients of the polynomial could be determined as following:

$$\begin{aligned}
a_i^{pk} &= q_i^p \\
b_i^{pk} &= 0 \\
c_i^{pk} &= \left(\frac{3}{t^2}\right) * (q_i^k - q_i^p) \\
d_i^{pk} &= \left(-\frac{2}{t^3}\right) * (q_i^k - q_i^p)
\end{aligned} \tag{4.8}$$

The solution of the optimization problem is obtained by using two different algorithms. The first one is based on nonlinear Sequential Quadratic Programming (SQP) method using an optimization routine provided by the NAG (Numerical Algorithms Group) commercial library, for more details see Abu-Dakka et al. 2007. The other algorithm solve the problem using genetic algorithms, Abu-Dakka et al. 2008.

4.1.3. Workspace Discretization

The first step of the optimization process is generating discrete space of configurations. This space is defined basing on the position of the end-effector and is considered as a rectangular prism between the initial and final configurations of the robot, with its axis parallel to the Cartesian reference system, see Figure (4.1). This space has been modeled in the same way as the space explained in sub-Section (3.1.2).

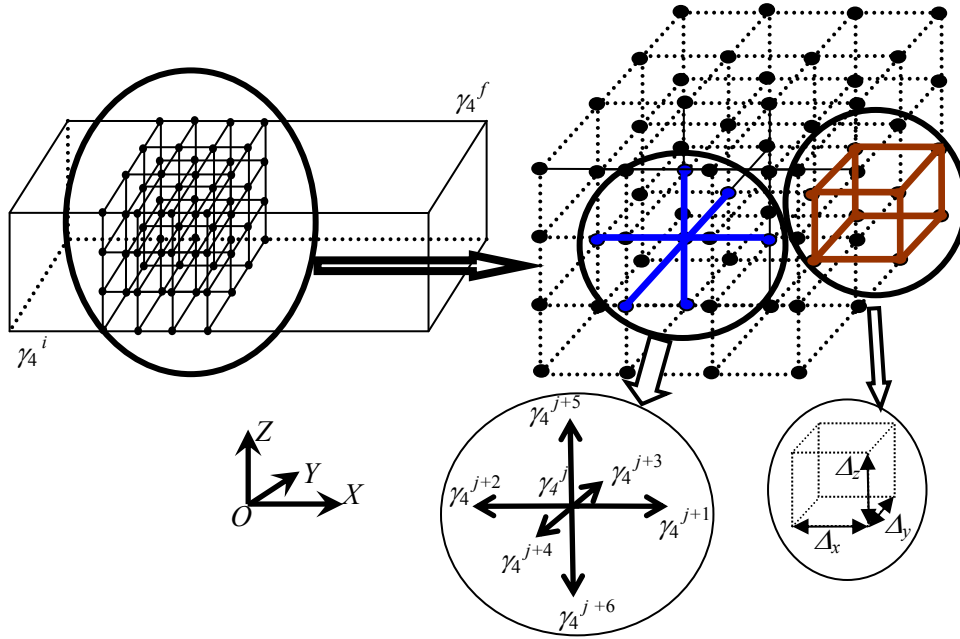


Figure 4.1: Workspace Generation.

As mentioned before, to generate this space of configurations, two optimization algorithms are tested: SQP and Genetic algorithms. The information (like: C^k , C^p , t_{pk} , from where you can access current position, etc.) that can be resulted from the algorithm, are stored in a database with the same form as the discretized prism.

4.1.4. Sequential Quadratic Programming Algorithm

As was observed in the previous part of this chapter and Chapter 2, the optimization process for trajectories consists of nonlinear cost function and nonlinear constraint equations. The problem variables are seven; the generalized coordinates ($q_{i; i=1,2,\dots,6}$) and the time t_{pk} . The problem constraints are varied as the following:

$$1) \tau_{\min,i} < \tau_i < \tau_{\max,i} \quad i = 1,2,\dots,6 \quad (4.9)$$

$$2) \|\gamma_j^k - \gamma_j^p\| < D \quad j = 1,2,3,4 \quad D = \text{smallest obstacle size} \quad (4.10)$$

$$3) \|\gamma_{4,obtained} - P_{goal \ position}\| \leq 0.02 \text{ meter} \quad (4.11)$$

4) Obstacle avoidance constraints.

The nonlinear actuators torques constraints, which are considered as dynamic constraints, see Section (2.3). The derivative of equation (4.9) cannot be obtained analytically, but it can be calculated numerically by NAG routine.

Equation (4.10) aims to restrict the distance between the significant points in C^k and their corresponding in C^p . The derivative is:

$$\frac{d\|\gamma_j^k - \gamma_j^p\|}{dq_i} = \frac{1}{\|\gamma_j^k - \gamma_j^p\|} \cdot \Sigma \left(\begin{array}{l} \left(\gamma_j^k - \gamma_j^p \right)_x \cdot \left(\frac{d\gamma_j}{dq_i} \right)_x + \\ \left(\gamma_j^k - \gamma_j^p \right)_y \cdot \left(\frac{d\gamma_j}{dq_i} \right)_y + \\ \left(\gamma_j^k - \gamma_j^p \right)_z \cdot \left(\frac{d\gamma_j}{dq_i} \right)_z \end{array} \right) \quad (4.12)$$

where $\frac{d\gamma_j}{dq_i}$ can be calculated by Equation 2.5.

Equation (4.11) aims to restrict the distance between the end-effector position of C^k and the goal position. The derivative is:

The analytical partial derivatives with respect to the generalized coordinates ($q_i, i=1,2,\dots,6$) and time for the nonlinear collision avoidance constraints were provided, see Section (2.5).

The cost function partial derivatives were calculated analytically as follows:

$$\frac{d\|C^k - C^p\|}{dq_i} = 2 \cdot B \cdot \sum_{i=1}^6 (q_i^f - q_i^k) + 2 \cdot C \cdot \left(\begin{array}{c} (\gamma_j^k - \gamma_j^p)_x * \left(\frac{d\gamma_4}{dq_i} \right)_x + \\ \sum_{j=1}^4 (\gamma_j^k - \gamma_j^p)_y * \left(\frac{d\gamma_4}{dq_i} \right)_y + \\ (\gamma_j^k - \gamma_j^p)_z * \left(\frac{d\gamma_4}{dq_i} \right)_z \end{array} \right) \quad (4.13)$$

$$\frac{d\|C^k - C^p\|}{dt} = 2 \cdot A \cdot t \quad (4.14)$$

where $\frac{d\gamma_4}{dq_i}$ can be calculated by Equation 2.5.

The obtained solution guarantees the prevention of collisions and the dynamic feasibility of the movement. In this problem, the objective function had been generated by heuristically adjustment of coefficients as in Equation (4.2).

This mathematical model has been solved using a nonlinear Sequential Quadratic Programming (SQP) optimization routine provided by the NAG (Numerical Algorithms Group) commercial library, see Abu-Dakka et al. 2007.

4.1.5. Genetic Algorithm Optimization Procedure

In this section, a genetic algorithm procedure will be introduced to obtain a space of adjacent configurations, see Abu-Dakka et al. 2008.

A SSGA procedure is used to obtain the C^k for a given γ_4 . In this algorithm, a real presentation (coding scheme) has been used. The main GA operators exposed for this algorithm will be as follows:

- Chromosome

The individual or the chromosome represents the robot configuration and the time to be optimized. Each chromosome consists of seven genes; six are the robot generalized coordinates ($q_i, i=1,2,\dots,6$) and the seventh is the time needed to move the robot end-effector between the adjacent configurations.

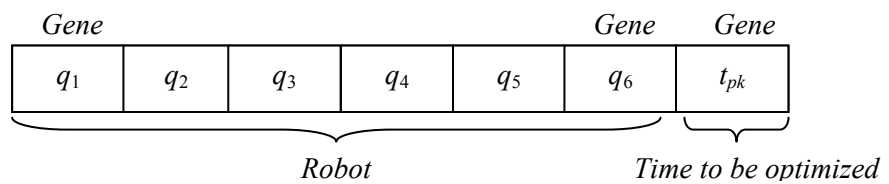


Figure4.2: Adjacent Configuration GA Chromosome.

- Selection

A roulette-wheel selection method is applied to select individuals for crossover and mutation.

- Crossover

The crossover operator defines the procedure for generating a child from two selected parents. A single point crossover used in this procedure, see Figure (3.3).

- Mutation

The mutation operator defines the procedure for mutating each genome. In this procedure, an offspring will be selected randomly then a gene will be selected randomly from that offspring, Figure (3.3). The mutation will occur with respect to the Equation (3.5).

- Objective Function

Minimize Equation (4.2).

The resulting offspring will be tested if it's a valid robot configuration with dynamic compatibility or not, regardless if it's better or worse than the original one (Because GA will deal with that).

As mentioned in Section (1.4), some of the advantages of the Genetic Algorithms over other SQP optimization technique are: the SQP are sensitive to the initial guess for the variable, while GA searches from a population of points, rather than a single one. Using GAs, there is no need for derivatives or any mathematical complexion. GAs use probabilistic transition not deterministic rules.

4.1.6. Comparison Results Between SQP and GA

Many examples were applied in various cases, next Table (4.1) shows the results of 6 different examples with different environments conditions. A space of adjacent configurations has been generated using SQP and GA algorithms. In each space, the average of the optimized time between each pair of adjacent configurations has been calculated. In addition, the average computational time, needed to generate a robot configuration adjacent to a given one, has been calculated. The GA algorithm shows a high ability of convergence and coverage in

comparison with the SQP algorithm, it reaches to 95-98% for cases without obstacles. The table results (4.1) shows the average time needed to move the robot between adjacent configurations and the percentage of total number of convergence; which means the total number of successful adjacent configurations generated with respect to the total trials of the algorithm in a specific workspace; which depends on the complexity of the workspace. Finally, it will focus on the calculation time needed. In the table, Case 1 will demonstrate the results obtained from SQP. On the other hand, Case 2 will demonstrate the results obtained from the GA procedure.

	Case 1: SQP			Case 2: GA		
	Sphere	Cylinder	Plane	Sphere	Cylinder	Plane
Avg. Time of motion (sec)	1.4491	1.0244	1.3608	0.3581	0.3478	0.3461
Avg. Time of calculation (sec)	12.97	39.0	16.2	16.19	17.10	19.38
Percentage of convergence	78.6%	58.6%	81.6%	79.4%	78%	78%
	Two Spheres	Three Spheres	Sphere and cylinder	Two Spheres	Three Spheres	Sphere and cylinder
Avg. Time of motion (sec)	1.4036	1.5220	1.3608	0.3432	0.3436	0.3513
Avg. Time of calculation (sec)	14.9	15.1	7.8	10.4	14.89	15.56
Percentage of convergence	75.4%	79.7%	61.1%	78%	84%	75%

Table 4.1: Comparison Results Between GA & SQP.

As shown in the table, the GA procedure demonstrates higher efficiency than the SQP. Hence, the GA procedure for the obtaining of adjacent configurations will be used in the process of the obtaining the pursued trajectory.

4.2. OBTAINING THE TRAJECTORY

In this thesis, a mixed method (in two stages) using genetic algorithms for obtaining the minimum time pursued trajectory for industrial robots (at least 6 DOF) working in complex environments, in which the intermediate configurations are unknown (i.e., no assumptions are previously made for the path), is presented. In the first stage, the algorithm will optimize the trajectory time depending on the optimized time from the adjacent configurations explained in the previous Section (4.1); where the pursued trajectory is composed of set of adjacent configurations. In the second stage, the obtained trajectory time from the first stage will be optimized using genetic algorithms subjected to continuous velocity and acceleration between intermediate configurations.

The method proposed deals with two facts: the obstacles in the workspace and unknown intermediate configurations between C^i and C^f . These facts lead to uncertainties about the kinematic characteristics of intermediate points, highlighting that the knowledge of these kinematic characteristics are indispensable to solve the inverse dynamic problem.

The algorithm works on a discretized configuration space which is generated gradually as the direct procedure solution evolves, demands less computational effort than the corresponding indirect procedure, Valero et al. 2006.

The determination of the trajectory from C^i is achieved by applying a random search algorithm to look for the next adjacent configuration in the discretized configuration space and so on till the C^f is reached. The problem of the obtained trajectory is that it suffers from velocity and acceleration discontinuity between the intermediate points.

The objective of the algorithm is to minimize the traveling time t between C^i and C^f , where t equal the summation of the optimized time t_{pk} (the time obtained while confirming the dynamic compatibility associated with the adjacency between the configurations C^k and C^p) that constructed the trajectory.

$$t = \sum_{i=1}^f t_{pk,i} \quad (4.15)$$

This time t still is not the optimal time as the trajectory suffers from velocity and acceleration discontinuity between the via points. As a solution, the clamped cubic spline algorithm is applied to make continuous velocity and acceleration connections between via points (see Section (3.4)).

4.2.1. Genetic Algorithm Procedure

In this procedure, three optimization processes using genetic algorithms are involved. Firstly, optimization process for obtaining the adjacent configurations (detailed in Section (4.1)). The order in which the adjacent configurations are generated will condition the Space of Configurations generated and, therefore, the trajectory to be obtained. Second optimization process is applied for obtaining the pursued trajectory. Finally, an optimization procedure using clamped cubic spline is applied to optimize the trajectory time and to make continuous connections for velocities and accelerations between intermediate configurations.

Genetic algorithm for adjacent configuration uses the technique of steady-state reproduction without duplicates. This technique creates a certain number of children to replace the parents in the population, but discards children, which are duplicates of current individuals in the population (see Section (4.1.5)). On the

other hand, for the trajectory, a parallel populations GA procedure with migration technique has been implemented. The objective here is to find the shortest path between two configurations of robotic manipulator. Real coding scheme has been used to encode the parameters to generate the path.

In the GA based solution procedure, a number of new individuals are created at each iteration. The remaining individuals are obtained by deterministically copying the individuals with the top fitness from the previous generation.

4.2.1.1. Genetic Algorithms Operators and Parameters

As mentioned before, a parallel populations GA procedure with migration technique has been implemented to obtain minimum time trajectories. The main operators and characteristics in the exposed GA are:

- Individual:

The individual or the chromosome is a complete trajectory between C^i and C^f . Each chromosome is composed of a set of genes. Each gene contains the robot configuration $C\{q_1, q_2, \dots, q_i\}$, and the time needed to move the robot to this configuration. See Figure (4.3).

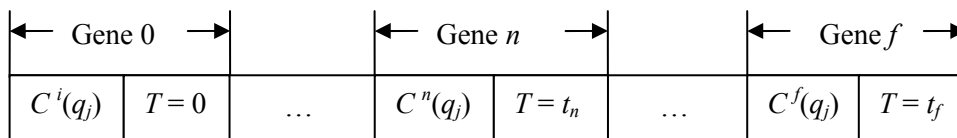


Figure 4.3: Trajectory GA Chromosome.

The first gene of each individual contains the initial configuration data. Then the ramification process to construct the chromosome will be started by

selecting randomly the next gene; based on the random search algorithm; by calling the adjacent configuration builder algorithm, and so on till the final configuration reached. The ramification can be done without repetition in seven directions: *X*-direction, *Y*-direction, *Z*-direction, *XY*-direction, *XZ*-direction, *YZ*-direction, and *XYZ*-direction. In this algorithm, there are no restrictions on the chromosome length and chromosomes can have different lengths.

In chromosome construction process, if the algorithm is not able to find the next adjacent configuration due to obstacles or dynamic incompatibility, a retuning back recursive technique will be applied. This technique depends on tracking back process, looking for the last possible configuration in which the robot can continue from it. If the tracking back drives the search to the initial configuration, this means that there is no possible trajectory in the workspace. In this case, the algorithm extends the workspace and starts again.

- Objective Function:

The objective of this algorithm is to minimize the equation (4.15).

- Crossover:

The crossover is made through the exchange of a part of the path between two trajectories chosen through the selection operation mentioned earlier. It is executed only if the probability of the crossover is satisfied. The crossover process for trajectory planning has been built in the same way detailed in sub-Section (3.2.1), Figure (3.3).

- Mutation:

Mutation is done by selecting randomly a point among the intermediate points in the trajectory (the first and final points are not considered for mutation). The point is then compared to the previous and next points in the trajectory. All

the possible changes with which the trajectory will remain incremental and quantum, are applied to the point. For more details see Figure (3.3) and sub-Section (3.2.1).

4.3. APPLICATION EXAMPLES & RESULTS

The introduced procedure has been applied to a Puma 560 robot using a computer with Intel Xeon CPU E5440 @ 2.83 GHz, 8 GB of RAM.

Four operational parameters have been studied when the procedure was applied to a numerous different examples. The parameters are:

- a) Execution time: The time needed to move the robot from the initial to the final configuration, denoted by t_e .
- b) Computational time, denoted by t_c :
- c) End-effector traveling distance, denoted by d_e .
- d) Summation of significant points traveling distance, Equation (4.16), denoted by d_s .

$$\sum_{i=1}^{n-1} \sum_{j=1}^m \sqrt{\left(\gamma_j^{i+1} - \gamma_j^i\right)_x^2 + \left(\gamma_j^{i+1} - \gamma_j^i\right)_y^2 + \left(\gamma_j^{i+1} - \gamma_j^i\right)_z^2} \quad (4.16)$$

where: j is the number of the significant and interesting points of the robot, and $m = 4$ for Puma 560 robot. $i = 1, 2, \dots, n$ is the number of robot configurations included in the trajectory.

4.3.1. Example 1: Comparison Results with Rubio et al. 2009b

This example demonstrates the effectiveness of the mentioned algorithm. The case of 0 obstacles and spherical obstacles of this example were solved by Rubio et al. 2009b. Thus, a results comparison will be done. Rubio compared his results by using three different approaches: A*, uniform cost (UC), and greedy (G). For more details about his procedure, please refer to his article.

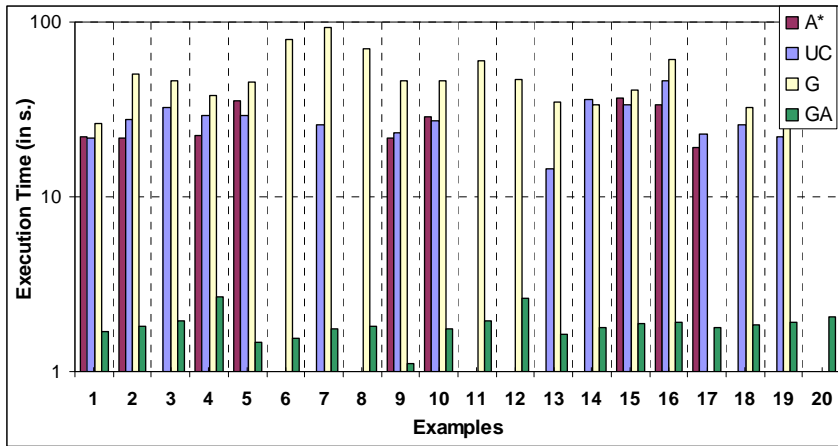
The robot initial and final configurations and obstacles are shown in the previous chapter Table (3.1) and (3.2) respectively.

The results of the part solved by Rubio et al. 2009b are tabulated and compared with the results of this thesis in Table (4.2).

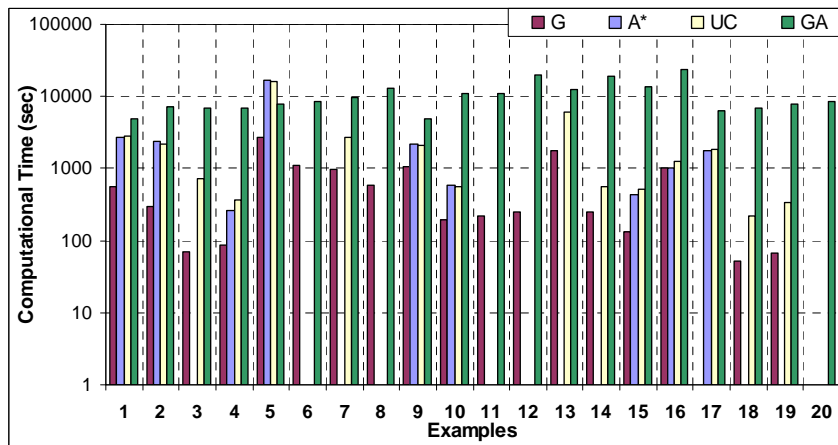
	Operational parameters	Results of this thesis	Rubio et al. 2009b Results		
			A *	uniform cost	greedy
0 Obstacles	t_e (s)	1.552	22.15	21.67	26.16
	d_s (m)	3.9446	3.65	3.65	4.19
	d_e (m)	1.68652			
	t_c (s)	4876	2691.27	2785.20	555.55
1 Spherical Obstacle	t_e (s)	1.80759	21.63	27.61	49.98
	d_s (m)	4.0642	4.48	4.11	5.94
	d_e (m)	1.5925			
	t_c (s)	7321	2360.28	2182.93	294.74
2 Spherical Obstacle	t_e (s)	1.96055		32.48	46.32
	d_s (m)	4.1335		5.81	5.71
	d_e (m)	1.5666			
	t_c (s)	6749		735.32	70.30
3 Spherical Obstacle	t_e (s)	2.67079	22.30	28.97	38.21
	d_s (m)	4.2554	5.35	4.90	5.57
	d_e (m)	1.6133			
	t_c (s)	6799	257.58	371.74	88.40

Table 4.2: Example 1 Results.

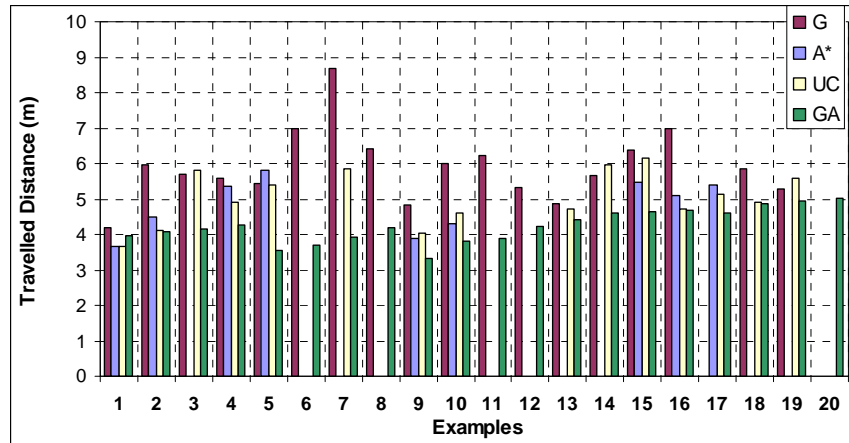
Moreover, Rubio solved a series of 20 examples in his article (Five different initial and final configurations * four different environments). These examples have been solved using the GA procedure presented in this thesis. The comparison of the results is illustrated in the following graphs:



Graph 4.1: Execution Time Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Harmonic Interpolation Functions.

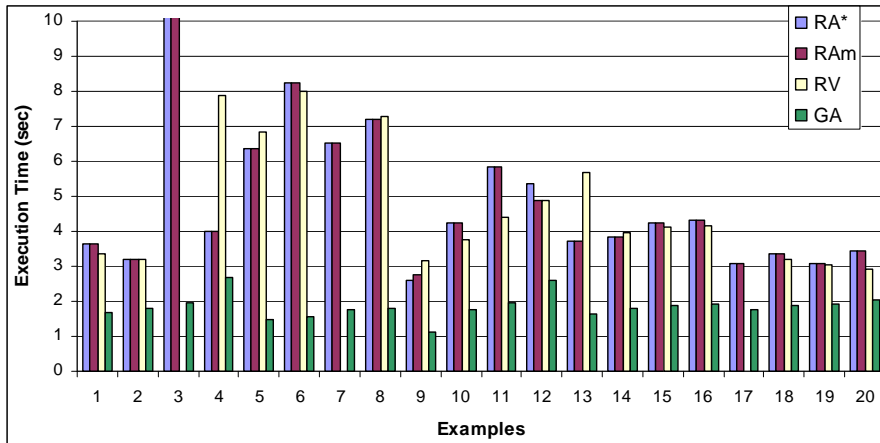


Graph 4.2: Travelled Distance Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Harmonic Interpolation Functions.

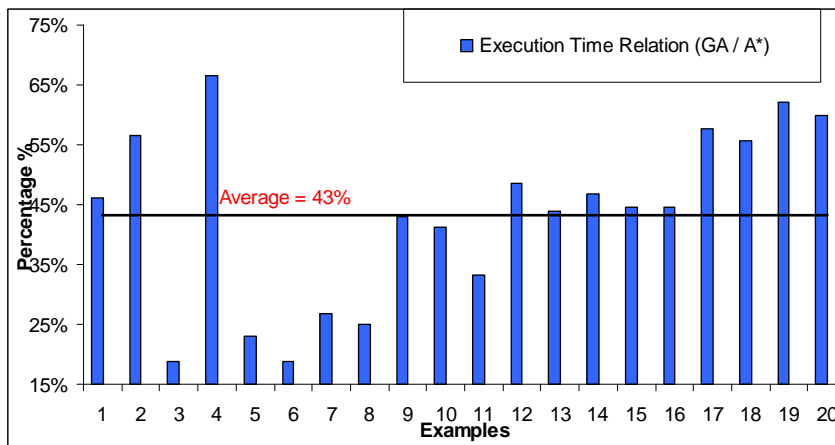


Graph 4.3: Computational Time Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Harmonic Interpolation Functions.

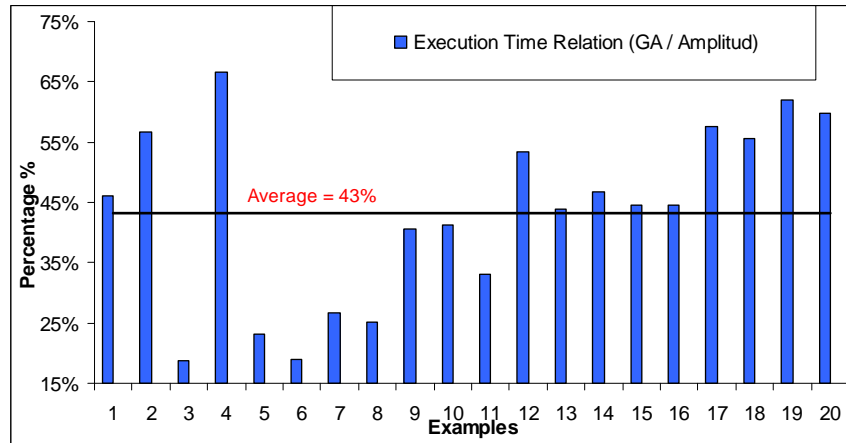
Rubio 2006, in his doctoral thesis solved the same series of examples using cubic polynomial functions with zero velocities at the ends of intermediate configurations trajectories. Then he adjusted the trajectory to obtain a continuous velocities and accelerations between intermediate configurations. The comparison between Rubio procedure and GA procedure presented in this thesis will be as following:



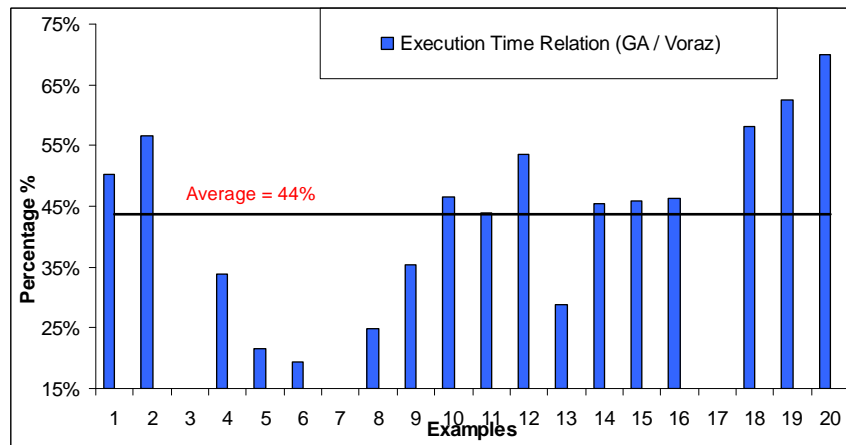
Graph 4.4: Execution Time Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Cubic Polynomial Interpolation Functions.



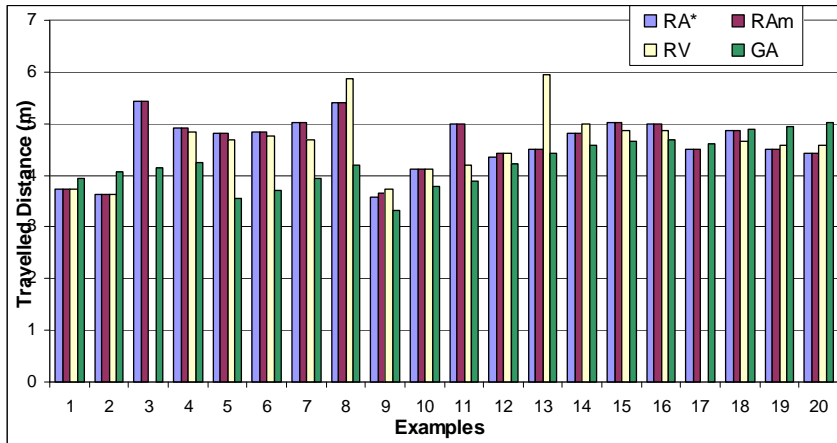
Graph 4.5: Execution Time Comparison Between GA Procedure & A* Procedure Produced by Rubio 2006.



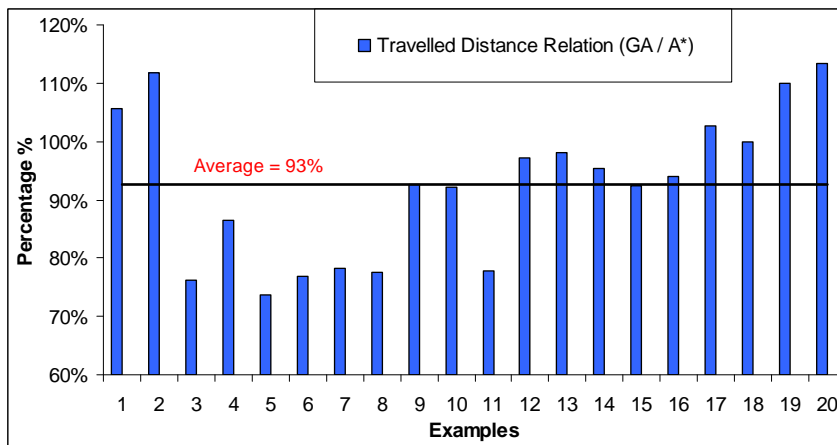
Graph 4.6: Execution Time Comparison Between GA Procedure & Amplitude Procedure Produced by Rubio 2006.



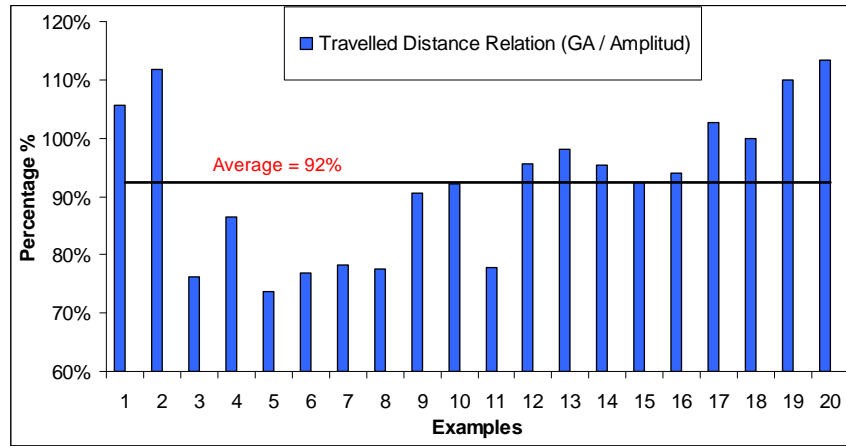
Graph 4.7: Execution Time Comparison Between GA Procedure & Voraz Procedure Produced by Rubio 2006.



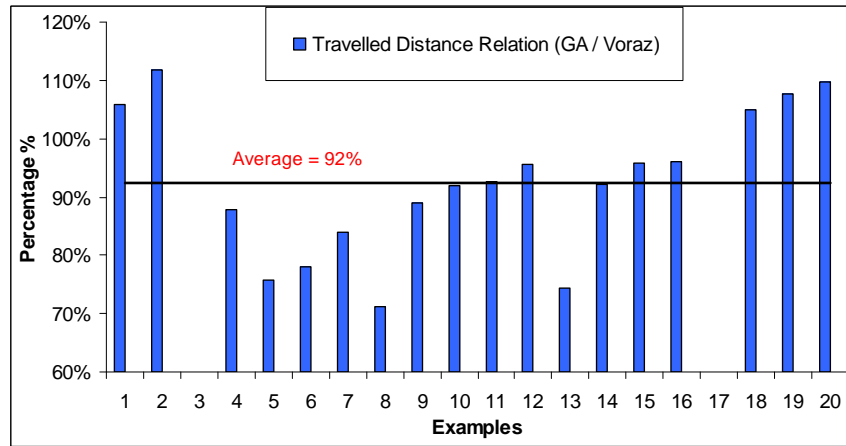
Graph 4.8: Travelled Distance Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Cubic Polynomial Interpolation Functions.



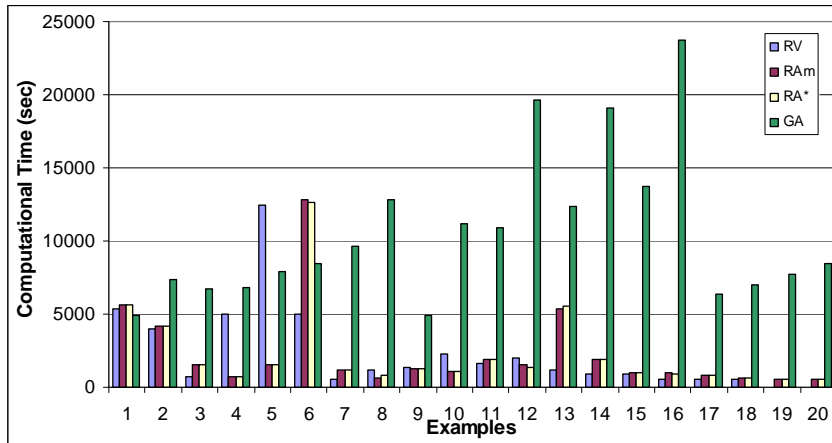
Graph 4.9: Travelled Distance Comparison Between GA Procedure & A* Procedure Produced by Rubio 2006.



Graph 4.10: Travelled Distance Comparison Between GA Procedure & Amplitud Procedure Produced by Rubio 2006.



Graph 4.11: Travelled Distance Between GA Procedure & Voraz Procedure Produced by Rubio 2006.



Graph 4.12: Computational Time Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Cubic Polynomial Interpolation Functions.

Observing the comparison of results of the 20 examples, you can see clearly that the main objective of this work “execution time” obtained by the GA procedure is better than the one published by Rubio 2006 and Rubio et al. 2009b. However, the traveling distance by significant points in some cases is not better than the ones obtained by him. In addition, the computational time is very high in all cases, which may be considered as one of the main disadvantages of the genetic algorithm.

4.3.2. Example 2: Indirect and Direct Methods Comparison

In this Example, a comparison between the indirect method explained in Chapter 3 and the method obtained in this Chapter will be demonstrated. The robot initial and final configurations and obstacles are shown in the previous chapter Table (3.1) and (3.2) respectively. The results of the indirect method are tabulated in Table (3.3) (the case with zero obstacles and spherical obstacles). The

corresponding results for the same example using the direct method are tabulated in Table (4.3).

		Execution Time (s)	Travelled Distance (m)	Computational Time (s)
Indirect (Chapter 3)	0 Obstacle	2,00408	3,7358	13413
	1 Obstacles	2,11442	3,8029	19805
	2 Obstacles	2,37418	4,0187	16292
	3 Obstacles	3,36064	4,1585	21408
Direct (Chapter 4)	0 Obstacle	1,68652	3,9446	4876
	1 Obstacles	1,80759	4,0642	7321
	2 Obstacles	1,96055	4,1335	6749
	3 Obstacles	2,67079	4,2554	6799

Table 4.3: Comparison Between Direct and Indirect Method.

As shown in the table, the trajectory time obtained from the direct method (the method obtained in this chapter) is better than the trajectory time obtained from the indirect one. This is because the direct method is based on the minimum time trajectory between adjacent configurations.

4.3.3. Example 3: Industrial Application – Comparison Results

This example demonstrates the effectiveness of the mentioned algorithm. The next Figure (4.4) shows the robot in the initial and final configuration. This example also was solved by Rubio et al. 2009b. The robot initial and final configurations are shown in Table (3.4). Obstacles are shown in Table (3.5).

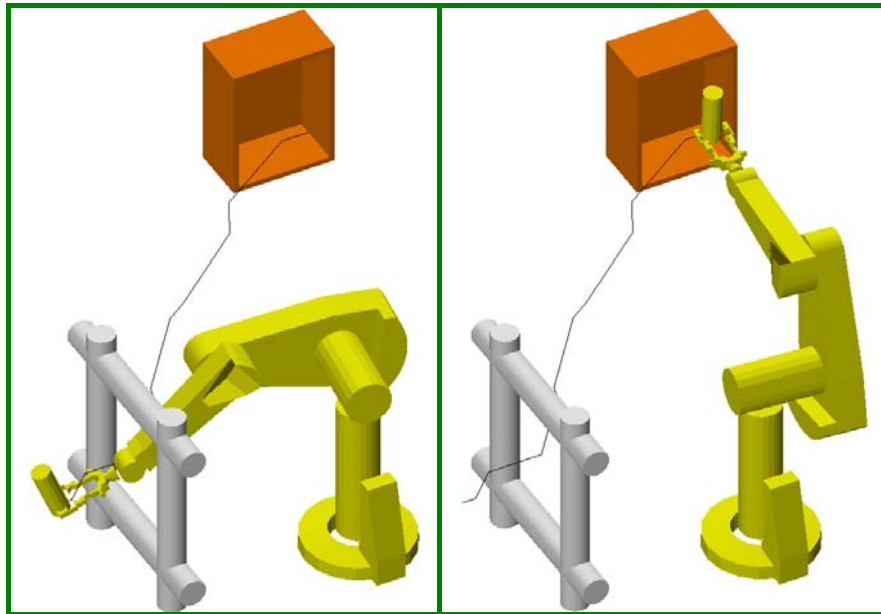


Figure 4.4: Example 3.

The comparison results for this example between the proposed approach and the procedure introduced by Rubio are shown in the next Table (4.4):

		Results of this thesis	Rubio et al. 2009b Results		
			A *	uniform cost	greedy
0 Obstacles	t_e (s)	1.42842	35.61	29.23	45.70
	d_s (m)	4.2556	5.82	5.41	5.43
	d_e (m)	1.7389			
	t_c (s)	28885	17049.94	16233.08	2674.69

Table 4.4: Example 3 Results.

4.3.4. Example 4: Industrial Application – Case With and Without Obstacles

This example demonstrates the effectiveness of the mentioned algorithm and its ability to adapt in absence or presence of obstacle. The robot initial and final configurations are shown in Table (4.5). Obstacles are shown in Table (4.6).

Joint No.	1	2	3	4	5	6
Initial configuration	-108.54	-9.88	194.36	-15.98	-86.93	0.00
Final configuration	0.13	-102.36	191.40	0.00	12.47	0.00

Table 4.5: Initial and Final Configurations for Example 4.

	1 st Cylindrical obstacle	2 nd Cylindrical obstacle	3 rd Cylindrical obstacle	4 th Cylindrical obstacle
Centre 1	$C_1^{Cyl,1} = (-0.30, -0.70, 0.00)$	$C_2^{Cyl,1} = (-0.30, -0.70, 0.00)$	$C_3^{Cyl,1} = (-0.30, -0.70, 0.58)$	$C_4^{Cyl,1} = (0.30, -0.70, 0.00)$
Centre 2	$C_1^{Cyl,2} = (0.30, -0.70, 0.00)$	$C_2^{Cyl,2} = (-0.30, -0.70, 0.58)$	$C_3^{Cyl,2} = (0.30, -0.70, 0.58)$	$C_4^{Cyl,2} = (0.30, -0.70, 0.58)$
Radius	$r_1^{Cyl} = 0.15$	$r_2^{Cyl} = 0.15$	$r_3^{Cyl} = 0.15$	$r_4^{Cyl} = 0.15$
	1 st Quadri-lateral obstacle	2 nd Quadri-lateral obstacle	3 rd Quadri-lateral obstacle	4 th Quadri-lateral obstacle
Point 1	$P_{11} = (0.45, 0.34, 0.89)$	$P_{21} = (0.45, 0.34, 1.38)$	$P_{31} = (0.45, -0.0400, 0.89)$	$P_{41} = (0.45, 0.34, 0.8900)$
Point 2	$P_{12} = (0.45, -0.04, 0.89)$	$P_{22} = (0.45, -0.04, 1.38)$	$P_{32} = (0.67, -0.04, 0.89)$	$P_{42} = (0.45, 0.34, 1.38)$
Point 3	$P_{13} = (0.67, -0.04, 0.89)$	$P_{23} = (0.67, -0.04, 1.38)$	$P_{33} = (0.67, -0.04, 1.38)$	$P_{43} = (0.67, 0.34, 1.38)$
Point 4	$P_{14} = (0.67, 0.34, 0.89)$	$P_{24} = (0.67, 0.34, 1.38)$	$P_{34} = (0.45, -0.04, 1.38)$	$P_{44} = (0.67, 0.34, 0.89)$

Table 4.6: Obstacles Locations (in m) for Example 4.

The results for this example in case of obstacle or without obstacles are shown in Table (4.7):

	t_e (s)	t_c (s)	d_e (m)	d_s (m)
0 Obstacles	2.42217	12915	1.7106	4.7870
With Obstacles	3.85854	57080	1.7802	5.2227

Table 4.7: Example 4 Results.

The next Figure (4.5) shows the robot in the initial and final configuration, the end-effector track, and the workspace. The left one is the final and the right one is the initial.

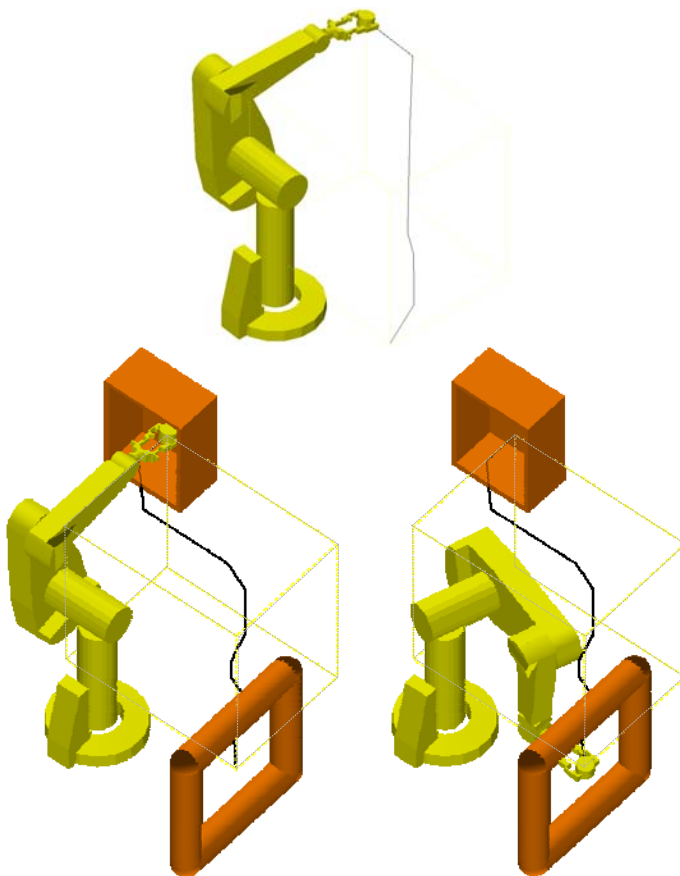


Figure 4.5: Example 4 (Case With and Without Obstacles).

4.3.5. Example 5: Typical Industrial Application

Here you can see the ability of the algorithm to modify the search space to adapt the robot to find the trajectory between the initial and final configurations.

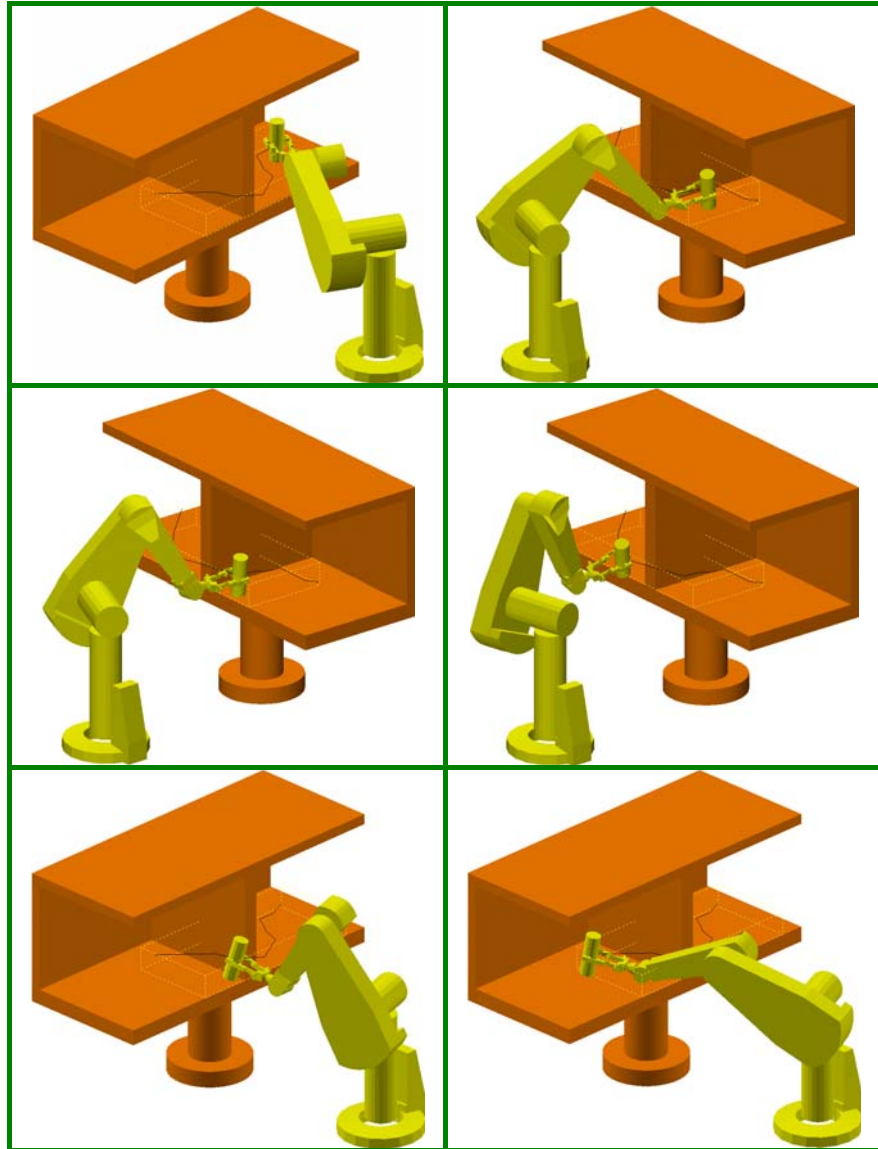


Figure 4.6: Example 4.

t_e (s)	t_c (s)	d_e (m)	d_s (m)
1.52102	9720-2173	1.1335	3.2619

Table 4.8: Example 4 Results.

4.4. DISCUSSION OF RESULTS

The examples illustrated in previous section prove the ability of the presented procedure to solve the trajectory-planning problem for industrial robots.

Using cubic polynomial as interpolation functions demonstrate results better than harmonic functions, and the proposed GA procedure provides minimum time trajectory better than the A* and Amplitude procedures produced by Rubio 2006 and Rubio et al. 2009b.

The computation time in all examples is high which may considered as the main disadvantage of the genetic algorithm in general.

The traveled distance by the significant points of the robot, are very acceptable, and by comparing the results with other works, in the most of cases, the traveled distance of the presented procedure is better than the ones of the other works.

The presented procedure shows a significant ability to adapt the robot and its trajectory to any workspace characteristics.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this thesis, new approaches have been presented to solve the path and trajectory planning problems for industrial robots operating in 3D complex environment. Genetic algorithms appear in this thesis to solve such problems. These approaches have led to two general classes of algorithms that are capable of obtaining the solution of the mentioned planning problems. These classes are:

- 1) An algorithm to solve the adjacent configuration problem. This algorithm in fact has two versions;
 - a) Version for solving the adjacent configuration for path planning considering just the kinematics, geometric, and obstacles constraints.
 - b) Version for solving the adjacent configuration problem for the trajectory planning considering dynamic constraints of the robotic system.
- 2) An algorithm to solve path and trajectory planning problems. This algorithm also has two versions.

- a) Version for path planning. This version aims to find the shortest path between two robot configurations subjected to the kinematics, geometric, and obstacle avoidance constraints.
- b) Version for trajectory planning. This version aims to find the minimum time trajectory between two robot configurations subjected to the dynamics constraints of the robotic system and the obstacle avoidance constraints.

The path planning algorithm aims to find the shortest path between two given robot configurations; initial configuration C^i and final configuration C^f ; avoiding the collision with obstacles in the workspace. In an indirect way, the minimum time trajectory has been calculated in this case by adjusting a trajectory to the resulting path. This could be achieved by building the clamped cubic spline algorithm and solving it by genetic algorithm procedure.

The trajectory planning algorithm aims to minimize the trajectory time needed to move the robot from an initial configuration C^i to a final configuration C^f avoiding the collision with obstacles in the workspace. The workspace has been built in a way that gives the capability to modify its dimensions if there is no feasible solution in the current one. The effectiveness of this technique has been shown clearly in the experimental results. A new crossover and mutation operators have been designed in a way to improve the solution and its quality.

The presented algorithms can be applied to any industrial robotic system. In this thesis, an application example has been developed using Puma 560 robot for testing the algorithms. In addition, an application program using object oriented C++ has been built in a way to simulate the dynamics and kinematic (direct and inverse) of the Puma 560 robot.

The presented algorithms have been tested and validated using a large number of examples. The analysis of the results shed light on the characteristics and properties of the algorithms used, and are reflected in two chapters of this document. Part of these examples is compared with the work of other authors and demonstrates the efficiency of the proposed procedure by an average of 81.3% for path planning (see sub-Section Example 1:3.5.1.2) and 43% for trajectory planning (see sub-Section 4.3.1). Another part of the examples is done using close to real industrial scenario to show the ability of the presented algorithm to adapt to any workspace.

An important parameter should be discussed here is the computational time. As shown in the illustrated example the computational time is relevantly high. Moreover, the computational time increases as well as the restrictions increase. Furthermore, the number of individuals should be increased, and so the number of generations for more accurate results using the GA procedure, especially for more complex problems and workspaces, which leads to an increase of the computational time. This maybe considered as a disadvantage of the GA in general. However, as the industrial robots work on a repetitive trajectories and paths, an offline planning normally takes place. This means that the computational time cost can be acceptable as the resulting planning and time trajectory are good enough.

Finally, because of the importance of the path and trajectory planning problems in the industry, it is necessary to introduce new operating assumptions to improve the quality and the functionality of the presented algorithms. These assumptions are:

- Adding a new optimization algorithm to deal with the orientation of the Gripper of the robot to achieve some tasks.

- Rebuilding the existing algorithms using Ant-Colony optimization instead of genetic algorithms could be very interesting. Knowing that, ants are very powerful in finding the shortest path and minimum time trajectory between their colonies and the food. Many authors like Liu et al. 2005 and Maurya and Shukla 2010 used the ant colony optimization procedure to solve the path and trajectory planning for mobile robots. It will be an opportunity to check their efficiency in planning paths and trajectories for industrial robots.

While there is considerable work yet to be addressed, this thesis provides useful approaches to deal with path and trajectory planning for industrial robots.

CHAPTER 6

REFERENCES

- K. Abdel-Malek, Z. Mi, J. Yang and K. Nebel, "Optimization-Based Trajectory Planning of the Human Upper Body", *Robotica*, vol. 24, pp. 683 - 696, 2006.
- F. Abu-Dakka, F. J. Valero and V. Mata, "Obtaining Adjacent Configurations with Minimum Time Considering Robot Dynamics Using Genetic Algorithm", presented at 17th International Workshop on Robotics in Alpe-Adria-Danube Region, Ancona, Italy, 2008.
- F. Abu-Dakka, F. J. Valero, A. Tubaileh and F. Rubio, "Obtaining Adjacent Configurations with Minimum Time Considering Robot Dynamics", presented at The 12th World Congress in Mechanism and Machine Science, Besançon, France, 2007.
- N. Ahmed, *Robot Motion Planning*. Department of Computer Science, James Cook University of North Queensland. Townsville, Queensland, Australia, 1997.
- J. M. Ahuactzin, E.-G. Talbi, P. Bessiere and E. Mazer, "Using Genetic Algorithms for Robot Motion Planning", presented at 10th European Conference on Artificial Intelligence, London, England, pp.671-675, 1992.

- S. Akella and J. Peng, "Time-Scaled Coordination of Multiple Manipulators", presented at IEEE International Conference on Robotics and Automation, New Orleans, LA, pp.3337- 3344, 2004.
- N. M. Amato, O. B. Bayazit, L. K. Dale, J. Jones and D. Vallejo, "Obprm: An Obstacle-Based Prm for 3d Workspaces", presented at The 3rd Workshop on The Algorithmic Foundations of Robotics on Robotics, Houston, Texas, US, pp.155-168, 1998.
- N. M. Amato and Y. Wu, "A Randomized Roadmap Method for Path and Manipulation Planning", presented at IEEE International Conference on Robotics and Automation, Minneapolis, MN, pp.113-120, 1996.
- J. Angeles, *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*, 2nd ed. New York, Springer-Verlag, 1997.
- J. Angeles, A. Alivizatos and P. J. Zsombor-Murray, "The Synthesis of Smooth Trajectories for Pick-and-Place Operations", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, pp. 173-178, 1988.
- T. Asano, T. Asano, L. Guibas, J. Hershberger and H. Imai, "Visibility of Disjoint Polygons", *Algorithmica*, vol. 1, pp. 49-63, 1986.
- F. Avnaim, J. D. Boissonnat and B. Faverjon, "A Practical Exact Motion Planning Algorithm for Polygonal Objects Amidst Polygonal Obstacles", presented at IEEE International Conference on Robotics and Automation, Philadelphia, PA, pp.1656-1661, 1988.
- M. Barbehenn and S. Hutchinson, "Toward an Exact Incremental Geometric Robot Motion Planner", presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburgh, PA, pp.39-44, 1995.
- J. Barraquand, L. E. Kavraki, J. C. Latombe, R. Motwani, T.-Y. Li and P. Raghavan, "A Random Sampling Scheme for Path Planning", *The International Journal of Robotics Research*, vol. 16, pp. 759-774, 1997.
- J. Barraquand, B. Langlois and J. C. Latombe, "Numerical Potential Field Techniques for Robot Path Planning", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, pp. 224-241, 1992.
- J. Barraquand and J. C. Latombe, "Robot Motion Planning: A Distributed Representation Approach", Department of Computer Science, Stanford University, Report No. STAN-CS-89-1257, 1989.

- J. Barraquand and J. C. Latombe, "A Monte-Carlo Algorithm for Path Planning with Many Degrees of Freedom", presented at IEEE International Conference on Robotics and Automation, Cincinnati, OH, pp.1712-1717, 1990.
- J. Barraquand and J. C. Latombe, "Robot Motion Planning: A Distributed Representation Approach", *The International Journal of Robotics Research*, vol. 10, pp. 628-649, 1991.
- R. H. Bartels, J. C. Beatty and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, New ed. San Mateo, CA, Morgan Kaufmann, 1987.
- S. Behzadipour and A. Khajepour, "Time-Optimal Trajectory Planning in Cable-Based Manipulators", *IEEE Transactions on Robotics*, vol. 22, pp. 559-563, 2006.
- F. Benimeli, *Estimación De Parametros Dinámicos En Robots Manipuladores*. Mechanical Engineering Department, Universidad Politécnica de Valencia. Valencia, 2006.
- E. Bertolazzi, F. Biral and M. D. L. M, "Real-Time Motion Planning for Multibody Systems - Real Life Application Examples", *Multibody System Dynamics*, vol. 17, pp. 119-139, 2007.
- Z. Bien and J. Lee, "A Minimum-Time Trajectory Planning Method for Two Robots", *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 414-418, 1992.
- J. E. Bobrow, S. Dubowsky and J. S. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths", *International Journal of Robotics Research*, vol. 4, pp. 3-17, 1985.
- R. Bohlin and L. E. Kavraki, "Path Planning Using Lazy Prm", presented at IEEE International Conference on Robotics & Automation, San Francisco. CA, pp.521-528, 2000.
- V. Boor, M. H. Overmars and A. F. v. d. Stappen, "The Gaussian Sampling Strategy for Probabilistic Roadmap Planners", presented at IEEE International Conference on Robotics and Automation, Detroit, MI, USA, pp.1018-1023, 1999.

- R. A. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space", *IEEE Systems, Man and Cybernetics*, vol. SMC-13, pp. 190–197, 1983a.
- R. A. Brooks, "Planning Collision- Free Motions for Pick-and-Place Operations", *The International Journal of Robotics Research*, vol. 2, pp. 19-44, 1983b.
- R. A. Brooks and T. Lozano-Pérez, "A Subdivision Algorithm in Configuration Space for Findpath with Rotation", presented at The 8th International Joint Conference on Artificial Intelligence, Karlsruhe, W. Germany, pp.799-806, 1983.
- S. J. Buckley, "Fast Motion Planning for Multiple Moving Robots", presented at the IEEE International Conference on Robotics and Automation, Scottsdale, AZ, USA, pp.322-326, 1989.
- J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MIT Press, 1988.
- J. F. Canny, A. Rege and J. H. Reif, "An Exact Algorithm for Kinodynamic Planning in the Plane", presented at The 6th Annual Symposium on Computational Geometry, Berkley, California, USA, pp.271 - 280, 1990.
- J. F. Canny, J. H. Reif, B. R. Donald and P. Xavier, "On the Complexity of Kinodynamic Planning", presented at The 29th Annual Symposium on Foundations of Computer Science, White Plains, NY, USA, pp.306-316, 1988.
- B. Cao, G. I. Dodds and G. W. Irwin, "Time-Optimal and Smooth Constrained Path Planning for Robot Manipulators", presented at IEEE International Conference on Robotics and Automation, San Diego, CA, USA, pp.1853–1858, 1994.
- S. Caselli and M. Reggiani, "Erpp: An Experience-Based Randomized Path Planner", presented at IEEE International Conference on Robotics & Automation, San Francisco, CA, USA, pp.1002-1008, 2000.
- S. Caselli, M. Reggiani and R. Rocchi, "Heuristic Methods for Randomized Path Planning in Potential Fields", presented at IEEE International Symposium on Computational Intelligence in Robotics and Automation, Banff, Alberta, Canada, pp.426- 431, 2001.

- C. I. Connolly, J. B. Burns and R. Weiss, "Path Planning Using Laplace's Equation", presented at IEEE International Conference on Robotics and Automation, Cincinnati, OH, pp.2102-2106, 1990.
- D. Constantinescu and E. A. Croft, "Smooth and Time-Optimal Trajectory Planning for Industrial Manipulators Along Specified Paths", *Journal of Robotic Systems*, vol. 17, pp. 233-249, 2000.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 2 ed., The MIT Press, 2001.
- J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed., Prentice Hall, 2005.
- H. Chang and T.-Y. Li, "Assembly Maintainability Study with Motion Planning", presented at IEEE International Conference on Robotics and Automation, Nagoya, Japan, pp.1012-1019, 1995.
- R. Chatila, "Path Planning and Environment Learning in a Mobile Robot System", *Journal: European Conference on Artificial Intelligence*, 1982.
- P. C. Chen and Y. K. Hwang, "Sandros: A Motion Planner with Performance Proportional to Task Difficulty", presented at IEEE International Conference on Robotics and Automation, Nice, France, pp.2346-2353, 1992.
- P. C. Chen and Y. K. Hwang, "Sandros: A Dynamic Graph Search Algorithm for Motion Planning", *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 390-403, 1998.
- Y. C. Chen, "Solving Robot Trajectory Planning Problems with Uniform Cubic B-Splines", *Optimal Control Applications & Methods*, vol. 12, pp. 247-262, 1991.
- Y. C. Chen and A. A. Desrochers, "A Proof of the Structure of the Minimum-Time Control Law of Robotic Manipulators Using a Hamiltonian Formulation", *IEEE Transactions on Robotics and Automation*, vol. 6, pp. 388-393, 1990.
- Y. C. Chen and M. Vidyasagar, "Optimal Trajectory Planning for Planar N-Link Revolute Manipulators in the Presence of Obstacles", presented at IEEE International Conference on Robotics and Automation, Philadelphia, pp.202-208, 1988.

- T. Chettibi, H. Lehtihet, M. Haddad and S. Hanchi, "Minimum Cost Trajectory Planning for Industrial Robots", *European Journal of Mechanics a-solids*, vol. 23, pp. 703-715, 2004.
- Y.-K. Choi, J.-H. Park, H.-S. Kim and J. H. Kim, "Optimal Trajectory Planning and Sliding Mode Control for Robots Using Evolution Strategy", *Robotica*, vol. 18, pp. 423-428, 2000.
- H. Choset and J. Burdick, "Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph", presented at Workshop on Algorithmic Foundations of Robotics, 1996.
- H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.
- J.-H. Chuang, "Potential-Based Modeling of Three-Dimensional Workspace for Obstacle Avoidance", *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 778-785, 1998.
- J.-H. Chuang and N. Ahuja, "Path Planning Using the Newtonian Potential", presented at IEEE International Conference on Robotics and Automation, Sacramento, CA, pp.558-563, 1991.
- Y. Davidor, *Genetic Algorithms and Robotics*. World Scientific Publishing Co., Inc., 1991.
- J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", *Journal of Applied Mechanics*, vol. 22, pp. 215-221, 1955.
- E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.
- B. R. Donald, "Motion Planning with Six Degrees of Freedom", MIT Artificial Intelligence Lab., Massachusetts Institute of Technology, Report no. AITR-791, May 1984.
- A. H. Eltimsahy and W. S. Yang, "Near Minimum-Time Control of Robotic Manipulator with Obstacles in the Workspace", presented at IEEE International Conference on Robotics and Automation, Philadelphia, pp.358-363, 1988.

- M. Erdmann and T. Lozano-Pérez, "On Multiple Moving Objects", MIT, Artificial Intelligence Laboratory, Report no.: AIM-883, 1-May 1986.
- B. Faverjon and P. Tournassoud, "A Local Based Approach for Path Planning of Manipulators with a High Number of Degrees of Freedom", presented at IEEE International Conference on Robotics and Automation, Raleigh, NC, pp.1152- 1159, 1987.
- B. Faverjon and P. Tournassoud, "A Practical Approach to Motion-Planning for Manipulators with Many Degrees of Freedom", Research Report, RR-0951, 1988.
- P. W. Finn, L. E. Kavraki, J. C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian and A. Yao, "Rapid: Randomized Pharmacophore Identification for Drug Design", presented at The 13th Annual Symposium on Computational Geometry, Nice, France, pp.324 - 333, 1997.
- P. Fiorini and Z. Shiller, "Robot Motion Planning in Dynamic Environments", presented at International Symposium of Robotic Research, Munich, Germany, pp.237-248, 1995.
- P. Fiorini and Z. Shiller, "Time Optimal Trajectory Planning in Dynamic Environments", presented at IEEE International Conference on Robotics and Automation, Minneapolis, MN, pp.1553 - 1558, 1996.
- S. Fortune, G. Wilfong and C. K. Yap, "Coordinated Motion of Two Robot Arms", presented at IEEE International Conference on Robotics and Automation, pp.1216-1223, 1986.
- T. Fraichard, "Dynamic Trajectory Planning with Dynamic Constraints: A 'State-Time Space' Approach", presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Yokohama, Japan, pp.1394-1400, 1993.
- T. Fraichard, "Trajectory Planning Amidst Moving Obstacles: Path-Velocity Decomposition Revisited", *Journal of the Brazilian Computer Society*, vol. 4, 1998.
- T. Fraichard, "Trajectory Planning in a Dynamic Workspace: A 'State-Time Space' Approach", *Advanced Robotics*, vol. 13, pp. 75-94, 1999.
- T. Fraichard and C. Laugier, "Dynamic Trajectory Planning, Path-Velocity Decomposition and Adjacent Paths", presented at The International Joint

- Conference on Artificial Intelligence, Chambéry, France, pp.1592-1597, 1993.
- T. Furukawa, "Time-Subminimal Trajectory Planning for Discrete Non-Linear Systems", *Journal of Engineering Optimization*, vol. 34, pp. 219-243(25), 2002.
- A. Gasparetto and V. Zanutto, "A New Method for Smooth Trajectory Planning of Robot Manipulators", *Mechanism and Machine Theory*, vol. 42, pp. 455–471, 2007.
- H. P. Geering, L. Guzzella, S. A. R. Hepner and C. H. Onder, "Time-Optimal Motions of Robots in Assembly Tasks", *IEEE Transactions on Automatic Control*, vol. AC-31, pp. 512-518, 1986.
- S. K. Ghosh and D. M. Mount, "An Output-Sensitive Algorithm for Computing Visibility Graphs", presented at the 28th IEEE Symposium on Foundations of Computer Science, Los Angeles, pp.11-19, 1987.
- B. Glavina, "Solving Findpath by Combination of Goal-Directed and Randomized Search", presented at IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, pp.1718-1723, 1990.
- H. H. González-Baños, D. Hsu and J. C. Latombe, "Motion Planning: Recent Developments", in *Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications*, S. S. Ge and F. L. Lewis, Ed ^Eds. CRC Press, 2006.
- L. Gouzènes, "Strategies for Solving Collision-Free Trajectories Problems for Mobile and Manipulator Robots", *The International Journal of Robotics Research*, vol. 3, pp. 51-65, 1984.
- T. A. Harden, *Minimum Distance Influence Coefficients for Obstacle Avoidance in Manipulator Motion Planning*. The University of Texas. Austin, 2002.
- P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100-107, 1968.
- C. Helguera and S. Zegloul, "A Local-Based Method for Manipulators Path Planning in Heavycluttered Environments", presented at IEEE International Conference on Robotics & Automation, San Francisco, CA, USA, pp.3467-3472, 2000.

- P. Henrici, *Essentials of Numerical Analysis with Pocket Calculator Demonstrations*, First ed., John Wiley & Sons Inc, 1982.
- D. Henrich, C. Wurll and H. Worn, "On-Line Path Planning by Heuristic Hierarchical Search", presented at The 24th Annual Conference of the IEEE Industrial Electronics Society, Aachen, Germany, pp.2239-2244, 1998.
- J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.
- J. M. Hollerbach, "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity", *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, vol. SMC-10, pp. 730-736, 1980.
- J. M. Hollerbach, "Dynamic Scaling of Manipulator Trajectories", MIT Artificial Intelligence Laboratory, Report no.: AIM-700, 1-Jan 1983.
- J. E. Hopcroft, D. A. Joseph and S. H. Whitesides, "Movement Problems for 2-Dimensional Linkages", *SIAM Journal on Computing*, vol. 13, pp. 610-629, 1984.
- J. E. Hopcroft, J. T. Schwartz and M. Sharir, "On the Complexity of Motion Planning for Multiple Independent Objects; Pspace- Hardness of the "Warehouseman's Problem"", *The International Journal of Robotics Research*, vol. 3, pp. 76-88, 1984.
- D. Hsu, L. E. Kavraki, J. C. Latombe, R. Motwani and S. Sorkin, "On Finding Narrow Passages with Probabilistic Roadmap Planners", presented at The 3rd International Workshop on Algorithmic Foundations of Robotics (WAFR), 1998.
- D. Hsu, R. Kindel, J. C. Latombe and S. Rock, "Randomized Kinodynamic Motion Planning with Moving Obstacles", *International Journal of Robotics Research*, vol. 21, pp. 233-255, 2002.
- D. Hsu, J. C. Latombe and R. Motwani, "Path Planning in Expansive Configuration Spaces", presented at IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, pp.2719-2726, 1997.

- Y. K. Hwang and N. Ahuja, "Gross Motion Planning a Survey", *ACM Computing Surveys*, vol. 24, pp. 219–291, 1992a.
- Y. K. Hwang and N. Ahuja, "A Potential Field Approach to Path Planning", *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 23-32, 1992b.
- Y. K. Hwang and P. C. chen, "A Heuristic and Complete Planner for the Classical Mover's Problem", presented at IEEE International Conference on Robotics and Automation, Nagoya, Japan, pp.729-736, 1995.
- P. Isto, "Path Planning by Multiheuristic Search Via Subgoals", presented at the 27th International Symposium on Industrial Robots, CEU, Milan, pp.712-726, 1996.
- P. Isto, "A Two-Level Search Algorithm for Motion Planning", presented at IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, pp.2025-2031, 1997.
- E. Jamhour and P. J. Andre, "Planning Smooth Trajectories Along Parametric Paths", *Mathematics and Computers in Simulation*, vol. 41, pp. 615-626(12), 1996.
- R. A. Jarvis and J. C. Byrne, "Robot Navigation: Touching, Seeing and Knowing", presented at The 1st Australian Conference on Artificial Intelligence, 1986.
- M. E. Kahn and B. Roth, "The near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains", *Journal of Dynamic Systems, Measurement, and Control*, vol. 93, pp. 164-172, 1971.
- K. Kant and S. W. Zucker, "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition", *International Journal of Robotics Research*, vol. 5, pp. 72-89, 1986.
- L. E. Kavraki, *Random Networks in Configuration Space for Fast Path Planning*. Department of Computer Science, Stanford University. Stanford, California, 1995.
- L. E. Kavraki, M. N. Kolountzakis and J. C. Latombe, "Analysis of Probabilistic Roadmaps for Path Planning", *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 166-171, 1998a.

- L. E. Kavraki and J. C. Latombe, "Randomized Preprocessing of Configuration Space for Fast Path Planning", presented at IEEE International Conference on Robotics and Automation, San Diego, CA, pp.2138–2145, 1994a.
- L. E. Kavraki and J. C. Latombe, "Randomized Preprocessing of Configuration Space for Path Planning:Articulated Robots", presented at International Conference on Intelligent Robots and Systems, Munich, Germany, pp.1764-1772, 1994b.
- L. E. Kavraki and J. C. Latombe, "Probabilistic Roadmaps for Robot Path Planning", in *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, K. Gupta and A. P. Pobil, Ed ^Eds. John Wiley & Sons, Inc., 1998, pp. 33-53.
- L. E. Kavraki, J. C. Latombe, R. Motwani and P. Raghavan, "Randomized Query Processing in Robot Path Planning", *Journal of Computer and System Sciences*, vol. 57, pp. 50-60, 1998b.
- L. E. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces", *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566-580, 1996.
- K. Kedem and M. Sharir, "An Efficient Algorithm for Planning Collision-Free Translational Motion of a Convex Polygonal Object in 2-Dimensional Space Amidst Polygonal Obstacles", presented at The 1st Annual ACM Symposium on Computational Geometry, Baltimore, Maryland, pp.75 - 80, 1985.
- K. Kedem and M. Sharir, "An Automatic Motion Planning System for a Convex Polygonal Mobile Robot in 2-Dimensional Polygonal Space", presented at The 4th Annual ACM Symposium on Computational Geometry, Urbana-Champaign, Illinois, pp.329 - 340, 1988.
- O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *International Journal of Robotics Research*, vol. 5, pp. 90-98, 1986.
- A. R. Khoogar and J. K. Parker, "Obstacle Avoidance of Redundant Manipulators Using Genetic Algorithms", presented at IEEE Intemational Conference on Robotics and Automation, Williamsburg, VA, pp.317-320, 1991.

- P. Khosla and R. Volpe, "Superquadric Artificial Potential for Obstacle Avoidance and Approach", presented at IEEE International Conference on Robotics and Automation, Philadelphia PA, pp.1778-1784, 1988.
- J. Kieffer, A. J. Cahill and M. R. James, "Robust and Accurate Time-Optimal Path-Tracking Control for Robot Manipulators", *IEEE Transactions on Robotics and Automation*, vol. 13, pp. 880-890, 1997.
- B. K. Kim and K. G. Shin, "Minimum-Time Path Planning for Robot Arms and Their Dynamics", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, pp. 213-223, 1985.
- J. Kim and P. Khosla, "Real-Time Obstacle Avoidance Using Harmonic Potential Functions", *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 338-349, 1992.
- D. E. Koditschek, "Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations", presented at IEEE International Conference on Robotics and Automation, Raleigh, NC, pp.1- 6, 1987.
- Y. Koga, K. Kondo, J. Kuffner and J. C. Latombe, "Planning Motions with Intentions", presented at The 21st Annual Conference on Computer Graphics and Interactive Techniques, Orlando, Florida, pp.395-408, 1994.
- K. Kondo, "Motion Planning with Six Degrees of Freedom by Multistrategic Bidirectional Heuristic Free-Space Enumeration", *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 267-277, 1991.
- N. Kubota, T. Arakawa and T. Fukuda, "Motion Learning for Redundant Manipulator with Structured Intelligence", presented at The 24th Annual Conference on The IEEE Industrial Electronics Society (IECON '98), pp.104-109, 1998.
- J. Kuffner and J. C. Latombe, "Interactive Manipulation Planning for Animated Characters", presented at The 8th Pacific Conference on Computer Graphics and Applications, Hong Kong, China, pp.417-418, 2000.
- F. Lamiroux and J. Laumond, "On the Expected Complexity of Random Path Planning", presented at IEEE International Conference on Robotics and Automation, Minneapolis, MN, pp.3014-3019, 1996.
- J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

- J. C. Latombe, "Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts", *The International Journal of Robotics Research - Special Issue on Robotics at the Millennium*, vol. 18, pp. 1119-1128, 1999.
- S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning", Department of Computer Science, Iowa State University, Ames, IA, Technical Report 98-11, October 1998.
- S. M. LaValle and S. Hutchinson, "Optimal Motion Planning for Multiple Robots Having Independent Goals", presented at the IEEE International Conference on Robotics and Automation (ICRA), Minneapolis (USA), pp.1847-1852, 1996.
- S. M. LaValle and J. Kuffner, "Randomized Kinodynamic Planning", presented at IEEE International Conference on Robotics and Automation, Detroit, MI, USA, pp.473-479, 1999.
- J. Lengyel, M. Reichert, B. R. Donald and D. P. Greenberg, "Real-Time Robot Motion Planning Using Rasterizing Computer Graphics Hardware", presented at The 17th Annual Conference on Computer Graphics and Interactive Techniques, Dallas, TX, USA, pp.327 - 335, 1990.
- L. I. Liebermann and M. A. Wesley, "Autopass: An Automatic Programming System for Computer Controlled Mechanical Assembly", *IBM Journal of Research and Development*, vol. 21, pp. 321-333, 1977.
- C. Lin, P. Chang and J. Luh, "Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots", *IEEE Transactions on Automatic Control*, vol. 28, pp. 1066- 1074, 1983.
- G. Liu, T. Li, Y. Peng and X. Hou, "The Ant Algorithm for Solving Robot Path Planning Problem", presented at Third International Conference on Information Technology and Applications (ICITA'05), Sydney, Australia, pp.25-27, 2005.
- T. Lozano-Pérez, "The Design of a Mechanical Assembly System", Technical Report AI-TR 397, Artificial Intelligence Laboratory, MIT, 1976.
- T. Lozano-Pérez, "Automatic Planning of Manipulator Transfer Movements", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-11, pp. 681-698, Also MIT AI Memo 606, December 1980, 1981.

- T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach", *IEEE Transactions on Computers*, vol. C-32, pp. 108-120, 1983.
- T. Lozano-Pérez, "A Simple Motion Planning Algorithm for General Robot Manipulators", Massachusetts Institute of Technology, Cambridge, MA, AIM-896, 1986.
- T. Lozano-Pérez, "A Simple Motion Planning Algorithm for General Robot Manipulators", *IEEE Journal of Robotics and Automation*, vol. 3, pp. 224-238, 1987.
- T. Lozano-Pérez, J. Jones, E. Mazer, P. O'Donnell, W. Grimson, P. Tournassoud and A. Lanusse, "Handey: A Robot System That Recognizes, Plans, And. Manipulates", presented at IEEE International Conference on Robotics and Automation, Raleigh, NC, pp.843- 849, 1987.
- T. Lozano-Pérez and M. A. Wesley, "An Algorithm for Planning Collision-Free Path among Polyhedral Obstacles", *Communications of the ACM*, vol. 22, pp. 560-570, 1979.
- T. Lozano-Pérez and M. A. Wesley, "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles", *Communications of ACM*, vol. 22, pp. 560-570, 1979.
- J. Y. S. Luh, M. W. Walker and R. P. C. Paul, "On-Line Computational Scheme for Mechanical Manipulators", *Journal of Dynamic Systems, Measurement, and Control*, vol. 102, pp. 69-76, 1980.
- A. A. Maciejewski and J. J. Fox, "Path Planning and the Topology of Configuration Space", *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 444-456, 1993.
- V. Mata, S. Provenzano, F. Valero and J. I. Cuadrado, "Serial-Robot Dynamics Algorithms for Moderately Large Numbers of Joints", *Mechanism and Machine Theory*, vol. 37, pp. 739-755, 2002.
- R. Maurya and A. Shukla, "Generalized and Modified Ant Algorithm for Solving Robot Path Planning Problem", presented at Third IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, pp.643-646, 2010.
- E. Mazer, J. M. Ahuactzin and P. Bessiere, "The Ariadne's Clew Algorithm", *Journal of Artificial Intelligence Research*, vol. 9, pp. 295–316, 1998.

- J. M. McCarthy and J. E. Bobrow, "The Number of Saturated Actuators and Constraint Forces During Time-Optimal Movement of a General Robotic System", presented at IEEE International Conference on Robotics and Automation, Nice, France, pp.542-546, 1992.
- M. C. McHenry, *Slice-Based Path Planning*. University of Southern California. 1998.
- H. P. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. Department of Computer Science, Stanford University. Stanford, CA, 1980.
- N. J. Nilsson, "A Mobile Automaton: An Application of Artificial Intelligence Techniques", presented at 1st International Joint Conference on Artificial Intelligence, Washington D.C., pp.509-520, 1969.
- P. O'Donnell and T. Lozano-Pérez, "Deadlock-Free and Collision-Free Coordination of Two Robot Manipulators", presented at IEEE International Conference on Robotics and Automation, Scottsdale, AZ, USA, pp.484-489, 1989.
- C. O'Dunlaing, "Motion Planning with Inertial Constraints", NYU Robotics Laboratory, Technical Report TR-230, 1986.
- C. O'Dunlaing and C. K. Yap, "A "Retraction" Method for Planning the Motion of a Disc", *Journal of Algorithms*, vol. 6, pp. 104-111, 1982.
- K. Oh, J. P. Hwang, E. Kim and H. Lee, "Path Planning of a Robot Manipulator Using Retrieval Rrt Strategy", *TRANSACTIONS ON ENGINEERING, COMPUTING AND TECHNOLOGY*, vol. 19, pp. 171-174, 2007.
- D. E. Orin, R. B. McGhee, M. Vukobratovic and G. Hartoch, "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods", *Mathematical Biosciences*, vol. 43, pp. 107-130, 1979.
- M. H. Overmars, "A Random Approach to Motion Planning", Department of Computer Science, Utrecht University, Technical Report RUU-CS-92-32, 1992.
- M. H. Overmars and P. Svestka, "A Probabilistic Learning Approach to Motion Planning", Department of Computer Science, Utrecht University, Technical Report UU-CS-1994-03, Jan 1994.

- J. K. Parker and D. E. Goldberg, "Inverse Kinematics of Redundant Robots Using Genetic Algorithms", presented at IEEE International Conference on Robotics and Automation, Scottsdale, AZ, pp.271-276, 1989.
- F. Pfeiffer and R. Johanni, "A Concept for Manipulator Trajectory Planning", *IEEE Journal of Robotics and Automation*, vol. 3, pp. 115–123, 1987.
- A. Piazzzi and A. Visioli, "A Global Optimization Approach to Trajectory Planning for Industrial Robots", presented at The IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.1553-1559, 1997a.
- A. Piazzzi and A. Visioli, "A Cutting-Plane Algorithm for Minimum-Time Trajectory Planning of Industrial Robots", presented at The 36th Conference on Decision & Control, San Diego, California, USA, pp.1216-1218, 1997b.
- A. Piazzzi and A. Visioli, "An Interval Algorithm for Minimum-Jerk Trajectory Planning of Robot Manipulators", presented at The 36th Conference on Decision and Control, San Diego, California USA, pp.1924-1927, 1997c.
- A. Piazzzi and A. Visioli, "Global Minimum-Jerk Trajectory Planning of Robot Manipulators", *IEEE Transactions on Industrial Electronics*, vol. 47, pp. 140-149, 2000.
- I. T. Pietsch, O. Becker, M. Krefft and J. Hesselbach, "Time-Optimal Trajectory Planning for Adaptive Control of Plane Parallel Robots", presented at The 4th International Conference on Control and Automation (ICCA'03), Montreal, Canada, pp.639-643, 2003.
- I. T. Pietsch, M. Krefft, O. Becker, C. C. Bier and J. Hesselbach, "How to Reach the Dynamic Limits of Parallel Robots? An Autonomous Control Approach", *IEEE Transactions on Automation Science and Engineering*, vol. 2, pp. 369-380, 2005.
- L. d. Plessis and J. Snyman, "Trajectory-Planning through Interpolation by Overlapping Cubic Arcs and Cubic Splines", *International Journal for Numerical Methods in Engineering*, vol. 57, pp. 1615-1641, 2003.
- W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, Second ed., Cambridge University Press, 1992.

- S. Provenzano, *Aplicación De Las Ecuaciones De Gibbs-Appell a La Dinámica De Robots*. Mechanical Engineering Department, Universidad Politécnica de Valencia. Valencia, 2001.
- V. Rajan, "Minimum Time Trajectory Planning", presented at IEEE International Conference on Robotics and Automation, pp.759-764, 1985.
- A. Ram, R. Arkin, G. Boone and M. Pearce, "Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation", *Adaptive Behavior*, vol. 2, pp. 277 - 305, 1994.
- J. H. Reif, "Complexity of the Mover's Problem and Generalizations", presented at 20th IEEE Symposium on Foundations of Computer Science, San Juan, Puerto Rico, pp.421-427, 1979.
- J. H. Reif and M. Sharir, "Motion Planning in the Presence of Moving Obstacles", presented at 26th IEEE Symposium on Foundations of Computer Science, pp.144-154, 1985.
- J. H. Reif and J. A. Storer, "Shortest Paths in Euclidean Space with Polyhedral Obstacles", presented at Symposium on Mathematical Foundations of Computer Science, Czechoslovakia, 1988.
- J. H. Reif and J. A. Storer, "Shortest Paths in the Plane with Polygonal Obstacles", *Journal of the Association for Computing Machinery (JACM)*, vol. 41, pp. 982-1012, 1994.
- E. Rimon and D. E. Koditschek, "Exact Robot Navigation Using Cost Functions: The Case of Distinctspherical Boundaries in E^n ", presented at IEEE International Conference on Robotics and Automation, Philadelphia, PA, pp.1791-1796, 1988.
- P. F. Rowat, *Representing Spatial Experience and Solving Spatial Problems in a Simulated Robot Environment*. University of British Columbia. 1979.
- F. Rubio, *Planificación De Trayectorias De Robots Industriales. Análisis En Entornos Con Obstáculos*. Departamento de Ingeniería Mecánica y de Materiales, Universidad Politécnica de Valencia. Valencia, 2006.
- F. Rubio, F. Valero, J. L. Suñer and G. A., "The Simultaneous Algorithm and the Best Interpolation Function for Trajectory Planning", *Industrial Robot: An International Journal*, vol. 37, pp. 441-451, 2010.

- F. Rubio, F. Valero, J. L. Suñer and V. Mata, "Direct Step-by-Step Method for Industrial Robot Path Planning", *Industrial Robot: An International Journal*, vol. 36, pp. 594-507, 2009a.
- F. J. Rubio, F. J. Valero, J. L. Suñer and V. Mata, "Simultaneous Algorithm to Solve the Trajectory Planning Problem", *Mechanism and Machine Theory*, vol. 44, pp. 1910-1822, 2009b.
- S. F. P. Saramago and V. S. Junior, "Optimal Trajectory Planning of Robot Manipulators in the Presence of Moving Obstacles", *Mechanism and Machine Theory*, vol. 35, pp. 1079-1094(16), 2000.
- S. F. P. Saramago and V. Steffen, "Optimization of the Trajectory Planning of Robot Manipulators Taking into Account the Dynamics of the System", *Mechanism and Machine Theory*, vol. 33, pp. 883-894(12), 1998.
- S. F. P. Saramago and V. Steffen, "Dynamic Optimization for the Trajectory Planning of Robot Manipulators in the Presence of Obstacles", *Journal of the Brazilian Society of Mechanical Sciences*, vol. 21, pp. 372-383, 1999.
- S. F. P. Saramago and V. Steffen, "Trajectory Modeling of Robot Manipulators in the Presence of Obstacles", *Journal of Optimization Theory and Applications*, vol. 110, pp. 17-34, 2001.
- J. T. Schwartz and M. Sharir, "On the "Piano Movers" Problem. I: The Case of a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers", *Communications On Pure And Applied Mathematics*, vol. 36, pp. 345-398, 1983a.
- J. T. Schwartz and M. Sharir, "On the "Piano Movers" Problem. Ii. General Techniques for Computing Topological Properties of Real Algebraic Manifolds", *Advances in Applied Mathematics*, vol. 4, pp. 298-351, 1983b.
- J. T. Schwartz and M. Sharir, "On the Piano Movers' Problem: Iii. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers", *The International Journal of Robotics Research*, vol. 2, pp. 46-75, 1983c.
- J. T. Schwartz and M. Sharir, "On the Piano Movers' Problem. V: The Case of a Rod Moving in Three-Dimensional Space Amidst Polygonal Obstacles", *Communications On Pure And Applied Mathematics*, vol. 37, pp. 815-848, 1984.

- M. Sharir and E. Ariel-Sheffi, "On the Piano Movers' Problem. Iv: Various Decomposable Two-Dimensional Motion Planning Problem", *Communications On Pure And Applied Mathematics*, vol. 37, pp. 479-493, 1984.
- T. Shibata, T. Abe, K. Tanie and M. Nose, "Motion Planning by Genetic Algorithm for a Redundant Manipulator Using a Model of Criteria of Skilled Operators", presented at IEEE International Conference on Robotics and Automation, Nagoya, Japan, pp.2476-2481, 1995.
- T. Shibata and T. Fukuda, "Coordinative Behavior by Genetic Algorithm and Fuzzy in Evolutionary Multi-Agent System", presented at IEEE International Conference on Robotics and Automation, Atlanta, Georgia, USA, pp.760-765, 1993.
- Z. Shiller and S. Dubowsky, "Global Time Optimal Motions of Robotic Manipulators in the Presence of Obstacles", presented at IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, pp.370-375, 1988.
- Z. Shiller and S. Dubowsky, "On Computing the Global Time-Optimal Motions of Robotic Manipulators in the Presence of Obstacles", *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 785-797, 1991.
- K. Shin and N. McKay, "A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators", *IEEE Transactions on Automatic Control*, vol. 31, pp. 491-500, 1986.
- K. G. Shin and N. McKay, "Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints", *IEEE Transactions on Automatic Control*, vol. 30, pp. 531-541, 1985.
- T. Siméon, J. Laumond and C. Nissoux, "Visibility-Based Probabilistic Roadmaps for Motion Planning", *Advanced Robotics*, vol. 14, pp. 477-493, 2000.
- N. H. Sleumer and N. Tschichold-Gürman, "Exact Cell Decomposition of Arrangements Used for Path Planning in Robotics", Institute of Theoretical Computer Science, ETH Zürich, Technical Reports 329, 1999.
- J.-J. E. Slotine and H. S. Yang, "Improving the Efficiency of Time-Optimal Path-Following Algorithms", *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 118-124, 1989.

- P. Spirakis and C. K. Yap, "Strong Np-Hardness of Moving Many Discs", *Information Processing Letters*, vol. 19, pp. 55-59, 1984.
- V. J. Steffen and S. F. P. Samarago, "Optimization Techniques for Off-Line Trajectories Planning of Robot Manipulators", presented at ICONE--Second International Conference on Non-linear Dynamics, Chaos, Control and their Applications in Engineering Sciences, São Pedro, S.P. Brazil, 1996.
- Y. Stepanenko and M. Vukobratovic, "Dynamics of Articulated Open-Chain Active Mechanisms", *Mathematical Biosciences*, vol. 28, pp. 137-170, 1976.
- K. Sugihara and J. Smith, "A Genetic Algorithm for 3-D Path Planning of a Mobile Robot", Department of Information and Computer Science, University of Hawaii at Manoa, Tech. Report, Sept. 1996.
- Z. Sun and J. H. Reif, "Adaptive and Compact Discretization for Weighted Region Optimal Path Finding", presented at 14th Symposium on Fundamentals of Computation Theory, Malmö Högskola, Sweden, 2003.
- S. E. Thompson and R. V. Patel, "Formulation of Joint Trajectories for Industrial Robots Using B-Splines", *IEEE Transactions on Industrial Electronics*, vol. IE-34, pp. 192-199, 1987.
- L. Tian and C. Collins, "Optimal Placement of a Two-Link Planar Manipulator Using a Genetic Algorithm", *Robotica*, vol. 23, pp. 169-176, 2005.
- R. Toogood, H. Hao and C. Wong, "Robot Path Planning Using Genetic Algorithms", presented at IEEE International Conference on Systems, Man and Cybernetics, Vancouver, BC, Canada, pp.489-494, 1995.
- P. Tournassoud, *Géométrie Et Intelligence Artificielle Pour Les Robots*, Hermes ed., 1988.
- K.-M. Tse and C.-H. Wang, "Evolutionary Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots", presented at IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA, USA, pp.3272-3276, 1998.
- S. Udupa, *Collision Detection and Avoidance in Computer Controlled Manipulators*. Department of Electrical Engineering, California Institute of Technology. 1977.

- P. Vadakkepat, K. C. Tan and W. Ming-Liang, "Evolutionary Artificial Potential Fields and Their Application in Real Time Robot Path Planning", presented at Congress of Evolutionary Computation, San Diego, CA, pp.256-63, 2000.
- F. Valero, V. Mata and A. Besa, "Trajectory Planning in Workspaces with Obstacles Taking into Account the Dynamic Robot Behaviour", *Mechanism and Machine Theory*, 2005.
- F. Valero, V. Mata and M. Ceccarelli, "Path Planning in Complex Environments for Industrial Robots with Additional Degrees of Freedom", *COURSES AND LECTURES- INTERNATIONAL CENTRE FOR MECHANICAL SCIENCES*, pp. 431-438, 2000.
- F. J. Valero, *Planificación De Trayectorias Libres De Obstáculos Para Un Manipulador Plano*. Departamento Ingeniería Mecánica y de Materiales, Universidad Politécnica de Valencia. Valencia, Spain, 1990.
- F. J. Valero, V. Mata and A. Besa, "Trajectory Planning in Workspaces with Obstacles Taking into Account the Dynamic Robot Behaviour", *Mechanism and Machine Theory*, vol. 41, pp. 525-536, 2006.
- F. J. Valero, V. Mata and M. Ceccarelli, "Path Planning in Complex Environments for Industrial Robots with Additional Degrees of Freedom", presented at 13th CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators Ro.Man.Sy, pp.431-438, 2000.
- F. J. Valero, V. Mata, J. I. Cuadrado and M. Ceccarelli, "A Formulation for Path Planning of Manipulators in Complex Environments by Using Adjacent Configurations", *Advanced Robotics*, vol. 11, pp. 33-56, 1997.
- R. Volpe and P. Khosla, "Manipulator Control with Superquadric Artificial Potential Functions: Theory and Experiments", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, 1990.
- C.-H. Wang and J.-G. Horng, "Constrained Minimum-Time Path Planning for Robot Manipulators Via virtual Knots of the Cubic B-Spline Functions", *IEEE Transactions on Automatic Control*, vol. 35, pp. 573-577, 1990.
- E. Welzl, "Constructing the Visibility Graph for N-Line Segments in $O(N^2)$ Time", *Information Processing Letters*, vol. 20, pp. 167-171, 1985.

- S. A. Wilmarth, N. M. Amato and P. F. Stiller, "Maprm: A Probabilistic Roadmap Planner with Sampling on the Medial Axis of the Free Space", presented at IEEE International Conference on Robotics and Automation, Detroit, Michigan, USA, pp.1024-1031, 1999.
- X. J. Wu, J. Tang and K. H. Heng, "On the Construction of Discretized Configuration Space of Manipulators", *Robotica*, vol. 25, pp. 1-11, 2007.
- Y. Wu, *An Obstacle-Based Probabilistic Roadmap Method for Path Planning*. Department of Computer Science, Texas A&M University. 1996.
- T. Yoshikawa, *Foundations of Robotics Analysis and Control*. MIT Press, 1990.
- W.-M. Yun and Y.-G. Xi, "Optimum Motion Planning in Joint Space for Robots Using Genetic Algorithms", *Robotics and Autonomous Systems*, vol. 18, pp. 373-393, 1996.
- A. Zelinsky, *Environment Exploration and Path Planning Algorithms for a Mobile Robot Using Sonar*. Department of Computer Science, University of Wollongong. 1991.
- A. Zelinsky and S. Yuta, "A Unified Approach to Planning, Sensing and Navigation for Mobile Robots", presented at The 3rd International Symposium on Experimental Robotics, Kyoto, Japan, pp.444-455, 1993.
- C. S. Zhao, M. Farooq and M. M. Bayoumi, "Analytical Solution for Configuration Space Obstacle Computation and Representation", presented at IEEE IECON 21st International Conference on Industrial Electronics, Control, and Instrumentation, Orlando, FL, USA, pp.1278-1283, 1995.
- M. Zhao, N. Ansari and S. H. Hou, "Mobile Manipulator Path Planning by a Genetic Algorithm", presented at IEEE/RSJ International Conference on Intelligent Robots and Systems, Raleigh, NC, pp.681-688, 1992.
- D. Zhu and J. C. Latombe, "Constraint Reformulation in a Hierarchical Path Planner", presented at IEEE International Conference on Robotics and Automation, Cincinnati, OH, pp.1918-1923, 1990.

APPENDIX A

PUMA 560 ROBOTIC MANIPULATOR

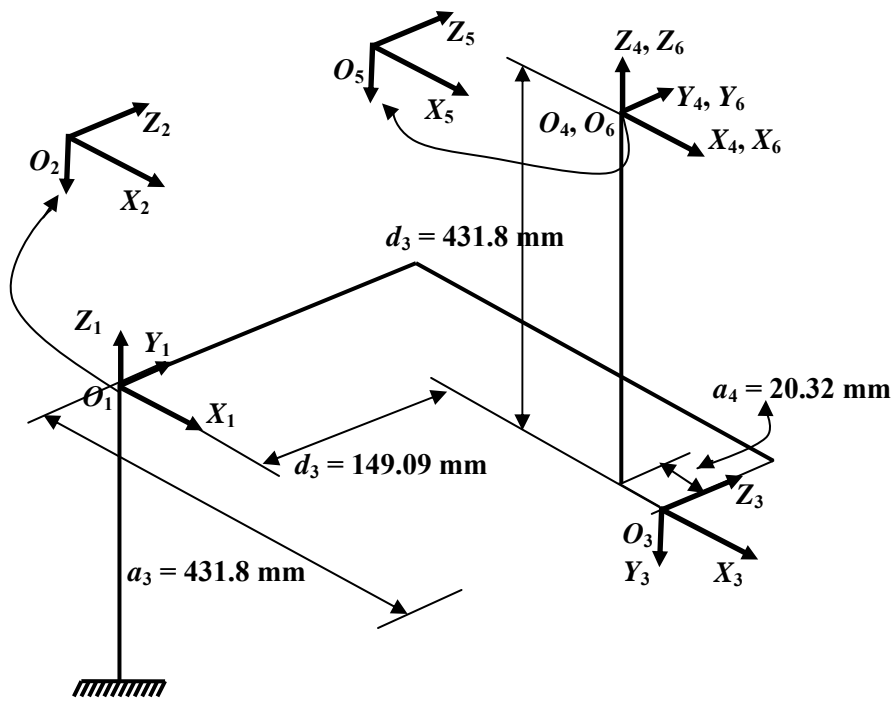


Figure A.1: Skeleton of the Puma 560 Robot with Local Coordinate Frames and Modified-DH Parameters (Out of scale).

The Puma 560 is a six degree of freedom robot manipulator. The end-effector of the robot arm can reach a point within its workspace from any direction. In this appendix all parameters and specifications used in this thesis for robot Puma 560 will be indicated.

A.1. DENAVIT-HARTENBERG PARAMETERS

A table that considers the common Modified Denavit-Hartenberg parameters a_i , α_i , d_i and θ_i for Puma 560 like robot, will be presented.

Link	α_i	a_i	d_i	θ_i
1	0	0	0	θ_1
2	-90	0	0	θ_2
3	0	431.8	149.09	θ_3
4	90	-20.32	431.8	θ_4
5	-90	0	0	θ_5
6	90	0	0	θ_6

**Table A.1: Modified-DH Parameters
Values for Robot Puma 560, Angles in
[degrees], Distances in [millimeter].**

A.2. DYNAMIC PARAMETERS

Robot links	Positions of the Centre of Gravity			Links Masses (kg)
	x	y	z	
1	0.0	- 0.054	0.0	10.521
2	0.1398	0.0	0.14909	15.781
3	- 0.00032	- 0.197	0.0	8.767
4	0.0	0.0	- 0.057	1.052
5	0.0	- 0.007	0.0	1.052
6	0.0	0.0	0.03725	0.351

Table A.2: Positions of the Centre of Mass and Masses for Puma 560 Links, Values in [meter].

Robot links	I_{xx}	I_{yy}	I_{zz}	I_{xy}	I_{xz}	I_{yz}
1	1.6120	0.5091	1.6120	0.0	0.0	0.0
2	0.4898	8.0783	8.2672	0.0	0.0	0.0
3	3.3768	0.3009	3.3768	0.0	0.0	0.0
4	0.1810	0.1810	0.1273	0.0	0.0	0.0
5	0.0735	0.0735	0.1273	0.0	0.0	0.0
6	0.0071	0.0071	0.0141	0.0	0.0	0.0

Table A.3: Inertia Tensor for Puma 560 Links, Values in [kg/m^2].

A.3. LOCAL POSITIONS OF SIGNIFICANT AND INTERESTING POINTS

Significant Points are γ_1 , γ_2 , γ_3 , and γ_4 . Interesting Points are λ_1 , λ_2 , λ_3 , and λ_4

Significant Points		
	Reference frame	Local Position
γ_1	2	0.0, 0.0, 0.255
γ_2	3	0.0, 0.0, 0.105
γ_3	3	0.0, -0.351, 0.0
γ_4	6	0.0, 0.0, 0.267
Interesting Points		
	Reference frame	Local Position
λ_1	2	-0.229, 0.0, 0.255
λ_2	3	0.0, 0.1, 0.0
λ_3	4	0.0, 0.0, -0.081
λ_4	5	0.0, 0.0, 0.0

Table A.4: Significant and Interesting Points used for Puma 560 in This Thesis, Values in [meter].

A.4. JOINTS LIMITS

Joint	Minimum Value	Maximum Value
1	-160	160
2	-215	35
3	-45	225
4	-140	140
5	-100	100
6	-266	266

Table A.5: Admissible Movement Range for Each Joint, Values in [degree].

LIST OF FIGURES

Figure 1.1: Visibility Graph Example, Latombe 1991.	18
Figure 1.2: Voronoi Diagram Example, Latombe 1991.	19
Figure 1.3: Exact Cell Decomposition, Latombe 1991.	21
Figure 2.1: Robot Wired Model.	48
Figure 2.2: Modified Denavit-Hartenberg Assignment Criteria for Link with Revolute Joint.	51
Figure 2.3: Some Kinematic Parameters and Frame Assignments for the Puma 560 Manipulator.	53
Figure 2.5: Minimum Distance Derivation Between Robot Link and Sphere.	69
Figure 2.6: The Flow-Chart of the Algorithm to find the Shortest Distance Between Robot Link and Sphere.	70
Figure 2.8: Minimum Distance WHEN $x_1 \neq x_{11}$ or $x_2 \neq x_{22}$.	77
Figure 2.9: The Flow-Chart to find the Shortest Distance Between Robot Link and Cylisphere.	78
Figure 2.10: Minimum Distance Derivation Between Robot Link and a Quadri-Lateral Plane.	79
Figure 2.11: Minimum Distance When Line Endpoints Outside Quadri-Lateral.	83
Figure 2.12: The Algorithm to Find the Distance Between Robot Link and Quadri-Lateral.	84
Figure 3.1: Adjacent Configuration for Path Planning GA Chromosome.	92
Figure 3.2: Flow-Chart Indicates How to calculate the q_i Intervals.	93
Figure 3.3: Adjacent Configuration Crossover and Mutation.	94
Figure 3.4: Path Planning GA Chromosome.	96
Figure 3.5: Crossover Between Two Robot Paths.	99
Figure 3.6: Path GA Mutation.	101
Figure 3.7: Example 1 – The Case of Three Spherical Obstacles.	113
Figure 3.8: Example 1– The Case of Three Cylispherical Obstacles.	113

Figure 3.9: Example 1– The Case of Three Quadri-Lateral Plane Obstacles.	113
Graph 3.1: Example 1 – A Comparison Results of d_s (m) Between GA Procedure and Rubio Algorithms.	116
Graph 3.2: Traveled Distance Comparison Between GA Procedure and Rubio Procedures.	117
Graph 3.3: Travelled Distance Comparison Between GA Procedure & Seq Procedure Produced by Rubio et al. 2009a.	117
Graph 3.4: Travelled Distance Comparison Between GA Procedure & A* Procedure Produced by Rubio et al. 2009a.	118
Graph 3.5: Travelled Distance Comparison Between GA Procedure & UC Procedure Produced by Rubio et al. 2009a.	118
Graph 3.6: Travelled Distance Between GA Procedure & G Procedure Produced by Rubio et al. 2009a.	119
Graph 3.7: Computational Time Comparison Between GA and Rubio Procedures.	119
Graph 3.8: Objective Function (Time in seconds) vs. No. of Generations, Example 1.	123
Graph 3.9: Joint Variables and Derivatives vs. Time.	125
Graph 3.10: Joints (Coordinates, Velocities, Accelerations, Torques, and Power) vs. Execution Time.	126
Figure 4.1: Workspace Generation.	137
Figure4.2: Adjacent Configuration GA Chromosome.	140
Figure 4.3: Trajectory GA Chromosome.	145
Graph 4.1: Execution Time Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Harmonic Interpolation Functions.	149
Graph 4.2: Travelled Distance Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Harmonic Interpolation Functions.	149
Graph 4.3: Computational Time Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Harmonic Interpolation Functions.	150
Graph 4.4: Execution Time Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Cubic Polynomial Interpolation Functions.	151

Graph 4.5: Execution Time Comparison Between GA Procedure & A* Procedure Produced by Rubio 2006.	151
Graph 4.6: Execution Time Comparison Between GA Procedure & Amplitude Procedure Produced by Rubio 2006.	152
Graph 4.7: Execution Time Comparison Between GA Procedure & Voraz Procedure Produced by Rubio 2006.	152
Graph 4.8: Travelled Distance Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Cubic Polynomial Interpolation Functions.	153
Graph 4.9: Travelled Distance Comparison Between GA Procedure & A* Procedure Produced by Rubio 2006.	153
Graph 4.10: Travelled Distance Comparison Between GA Procedure & Amplitud Procedure Produced by Rubio 2006.	154
Graph 4.11: Travelled Distance Between GA Procedure & Voraz Procedure Produced by Rubio 2006.	154
Graph 4.12: Computational Time Comparison Between GA Procedure Using Cubic Polynomial for Interpolation and Rubio Procedure Using Cubic Polynomial Interpolation Functions.	155
Figure 4.4: Example 3.	157
Figure 4.5: Example 4 (Case With and Without Obstacles).	159
Figure 4.6: Example 4.	160
Figure A.1: Skeleton of the Puma 560 Robot with Local Coordinate Frames and Modified-DH Parameters (Out of scale).	189

LIST OF TABLES

Table 2.1: The Obstacle Three Basic Elements.	67
Table 3.1: Initial and Final Configurations for Example 1.	111
Table 3.2: Obstacles Locations (in m) for Example 1.	112
Table 3.3: Example 1 Results.	116
Table 3.7: Sequence of Configurations [degree].	122
Table 3.8: Velocity, Acceleration, and Jerk Constraints.	123
Table 3.9: Parameter Values for the Genetic Algorithm Procedure.	123
Table 3.10: Best Minimum Time with Crossover Rate = 0.95, Example 1.	124
Table 4.1: Comparison Results Between GA & SQP.	142
Table 4.2: Example 1 Results.	148
Table 4.3: Comparison Between Direct and Indirect Method.	156
Table 4.4: Example 3 Results.	157
Table 4.5: Initial and Final Configurations for Example 4.	158
Table 4.6: Obstacles Locations (in m) for Example 4.	158
Table 4.7: Example 4 Results.	159
Table 4.8: Example 4 Results.	161
Table A.1: Modified-DH Parameters Values for Robot Puma 560, Angles in [degrees], Distances in [millimeter].	190
Table A.2: Positions of the Centre of Mass and Masses for Puma 560 Links, Values in [meter].	191
Table A.3: Inertia Tensor for Puma 560 Links, Values in [kg/m^2].	191
Table A.4: Significant and Interesting Points used for Puma 560 in This Thesis, Values in [meter].	192
Table A.5: Admissible Movement Range for Each Joint, Values in [degree].	193