ROBOT PROGRAMMING AND TRAJECTORY PLANNING USING AUGMENTED REALITY

FANG HONGCHAO

(B. Eng., M. Eng.)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2012

Acknowledgements

First and foremost, I would like to express my utmost gratitude to my supervisors, Prof. Andrew Nee Yeh Ching and Assoc Prof. Ong Soh Khim, for giving me the opportunity to pursue this degree. Without their guidance and encouragement, this research would not have been possible. Their patience and suggestions encouraged me to learn and progress. I can still remember the conversations and discussions we have had especially when my research seemed off the track.

I greatly appreciate the help from the members in the ARAT Lab, a source of friendships, good advice and collaboration. I would like to thank Dr. Yuan Miaolong, Dr. Louis Fong, Dr. Shen Yan, Dr. Zhang Jie, Wang Zhenbiao, and many others, for they have contributed immensely to my personal and professional time.

I like to show my sincere appreciation to my family, my parents and my sister, for all their love, understanding and support. Special thanks to Zhao Ben and Chen Jian, warm-hearted friends who have been my roommates in Singapore, for their help and accompaniment.

Last but not least, I would like to thank the National University of Singapore and the Department of Mechanical Engineering for providing me the opportunity to conduct this research.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
ACL	Advanced Control Language
AI	Artificial Intelligence
AR	Augmented Reality
ARVIKA	Augmented Reality for Development, Production and Servicing
ATS	Advanced Terminal Software
AVILUS	Federal Augmented Reality Research Project
BRIEF	Binary Robust Independent Elementary Features
C-space	Configuration Space
CAD	Computer-Aided Design
CAVE	Cave Automatic Virtual Environment
CFV	Collision Free Volume
CG	Centre of Gravity
CNC	Computer Numerical Control
CV	Computer Vision
DD	Doppler Distance
D-H	Denavit-Hartenberg convention
DOF	Degrees-of-Freedom
EE	End-Effector
ESM	Efficient Second-order Minimization

FOV	Field of View
FRP	Functional Reactive Programming
GHz	GigaHertz
GPS	Global Positioning System
HCI	Human-Computer Interface
HMD	Head Mounted Display
HRI	Human-Robot Interaction
IFR	International Federation of Robotics
IMU	Inertial Measurement Unit
IR	Infra-Red
Kg	Kilogram
Kgm^2	Kilogram Square Meter
LOA	Level of Autonomy
MEMS	Micro-Electro-Mechanical Systems
mm	millimetre, or one thousandth of a metre
MORPHA	Anthropomorphic Robot Assistants Project
PbD	Programming by Demonstration
PDA	Personal Digital Assistant
PTAM	Parallel Tracking and Mapping
PTAMM	Parallel Tracking and Multiple Mapping
RLW	Remote Laser Welding
RNE	Recursive Newton-Euler
RT	Random Tree
SIFT	Scale Invariant Feature Transform

SLAM	Simultaneous Location and Mapping
SMEs	Small and Medium Enterprises
SURF	Speeded Up Robust Features
TOF	Time-of-Flight
UI	User Interface
VE	Virtual Environment
VR	Virtual Reality

List of Symbols

Chapter 3

x, y 2D image coordinates

X, Y, Z 3D Euclidean space coordinates

${}^{(1)}_{(2)}T$ Transformation matrix relating coordinate system (2) with respect to coordinate system (1)

q Vector of generalised coordinates for a robot

θ Vector of angular joints for a robot

Chapter 4

$\Re^3$ 3D Euclidean space

S_i Virtual spheres

c_i Coordinates for the centre of S_i

r_0 Radius of virtual spheres

N_0 Number of virtual spheres recorded during human demonstrations

r, h, o, z Parameters of bounding cylinder

V Control points used to form a robot path

N_p Number of control points

d_i Euclidean distance between *i-1*th and *i*th control points

t_j Time stamp attached to the *j*th control point

ndof	Degree-of-freedom of the robot
q	Joint coordinate vector
$\dot{\mathrm{q}}$	Joint velocity vector
$\ddot{\mathrm{q}}$	Joint acceleration vector
$\bar{\dot{\mathrm{q}}}, \underline{\dot{\mathrm{q}}}$	Upper and lower bounds for joint velocity vector
s	Scalar path coordinate
Δs	Difference of two consecutive path coordinate
$\dot{s}$	Time-derivative of the path coordinate
$\ddot{s}$	Second time-derivative of the path coordinate
N_s	Number of path segments for trajectory optimization
$M(\mathrm{q})$	Mass matrix
$C(\mathrm{q},\mathrm{q}')$	Coriolis and centrifugal matrix
$G(\mathrm{q})$	Gravity and joint angle dependent torque vector
F_s	Friction vector
$\bar{\tau}, \underline{\tau}$	Upper and lower bounds for joint torque vector
τ	Joint torque vector
E_T	Optimal value of objective function (path duration)
κ	Log-barrier parameter
$E_{T-\Gamma}$	Optimal value of log-barrier objective function
τ_u	Unified joint torque vector

Chapter 5

u, v Ruled surface parameters

$\mathbf{x}(u,v)$ Ruled surface

$\mathbf{b}(u)$ Directrix vector of a ruled surface

$\delta(v)$ Rulings vector if a ruled surface

υ Length of the rulings

Π_0 Plane that consists of all the possible orientations of the EE for a TCP

R_x, R_y, R_z Rotational vectors of the EE with respect to X-, Y-, and Z- axis (Obtained via interaction device)

mR_x, mR_y, mR_z Rotational vectors (Valid) of the EE with respect to X-, Y-, and Z- axis after mapping and projection

θ Vector of angular joints for a robot

$\Re^3$ 3D Euclidean space

t^L Vector of default index of *parameterized* points on curve model

a^L Vector of auxiliary index of *parameterized* points on curve model

p^L Vector of spatial positions of *parameterized* points on curve model (with respect to the robot base frame)

N_l Number of *parameterized* points on curve model

t^R Vector of default index of *re-parameterized* points on curve model

a^R Vector of auxiliary index of *re-parameterized* points on curve model

p^R Vector of spatial positions of *re-parameterized* points on curve model (with respect to the robot base frame)

N_s	Number of *re-parameterized* points on curve model
O_0	Origin of the coordinate frame defined on the interaction device
$S(\cdot,\cdot)$	Euclidean distance between the pair of points given in the bracket
CP	Control points
k	Index of control point
a	Auxiliary index of control point
p	Positional component of the coordinate frame defined on control point
R	Rotational component of the coordinate frame defined on control point
α	EE inclination range
o	Vector of EE orientation
β	Angle between the EE and the Z-axis of the robot base frame
p_{poi}	A point of interest in control point selection
N_k	Number of control points
τ_{sim}	Vector of simulated torque
$\overline{\tau}$	Upper bound for joint torque vector
$\dot{q}_{sim}$	Vector of simulated velocity
$\overline{\dot{q}}$	Upper bound for joint velocity vector
m_u	Vector of normalized measurement associated with the simulated torque and simulated velocity

Appendix

Ω	Minimization of the proposed cost function
b	Optimization variable

$P^{\tau}(\tau)$ Log-barrier item associated with joint torque

$P^{v}(\dot{q})$ Log-barrier item associated with joint velocity

$q(s)$ Joint angle represented with respect to path coordinate ***s***

$q'(s)$ First-order derivative of joint angle with respect to path coordinate ***s***

$q''(s)$ Second-order derivative of joint angle with respect to path coordinate ***s***

N_t Total number of path segments

Δs^i Discrete path segment, here it is equally spaced, $i = 0,1,2,...,N_s - 1$

κ^{τ} Log-barrier parameter for joint torque constraint, $\kappa^{\tau} \geq 0$

κ^{v} Log-barrier parameter for joint velocity constraint, $\kappa^{v} \geq 0$

$\tau(b^i, b^{i+1})$ Joint torque vector evaluated at midpoint of each path segment

$\dot{q}(b^i, b^{i+1})$ Joint velocity vector evaluated at midpoint of each path segment

$\overline{\tau}$, $\underline{\tau}$ Upper and lower bound for joint torque vector

W_{tau} Diagonal matrix associated with joint torque constraints

$\overline{\dot{q}}$, $\underline{\dot{q}}$ Upper and lower bounds for joint velocity vector

b^0 Initial state of the optimization variable

$\mathbf{J}(\mathbf{b})$ Jacobian of the cost function with respect to the optimization variable

$\mathbf{H}(\mathbf{b})$ Hessian Matrix of the cost function with respect to the optimization variable

$\mathbf{z}_{(j)}$ Intermediate variables in solving LU decomposition (with respect to $b_{(j)}$)

$\Delta\mathbf{b}_{(j)}$	Search direction of the optimization variable (with respect to $\mathrm{b}_{(j)}$)
$L_{k,l}$	The *kl*th entry of a lower diagonal matrix
$U_{k,l}$	The *kl*th entry of an upper diagonal matrix
ξ	Tolerance for the convergence of numerical solution (threshold for the Norm of the Jacobian matrix of the optimziation problem)
δ	Step size in backtracking line search
η	Parameter for adjusting step size δ, $\eta \in (0,1)$
γ	Parameter for adjusting the convergence of merit function during backtracking line search
b_{opt}	Solution to the trajectory optimization problem
tau_{opt}	Optimized joint torque vector
$\dot{\mathrm{q}}_{opt}$	Optimized joint velocity vector
$\ddot{\mathrm{q}}_{opt}$	Optimized joint acceleration vector
t_{opt}	Optimized path duration
E_{opt}	Optimized energy consumption

Abstract

The employment of multi-purpose industrial robots has raised challenges on the identification of intuitive interfaces for effective Human Robot Interaction (HRI) that are suitable for the users, who may not necessarily be experienced in the field of robotics, robot installation and re-programming. In this study, an application of Augmented Reality (RPAR-II) is proposed and validated, for intuitive path planning, end-effector (EE) orientation planning, and the transformation of the planned trajectories into task-optimized executable robot paths.

The proposed approach for path planning focuses on point-to-point robot operations, and aims to find a suitable path between a pair of start and goal configurations. After generating a collision-free volume (CFV), the users can create a series of control points interactively within the CFV to form a path. A Euclidean-based method has been developed for the selection of control points that are of interest and control point modification. A time-optimal trajectory optimization method, incorporating robot dynamics, is applied to the planned path to obtain the approximated optimal trajectory profile rapidly, which is used to simulate the virtual robot with a suitable control scheme.

The proposed approach for robot end-effector (EE) orientation planning focuses on continuous path following operations, which constrain the EE to follow a visible path on a workpiece at permissible inclination angles. The proposed

approach allows the users to select a sequence of control points on the curve, and specify the orientation of the EE at each point respectively. A modification scheme has been developed to facilitate the orientation planning, such that the resulting ruled surface representing the orientation profile of the EE along the curve can be adjusted interactively and efficiently. Applying the aforementioned trajectory optimization method, an optimal trajectory can be determined suitably for the direct transformation into executable robot controller codes.

Chapter 1. Introduction

1.1 Introduction

This chapter begins with a discussion of the current state of robotics and the issues identified by the researchers to achieve intuitive and user-friendly robot programming interface. One of the challenging issues has been the intuitive and efficient human-robot interaction (HRI) that is essential for the prevalence of robots supporting humans in key areas of activities, which will in turn affect the ways the robots are being programmed and robot tasks planned. The motivation, the objectives, and the scope of the research are identified and defined. Lastly, the organization of the thesis is presented.

1.2 Recent Advancements and Trends in Robotics

Robots are designed and employed primarily for carrying out programmed, repetitious tasks to promote productivity and efficiency, as well as replacing human workers for performing dangerous, dirty and dreary tasks. Recent developments and advancements in robotics have seen the emergence of various types of robots for a wide range of pervasive applications spanning from manufacturing operations to servicing activities outside the factory floors. Meanwhile, the increasing demand for higher quality of life has led to the

development of robots that are capable of performing tasks associated with everyday activities, e.g., for domestic services, or entertainment purposes, etc.

According to the International Federation of Robotics (IFR), robots can be classified into two categories, namely, industrial robots and service robots. The industrial robots have been widely used in various manufacturing processes where the tasks are often executed in prepared environments. Therefore, these robots, often with little autonomous capability, need to be re-programmed for a new task, in which the robots may need a different tool, fixture or environment (Meeussen 2006). Service robots are usually operated semi- or fully autonomously for the well-being of humans or equipment (IFR world robot 2011). For instance, professional robots are employed to reduce physical workloads and intervene in hazardous environments; robot companions are designed to improve personal well-being, security, as well as provide entertainment.

An IFR report on robotics reveals that there is a strong recovery for the sales of industrial robots in 2010, and the installations of industrial robots will continue to increase in the next few years. The main drivers are the automotive and the electronics industries, where new manufacturing lines, e.g., for electric vehicles, solar cells, etc., will be required to implement new technologies and use new materials. It is forecast that there will be a high potential for the installation of multi-purpose robots in the general industry, particularly in growing industries, such as the rubber and plastics, metal products, medical devices, food and beverage, etc. (IFR world robot 2011). The demand for service robots has risen

significantly, led by countries, such as Japan where robots have provided services and become home companions of the lonely and the elderly.. For service robots, it is predicted that for the period from 2011 to 2014, there will be 87,500 new service robots for professional use to be installed, such as defense robots, field robots, medical robots, etc., and about 14.4 million units of service robots for personal use to be sold, such as domestic robots, entertainment and leisure robots, etc. It is expected that the market for robots for providing handicap assistance, personal transportation, and home security and surveillance will increase substantially in the near future (IFR service robot 2011).

In summary, the expected growth in demand is largest for service robots, followed by industrial robots. For the latter, the working scenarios have been changing from mass production lines towards batch production work cells in small and medium enterprises (SMEs). This has eventually encouraged the collaboration among research institutions and industries towards multi-purpose robotic systems, as well as intuitive and effective robot programming interfaces suitable for the users who may not be experienced in the field of robotics (Thrun 2004; SMErobot 2009; IFR service robot 2011; IFR world robot 2011).

1.3 Robot Programming and HRI

HRI in industrial robotics has largely been confined to finding better ways to reconfigure or program the robots. Generally, HRI is referred to a process that interprets the task descriptions into a sequence of robot motions complying with

robot capabilities and the working requirements. In particular, it involves path planning, set-points extraction and robot programs generation. The path planning task is mainly concerned with the accessibility of the end-effector (EE), collision avoidance, etc.; the set-points are generated as a representation of the planned trajectory incorporating robot dynamics, such that they can be transferred directly to executable robot programs. This process can be carried out either manually or automatically according to the level of autonomy (LOA) (Parasuraman *et al*. 2000; Goodrich and Schultz 2007) achievable by the robotic system, which describes to what degree the robot can act on its own accord, or alternatively, to what degree the HRI is involved in completing a robot task.

Robot programming has evolved from simple robot guiding methods to advanced graphical offline programming approaches. In traditional methods, the robots are programmed either through manipulating the robot arms remotely using a teaching pendant, or guiding the EE of the robot directly by a user who has to be present within the workspace of the robot. Recently, more advanced interaction schemes, facilitated by sophisticated computer-aided design (CAD) models of both the robot and the entities in the working environment, have been developed (Neto *et al*. 2010a; 2010b; Chen and Sheng 2011). Using virtual reality (VR), a virtual robot is programmed in a totally immersive virtual environment (VE). Recent research on HRI issues suggests that augmented reality (AR) interfaces can enrich the interaction process in robot manipulation (Marin *et al*. 2005; Chintamani *et al*. 2010). The use of AR in robotic systems retains the advantage of the VR-based systems. In addition, the entire working environment does not need to be replicated,

and the users can perceive and interact with the geometric information associated with the planned paths instantaneously (Zaeh and Vogl 2006; Reinhart *et al*. 2008; Chong *et al*. 2009; Ong *et al*. 2010).

Robot safety ranks among the top priority issues and needs to be addressed in robot programming. Currently, most of the industrial robotic systems adopt semi-automatic programming approaches. First, completely manual programming is unintuitive and time-consuming, or requires an operator to be present within the workspace of the robot, which may pose safety concerns, e.g., physical HRI (pHRI) (Bicchi *et al*. 2007). Secondly, completely automatic programming requires complicated hardware and infrastructure, as well as sophisticated software development for environment sensing, motion control, exceptions processing, robust error handling and recovery, etc., during the programming process. Meanwhile, anecdotal evidence has suggested that some robot failures could have been prevented if the robot either has the ability to enter an appropriate autonomous state or the operator has commanded the robot to do so (Steinfeld *et al*. 2001). In addition, Breazeal *et al*. (2001) identified that one of the key requirements for effective HRI is the overlapping space that can be perceived by both the human user and the robot programming system. It would be desirable to have economically feasible solutions where the operator can work as an assistant with the complex robots to solve unpredictable problems (Brogårdh 2007). In summary, the HRI remains a vital component in the programming process, involving the understanding and shaping the interactions between the operators and the robots through the integration of various types of sensors.

1.4 RPAR-I System

Chong *et al.* (2007) proposed an approach for immersive robot programming, namely, RPAR-I, where the user moves a virtual robot directly among real objects in an unknown environment. This approach enables the users to perform robot programming for two classes of tasks, namely, Class I tasks such as pick-and-place tasks where there are a number of possible path solutions for a given start and goal configuration (Chong *et al*. 2009); and Class II tasks such as arc welding, etc., where the EE of the robot is constrained to follow a user-defined three-dimensional (3D) path at a certain orientation, consistently with respect to the path (Ong *et al*. 2010).

There are a number of potential research directions to further develop RPAR-I. In RPAR-I, the planned paths are verified with simulation using a virtual robot where the speed of the EE is kept constant. The paths cannot be transferred directly into executable robot controller codes as the capabilities, or limits of a particular robotic system, such as the constraints on joint velocity, joint acceleration, joint torque, etc., have not been considered. The appropriate and efficient orientation planning of the EE for path following tasks, such as arc welding, gluing, etc., in a complex working environment, such as obstacle-intensive working station, etc., has not been provided.

1.5 Research Motivations and Objectives

There is a progressive increase in the demand for robot solutions that are less expensive, safer, easier to install and reprogram, etc., and are adaptable by the SMEs. It would be desirable to have natural and intuitive HRI that enables average users, who might not have sophisticated robot programming skills as experts do, to work as an assistant with the robots in an unknown environment, particularly in professional and domestic domains. To this end, the HRI would need to utilize the human operators' common cognition/expertise, and address the issues regarding the conventional programming methods for industrial robots.

AR has been applied in a wide range of applications in manufacturing (Ong *et al*. 2008). The use of AR in robotic systems has been proven useful, by which users are able to interact with the spatial environment through highly intuitive interaction interfaces, such as a teaching stylus for manual guidance, and perceive instantaneous feedback through the integration of various types of sensors, such as force and torque sensors (SMErobot 2009), or optical tracking devices (Zaeh and Vogl 2006; Reinhart *et al*. 2008; Chong *et al*. 2009; Ong *et al*. 2010). The interaction with a virtual robot model, instead of the real robot, makes the operator safer even when he/she is present within the operating range of the robot. Numerous research efforts have been reported on AR for robots, and there are larger research initiatives, such as AVILUS (AVILUS; Lieberknecht *et al*. 2009) and SMErobot (SMErobot 2009; Hollmann *et al*. 2010). Many manufacturing

companies, e.g., Siemens, Audi, BMW, etc., have also participated in various AR projects for product development, production and service (Friedrich 2002).

The aim of the research is to develop an AR-based system capable of performing robot programming and task planning in efficient and intuitive manners, namely, Robot Programming and Trajectory Planning using Augmented Reality (RPAR-II). The research investigates the potential of AR in robot applications and the ways AR can add values in solving classic problems in robotics, particularly in path planning, robot EE orientation planning, as well as path optimization incorporating robot dynamics capabilities, such that the planned paths can be translated directly into robot controller codes. The objective of this research can be summarized as follows:

(1) Development of an AR-assisted HRI method for intuitive robot task planning. The intervention of the users in robot path planning process allows the path to be modified when the simulation is not satisfactory, which in turn increases the efficiency of the proposed method.

(2) Development of a novel AR-based approach for robot path planning given a pair of start and goal configurations in an unprepared environment. Examples of the target applications are material handling, robot spot and seam welding, etc. Decoupled motion planning is adopted, in which the users can create a number of control points interactively in the AR environment to form a collision-free geometric path, and it is then optimized incorporating robot dynamics constraints.

(3) Development of a new AR-based approach for intuitive and robot EE orientation planning in applications where the robot EE is constrained to follow a visible path, such as robot arc welding, laser welding, gluing, etc. The planning method should be interactive, flexible and efficient, allowing the users to modify the EE orientation iteratively at any point (or segment) of interest on a path, such as a region close to an obstacle, etc, until a suitable EE orientation profile is finally determined.

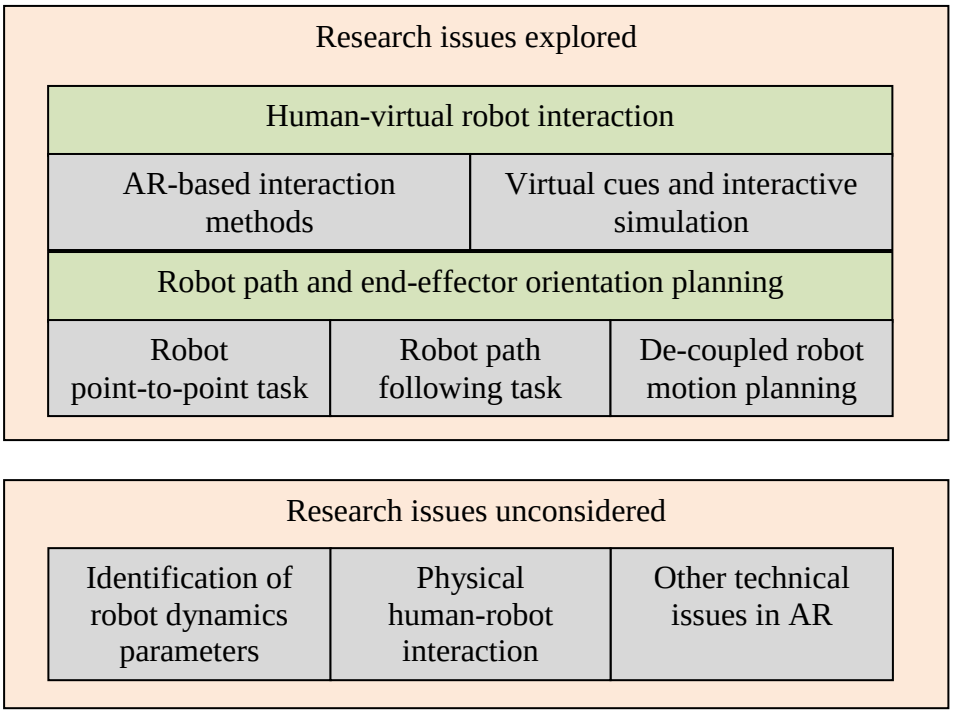


Figure 1.1: Research scope

The research scope, as shown in Figure 1.1, covers the following aspects:

(1) Human-virtual robot interactions, which include the realization of parametric robot modeling and virtual robot registration and tracking, a set

of AR-based interaction methods, and the use of visual cues to facilitate interactive simulation.

(2) Two types of robot operations, namely, pick-and-place tasks where a pair of start and goal configurations are given, and continuous path following operations, are considered in the proposed system. The users are able to interact with the virtual robot to intervene in the geometric path planning and robot EE orientation planning. An intuitive interface for human-virtual robot interaction will be established.

(3) Investigation of the de-coupled approach for robot motion planning in an AR environment. The motion planning problem can be divided into two sub-areas, namely, geometric path planning and trajectory optimization. A collision-free path can be formed through a set of AR-based HRI methods; a trajectory can be optimized subject to robot kinematics and dynamics constraints, as well as tasks requirements. The trajectories can be translated directly into executable robot programs.

There are a few relevant issues/topics that have not been covered in this research, e.g., the identification of robot dynamic parameters, physical interaction with real robot during robot task planning. Two essential activities in AR, namely, real-time tracking and registration, have been realized. However, other technical issues of AR, such as illumination, occlusion, etc., have not been addressed.

1.6 Organization of Thesis

This thesis is organized as follows:

In Chapter 2, current research in robot programming, robot motion planning, AR and AR applications in robotics is surveyed and discussed. Particularly, the relevant works within the research objectives and scope are discussed and investigated in detail, namely, AR-based robotic systems, the use of AR in robot path planning, and orientation planning of the EE.

In Chapter 3, a detailed architecture of the proposed RPAR-II system is described. An AR environment, including a hand-held device, is developed to facilitate HRI. A robot parametric model, registration of the robot model in the AR environment, and the relationships between the main coordinate systems defined in the system, are presented.

In Chapter 4, an approach for path planning and simulation using AR is presented. The target applications are those where a collision-free path among existing objects in the working environment need to be found given a pair of start and goal configurations, e.g., robot spot welding, material handling, etc. The proposed approach allows the users to create a series of control points interactively within the collision-free volume (CFV), which is the sub-space of the workspace, and form a path through interpolation using the control points created. A Euclidean distance-based method is developed to facilitate control point modification. An

optimal time-scale trajectory is obtained by solving a convex optimization problem subject to robot kinematics and dynamics constraints. The trajectory is used to simulate the virtual robot performing the trajectory in a real environment to allow the user to evaluate the quality of the resulting trajectory visually.

In Chapter 5, an approach is proposed for orientation planning of the robot EE using AR. The target applications are those where the EE of the robot is constrained to follow a 3D curve at permissible inclination angles with respect to the curve. A HRI scheme is developed through which the users can select a sequence of control points on the output curve, and specify the orientation of the EE at each control point with respect to a universal coordinate frame. A smooth path of the EE can be generated through orientation interpolation associated with the output curve. The trajectory is optimized using similar methods as presented in Chapter 4, subject to joint velocity constraints and the joint torque constraints. The successfully simulated trajectory is finally compiled into controller codes for further verification on the real robot.

In Chapter 6, the implementation and the Graphical-user interface (GUI) of the proposed system are presented, and the software modules and the programming environment are discussed. Three case studies are presented. The first case study involves finding a collision-free trajectory among existing objects for a pick-and-place task. The remaining two case studies are orientation planning of the robot EE to follow an *S*-shaped curve and a circular curve respectively. User

studies have been conducted to evaluate the proposed system and methodologies quantitatively.

Finally, Chapter 7 summarizes the thesis by presenting the key contributions of the research and future research opportunities.

Chapter 2. Literature Survey

2.1 Introduction

This chapter presents a survey of the literature pertinent to the studies on robot programming and trajectory planning, and the applications of virtual and augmented realities in robotics. This chapter begins with a review of recent progress on robot programming methods, followed by existing robot motion planning methods. A brief survey on Augmented Reality (AR) and its applications, including the key issues of AR-based systems, various applications of AR in manufacturing domain, particularly in robotics, are given.

2.2 Robot Programming

Robot programming refers to the practice of creating and fine tuning robot operations, involving a variety of tasks, such as robotic system setup and evaluation, cycle time optimization, safety and throughput improvement. Robot programming is an essential component in a successful execution of a robotic system. A review on industrial robot programming systems was first presented by Lozano-Perez (1983a) and during that period, most of the robots were employed in industrial environments, and programmed through manual guiding methods, or using robot controller-specific programming languages. Biggs and MacDonald (2001) reviewed robot programming methods developed more recently for both

industrial robots and service robots. Generally, there are three ways that an industrial robot can be programmed, namely, lead-through programming, walk-through programming, and off-line programming (OTM 1999). According to the degree of automation achievable, methods for robot programming can be categorized into manual programming, semi-automatic programming and automatic programming, as listed in Table 2-1.

Table 2-1**:** Categories of robot programming methods.

	Degree of Automation		
	Manual programming	Semi-automatic programming	Automatic programming
On-line Programming	- Walk-through - Lead-through (using teach pendant)		- PbD (Programming by Demonstration) - Multi-modal systems (voice-based, gesture-based, etc.) - …
Off-line Programming	- Text-based (textual interface) - Icon-based (graphical interface)	- Virtual Reality-based - Sensor-based - Augmented Reality-based - …	

2.2.1 On-line Robot Programming

2.2.1.1 On-line Manual Robot Programming

Lead-through and walk-through programming are two conventional methods for robot programming. In lead-through programming, a teaching pendant is used to drive the robot to the desired configurations, which are stored according to names used in robot programs. The movements of the robot are played back for verification. In walk-through programming, a user moves the end-effector (EE) of

the robot passing through a desired path with the drive of the robot powered off and the joint brakes applied. One of the disadvantages in these two methods is the safety issue as the users need to be present within the working environment of the robot. Walk-through programming is rather restrictive due to possible mechanical constraints in the working environment. In addition, the joint resistance of the actuators makes it difficult for a user to drag the robot following a given path. In general, conventional on-line programming methods are time-consuming and may pose safety concerns for both the human operators and the robot itself, e.g., physical injury to the human, collision among the robot arms and the obstacles in the working environment.

2.2.1.2 Robot Programming by Demonstration

Programming by Demonstration (PbD) is an on-line robot programming approach that has been studied extensively, and a comprehensive survey on this topic has been presented by Argall *et al*. (2009). The PbD approach involves a user performing the task manually. However, instead of a play-back of the recorded motion sequence, the PbD approach leaves the robot to observe, follow (replicate) and learn the human demonstrations in real-time. This enables a user who may not have any robot programming skills to reconfigure the robot in the development and maintenance of robot programs. The PbD approach has also been applied to program professional robots and humanoid robots, as these robots require learning new skills and adapting the existing skills to new contexts in continuously changing environments (Zollner *et al*. 2004; Calinon *et al*. 2007).

PbD has been applied in many industrial robot applications, such as maintenance, assembly, etc. However, there are several constraints in the PbD approach that need to be addressed. One key issue is that sub-optimalities often exist in the demonstrations with respect to both the robot and the learning system as a human may not be able to emulate the ability of a robot in terms of speed and accuracy when demonstrating the tasks (Chen and Zelinsky 2003; Aleotti and Caselli 2005). Kaiser *et al*. (1995) identified the sources of sub-optimalities that might occur with human demonstrations and provided a method for task refinement in skill acquisition through pre-processing sampled data, evaluating performance, and the further refinement of the skill. Chen and Zelinsky (2003) proposed a method to cope with the sub-optimalities in two steps, namely, (1) build a partial knowledge of geometry and represent the trajectories set as a curved surface in the configuration space, and (2) incrementally determine more optimal paths than those demonstrated ones based on the information obtained in step (1).

Another issue in PbD is the presence of noise in the data collected due to variations, such as jerks and jitters in the human demonstration. Multiple demonstrations are more practical when the task is to be executed many times and a higher quality of performance is required. This justifies the additional effort put in to obtain the sample data needed for learning (Aleotti and Caselli 2005).

2.2.2 Off-line Robot Programming

In off-line programming, the robot trajectory is generated with the assistance of a remote computer console and subsequently translated into the actual robot

controller codes. The operator does not need to be present within the working area of the robot. However, off-line methods require detailed knowledge of the work cells (including robot models) and the operator needs to acquire a particular set of programming skills, thus making it more complex than the lead-through and the walk-through programming methods. Nevertheless, off-line programming offers significant flexibility in evaluating robot planning solutions through simulation, and allows the robot to continue with the task while the operator programs a new task and the associated sequence. Examples of off-line programming systems are text-based systems, icon-based systems, and VR-based (or CAD-based) systems, which will be reviewed in the next sections.

2.2.2.1 Text-based Robot Programming

Text-based systems have been commonly used for programming industrial robots, and can be distinguished by the type of programming language used, e.g., the controller-specific languages, generic procedural languages, behavior-based languages, etc.

The controller-specific language is the original method for programming industrial robots as each robot controller has some form of machine language that can be used to create executable robot programs. Examples of controller-specific languages are Advanced Control Language (ACL 1999) for Scorbot robot, KUKA robot language (KRL) (KUKA 2011) and ABB robot programming language (RAPID) (ABB 2011), etc. A major drawback of controller-specific languages is the lack of a universal standard between languages from different robot

manufacturers. Generic procedural languages provide an alternative approach to controller-specific languages for programming robots. These languages can be Java (Kanayama and Wu 2000), C++ (Hopler and Otter 2001), Python (Pyro 2003), etc. They have been extended to provide robot-specific functionalities or abstractions, which could be a set of classes, methods, or operations that provides access to common robot functions. However, these methods suffer from the same drawback as controller-specific languages for industrial robots. Behavior-based languages provide an alternative to the generic languages. They typically specify the way that a robot should react to different conditions with very few codes, rather than providing a procedural description. One famous behavior-based method is the Functional Reactive Programming (FRP) (Hudak *et al*. 2003), where both continuous and discrete events can be used to trigger actions.

2.2.2.2 Icon-based Robot Programming

Icon-based robot programming methods typically use a graph or flow-chart view of the robot system, and provide graphical medium for robot programming. The advantage of the icon-based methods over the text-based methods is the modular design and implementation of the common robot functions, enabling rapid configuration of robot programs to perform a required task. PROGRESS (Naylor *et al*. 1987) is an example that uses graphics to interact with the sensors and actuators to simulate the robot functions and their logical conditions. Friedrich *et al*. (1997) have proposed a 3D-icon based method to support and facilitate the programming process in a PbD system. MORPHA (Bischoff *et al*. 2002) is a prototype style guide for defining the icons in a flow-chart system, where a touch

screen was adopted for easy robot programming and programs re-configuration. Other graphical systems using the flow-chart approach can be found in Grape (Enderle 2009), Lego Mindstorms robotics kit (Lego 2011), etc.

2.2.2.3 Virtual Reality in Robot Programming

Virtual Reality (VR) is an off-line programming method aimed at increasing the intuitiveness of the robot programming task for the human operators in a 3D CAD environment. Several VR-based applications have been developed to enhance HRI in robotics for tele-operations, surgery operations, and many other operations in industrial environments.

Tele-operation tasks are usually performed in highly dangerous environments where the presence of human is not advisable due to the safety issues. A Virtual Environment (VE) aided by the necessary sensors could offer the operator an immersive sense of his/her presence at the real location undertaking the tasks (He and Chen 2009). The VE allows the operator to project actions that are carried out in the virtual world onto the real world by means of robots. The use of an interaction device, such as a PHANToM (Kim *et al*. 1999), etc., can facilitate the programming process, such that the operator can feel the scaled-down representation of the dynamics of the slave manipulator.

Another category of VR-based applications is in surgical robots where a set of pre-determined robot motions are constructed, identified, processed and transferred to a real robot for execution, e.g., needle insertion, surgery, etc.

(Burdea 1996). Such systems often provide a virtual training environment so that the medical operations can be simulated and practiced before moving to a real operation on a real patient. In these applications, force feedback using a haptic interface is normally acquired to achieve realistic simulations.

Natonek *et al.* (1995) described a VR system for the training of robots for object manipulation where the environment is known *a priori*. Navon and Retik (1997) proposed a method to program construction robots using VR techniques in a frequently changing construction environment. In this study, the immersive programming mode was found to be more convenient than the non-immersive programming mode. In the immersive mode, the modeling of the entire working environment is required and advanced sensors are needed to obtain sufficient detailed information of the environment. Neto *et al*. (2010a; 2010b) presented a CAD-based system, using commercially available CAD package, to simulate and program a robot path so as to allow users with basic CAD skills to generate robot programs off-line without intervening robot operations. In the free-form surface manufacturing process, the framework for CAD-guided tool path planning has been developed based on the CAD model of a part, tool model, task constraints, and optimization criteria (Sheng *et al.* 2000). A transformative robot program generation method (Chen and Sheng 2011), which extends the CAD-guided tool path planning method, has also been developed such that robot programs for a new workpiece can be generated based on existing parts in the database, which contains the robot paths and programmers' knowledge and process parameters.

A major advantage of using VR is the provision of various evaluation options to assist robot programming due to its scalable modeling capability of the entire environment where a robot works in (Žlajpah 2008). The majority of the VR systems suffer from the same disadvantage, which is the need for extensive modeling of the environment to construct the virtual system, making them restrictive in unknown and/or unstructured environments. In addition, careful calibration and fine tuning is needed to compensate for inaccuracies of the virtual models when replicating simulated environments. The delay between the VR display of the movements of a remote robot and its physical movements is still considerable. Despite the use of various types of sensors in the remote working place for retrieving as much information as possible, the feedback from sensors would also be delayed. This is essentially not applicable for tasks that are carried out in a frequently changing environment. Systems that adopt a totally immersive environment, e.g., Cave Automatic Virtual Environment, provide more flexibility to the users, but are achieved at the expense of a more complex and costly structure. Typically, offline programming packages require the users to master complex programming languages, i.e., controller-specific or generic languages.

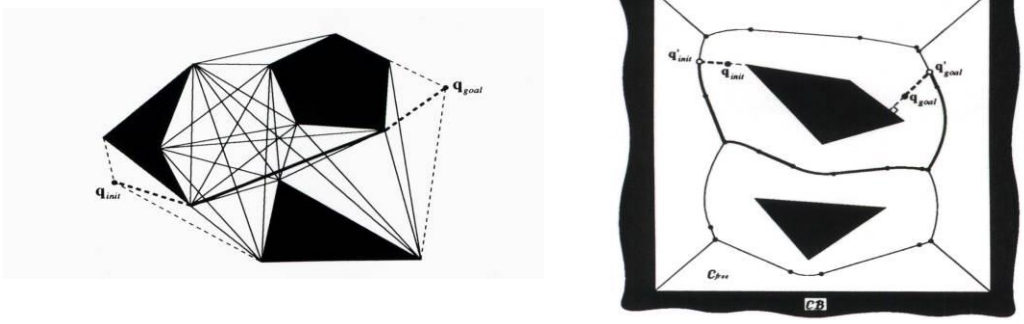
2.3 Robot Motion Planning

Robot motion planning refers to a process that finds a collision-free path between the initial and goal configurations among obstacles for a robot (Latombe 1991; Craig 2005). These motion planning algorithms can be categorized in terms of the completeness (exact or heuristic) or the scope (global or local) of the algorithms.

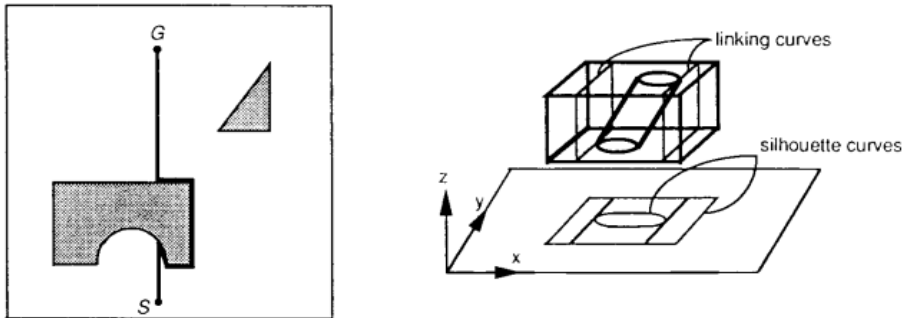
Exact algorithms can either find a solution or prove that there is no solution, while heuristic algorithms are aiming at generating a solution in a shorter time. Heuristic algorithms are important and frequently employed in engineering applications. However, they may find a poor solution, or fail to find one for difficult problems. Global algorithms first acquire a complete description of the free space where the robot can move safely, and subsequently produce a path from the initial to goal configuration (if there exists one). Local algorithms use information about nearby obstacles in the vicinity of the robot only. Local methods are often used as a component to a global planner to avoid unexpected obstacles not depicted in the model of the environment but detected by sensors during motion execution.

2.3.1 Classical Motion Planners

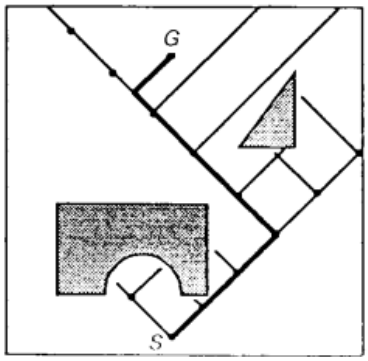
Most classical motion planners are based on either one or a combination of (1) skeletons, (2) cell decomposition, (3) potential field, and (4) sub-goal graph. Skeletons, also known as roadmaps, are representations of the Configuration space (C-space) in the form of networks of one-dimensional paths, where the nodes correspond to distinct features (such as intersections in obstacles) and edges represent the relationships between the nodes. The well-known skeletons, as shown in Figure 2.1, are visibility graph, Voronoi diagram, silhouette, and sub-goal network.



(a) Visible graph (b) Voronoi diagram



(c) Silhouette



(d) Subgoal network

Figure 2.1: Skeleton examples (Latombe 1991; Huang and Ahuja 1992)

The visibility graph is a collection of lines in a free space, such as in the form of a C-space, which connects a feature of an object to another (Lozano-Perez 1983b). Some algorithms have been adopted to find the shortest path in the visibility graph, such as the Dijkstra algorithm (Asano *et al*. 1985), A* search algorithm (Montgomery *et al*. 1987), etc. In a Voronoi diagram, a path can be found by first

moving the robot to the initial configuration, along the diagram, and then to the goal configuration (Kambhampati and Davis 1986). In a high-dimensional environment, the object of the higher-dimensional space can be projected to a lower-dimensional space, and the boundary of the projection can be traced out to form a silhouette (Canny 1988). The silhouette simplifies the path planning process by reducing the search space, and is usually used for complexity analysis in theoretical algorithms.

In the cell-decomposition approach, the C-space is represented as a union of cells, such as an Octree (Brooks and Lozano-Perez 1985; Hamada and Hori 1996), etc., and a path is given by a sequence of cells, which can be found using search algorithms, such as the A* search algorithm (Faverjon 1984; Brooks and Lozano-Perez 1985), greedy search algorithm (Jun and Shin 1988), etc. The potential field-based approaches have often been used for obstacle avoidance. Artificial potential functions are designed in the local path planner for imminent collision avoidance among robot arms and obstacles (Khatib 1985). Global path planner for obstacle avoidance can also be achieved by combining the idea of configuration space and potential functions (Warren 1989). In addition, the artificial potential field can be used to model the workspace, rather than the C-space, to achieve collision-free path planning for robots of high degree-of-freedom (DOF) (Lin and Chuang 2003). The sub-goals graph is a two-level planning approach typically combining the cell-decomposition and potential-field approach, in which a graph of sub-goals is maintained by a global planner, and the accessibility among sub-goals is determined by a local planner.

This approach is particularly efficient in finding collision-free paths in heavily clustered environments (Faverjon and Tournassoud 1987).

2.3.2 Decoupled Motion Planning

A robot path can be represented by a series of discrete robot configurations (or control points), either in the joint space or the Cartesian space. To transfer the path into executable robot programs, each discrete robot configuration needs to be assigned with a time stamp indicating when it will be reached. This process refers to robot trajectory generation. The general consideration in trajectory generation is the smoothness of the motion, i.e., continuous in position, velocity, and acceleration for each DOF. Many schemes for trajectory generation have been developed as any smooth function of time that passes through control points can be used to specify and quantify the exact path shape. Such functions can be cubic polynomial, high-order polynomial, linear function with parabolic blends, etc. (Craig 2005; Angeles 2007). Some other motion planning schemes have been proposed to consider additional constraints on the dynamics of the robot links and the actuator limits. To reduce the computational complexity and resources required in solving motion planning problems, the decoupled approach, which decomposes the original problem into two sub-problems, i.e., geometric path planning and trajectory tracking, has been well studied and reported (Lin *et al*. 1983; Bobrow *et al*. 1985; Shin and Mckay 1986; Shiller 1994). Several optimization criteria have been reported, such as the minimum-time trajectory planning (Bobrow *et al*. 1985; Shiller 1994), minimum-jerk trajectory planning (Constantinescu and Croft 2000), and hybrid trajectory planning, where a

trade-off is made between the path duration and the mechanical energy of the actuators (Duleba 1997; Verscheure *et al*. 2008), etc., for a given task on industrial robots. A variety of search methods, e.g., dynamic programming (Shin and Mckay 1986; Lee 1995), genetic algorithms (GA) (Yun and Xi 1996; Yue *et al*. 2002; Zha 2002), etc., have been applied to robot motion planning problems to find the optimal solution subject to the constraints imposed. Verscheure *et al*. (2009) reformulated the time-optimal trajectory planning problem into a convex optimization problem (Bobrow *et al*. 1985) and solved it using solvers based on computationally efficient interior point methods (Boyd 2004). These methods can solve optimization problems with equality and inequality constraints by reducing them to a sequence of linear equality constrained problems. They yield good worst-case complexity by exploiting the problems' structure efficiently.

2.4 Brief Review on Augmented Reality

AR augments the users' views with additional information by superimposing computer-generated graphics and text over real objects/scenes, letting the users interact with both virtual and real objects simultaneously in real-time (Azuma 1997). One of the most successful applications of AR has been in the movie industry, where computer graphics blends with real life footages making it difficult to discern the differences between the real scene and the virtual entities. The majority of AR research has been concerned with using it as a form of Human Computer Interface (HCI), where the augmented environment enhances the user's real-time interaction with the real world. An AR system is illustrated in

Figure 2.2. The technical research issues concerned with the implementation of an AR system, as shown in Figure 2.3, are tracking and registration, interface for human-machine interaction, etc. These issues need to be addressed in an AR-based robotic system.

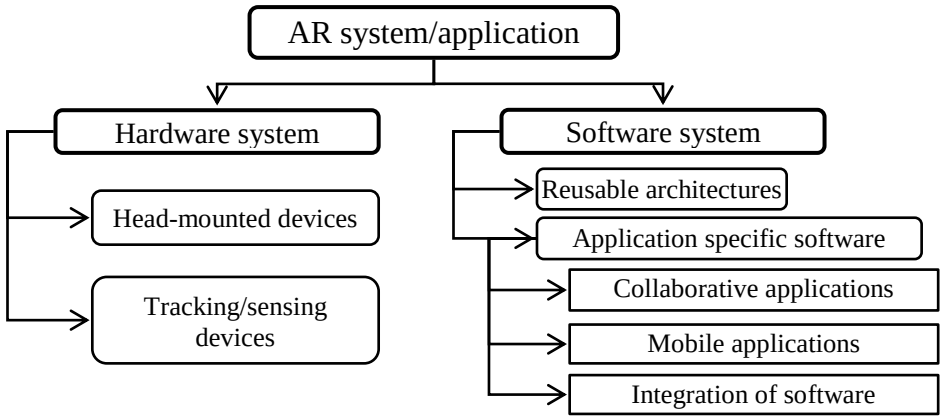


Figure 2.2: Basic Components of an AR system.

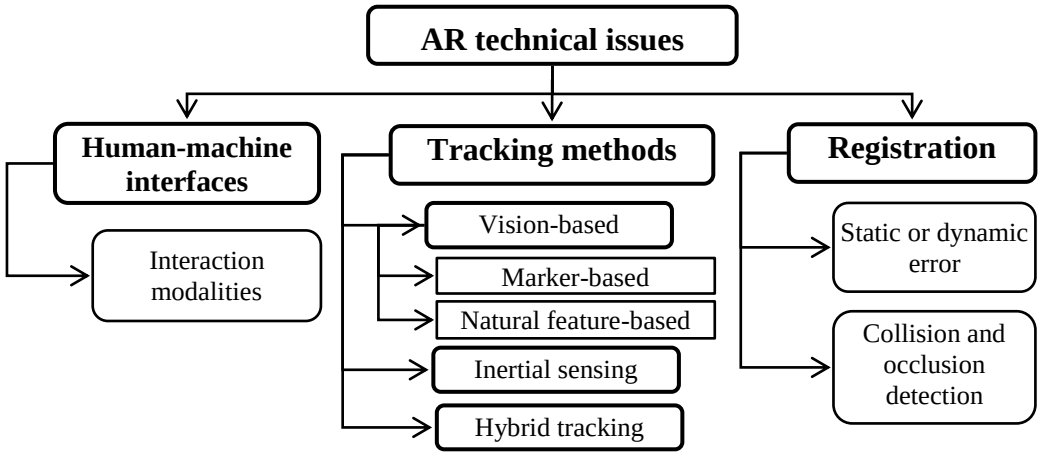
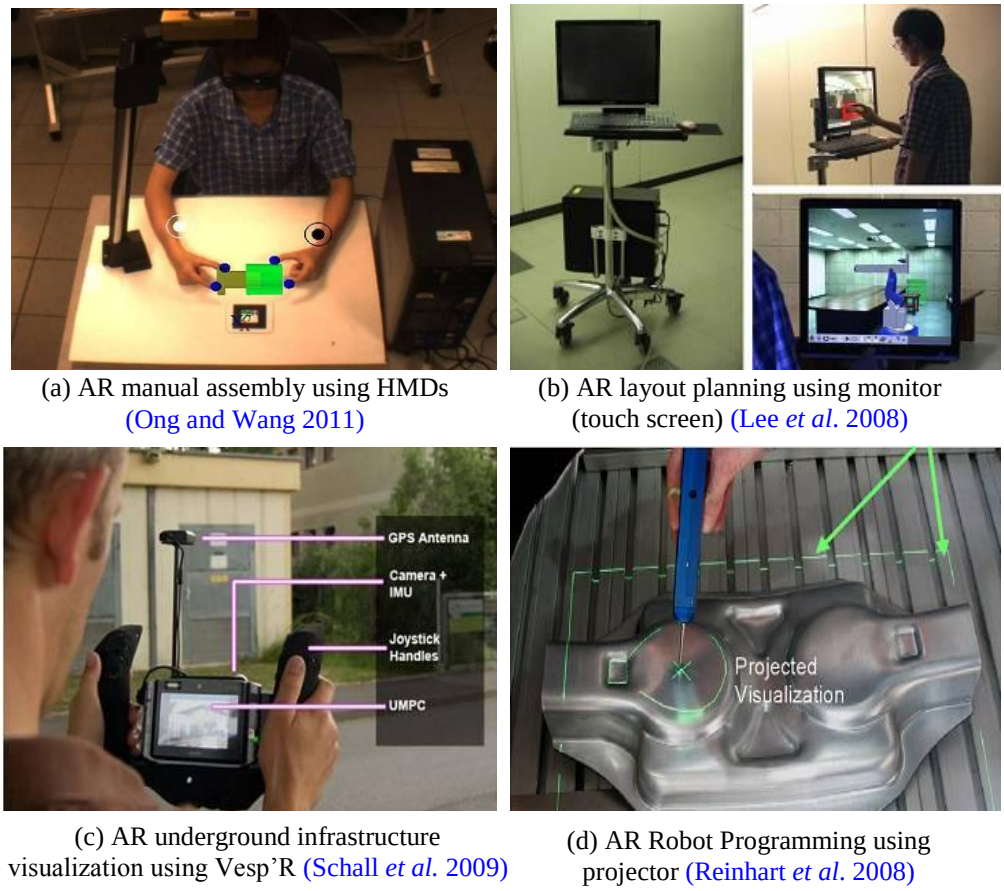


Figure 2.3: Technical issues to be considered in developing AR systems.

2.4.1 Hardware Systems

There are several types of display hardware in AR systems, namely, head-mounted displays (HMDs), monitors, mobile devices and projectors. Figure

2.4 shows a few display systems for AR applications. In particularly, Vesp'R is an ergonomic handheld mobile device designed around an ultra-mobile PC.



(a) AR manual assembly using HMDs (Ong and Wang 2011)

(b) AR layout planning using monitor (touch screen) (Lee *et al.* 2008)

(c) AR underground infrastructure visualization using Vesp'R (Schall *et al.* 2009)

(d) AR Robot Programming using projector (Reinhart *et al.* 2008)

Figure 2.4: Display hardware in AR systems.

There is a growing trend of using handheld devices in AR systems (Schraft and Meyer 2006; Xin *et al.* 2008; Schall *et al.* 2009; SMErobot 2009) due to the fact that mobile devices have become more popular and powerful. Studies on the possibility of using mobile devices, namely, PDA, to facilitate intuitive robot programming have been reported (Schraft and Meyer 2006; SMErobot 2009). Projectors have been employed in some AR systems to achieve wider field of view (FOV), compared with the HMDs or mobile devices which normally have

narrow FOV. Examples of such projector-based systems are the system reported by Olwal *et al*. (2008) to enhance the visibility of the machining tools and real-time data from machining process and the systems reported by Zaeh and Vogl (2006), Reinhart *et al*. (2008), etc.

There are five types of tracking devices that can be used in AR systems. They are based on time-frequency measurement, spatial scan, inertial sensing, mechanical linkages and direct-field sensing (Rolland *et al.* 2001). Each type of tracker has their weaknesses. Ultrasonic trackers suffer from noise and tend to be inaccurate at long range because of variations in the ambient temperature. Inertial trackers drift with time. Mechanical trackers are sufficiently accurate although they confine the user to a limited working volume and the accuracy is limited by the deformation and wear of the mechanical parts and the resolution of the encoder. Magnetic trackers are vulnerable to distortion due to the presence of metals in the environment. In robotic systems and applications, the majority of trackers used are vision-based (usually cameras, e.g., in visual servoing (Chaumette and Hutchinson 2006, 2007; Benhimane and Malis 2007), navigation of mobile robots (DeSouza and Kak 2002), etc.), inertial sensors (SMErobot 2009), or a fusion of vision and inertial sensors (Foxlin and Naimark 2003; Ribo *et al*. 2004; Ababsa and Mallem 2007).

2.4.2 Software Systems

Some well-known AR software platforms have been developed. The reusable architectures of these AR software platforms allow researchers with different backgrounds to develop special-purpose AR applications.

ARToolKit (Kato and Billinghurst 1999) is a well-known marker-based platform that has been extensively used for creating real-time AR applications. It is an open-source and free multi-platform that tracks pre-designed markers with arbitrary shapes to recognize different targets on which the virtual objects can be superimposed. ARToolKitPlus (Wagner and Schmalstieg 2007) is an extended version of ARToolKit and aims at handheld AR applications. ARToolKitPlus can detect markers with variable border width, and can recognize up to 4096 simple id-encoded markers. In addition, it has implemented a new pose estimation algorithm (Schweighofer and Pinz 2006) for more stable marker tracking.

Parallel Tracking And Mapping (PTAM) (Klein and Murray 2007) is a fast, accurate and robust camera tracking system for AR whereby a detailed 3D map of objects in an unknown environment can be obtained and maintained through tracking the natural features of the environment. It does not require markers, pre-made maps, known templates, or inertial sensors, and turns real world surfaces into platforms for virtual objects or characters. Parallel Tracking And Multiple Mapping (PTAMM) (Castle *et al*. 2008) extends the PTAM to allow maps of multiple workspaces to be made and individual AR applications to be associated with each map. PTAMM can switch between the maps automatically

as the user explores the world. There are some projects that have used PTAM for tracking, such as VideoTrace (Van Den Hengel *et al*. 2007), ProFORMA (Pan *et al*. 2009), live dense reconstruction (Newcombe and Davison 2010), etc.

A few other AR software platforms have been developed. ImageTclAR (Owen *et al*. 2003) is a multimedia development environment that provides a filter capable of merging tracking data from multiple trackers. Aura (Garlan *et al*. 2002), Studierstube (Schmalstieg *et al*. 2002), etc., have been developed to allow multi-users collaboration in 3D virtual or AR environments. Other AR platforms, such as MARS (Hollerer *et al*. 1999) and ARVIKA (Friedrich 2002), etc., have been developed for mobile AR applications. ARVIKA is primarily designed for industrial applications, and it adopts an integrated approach with different input/output devices, such as speech devices, for hands free applications. MARS is a platform based on the Coterie distributed VE infrastructure that enables indoor and outdoor user interaction, by which indoor users can assist the outdoor users through sketching paths or pointing out objects of interest. Several research projects investigated the possibility of integration two or more AR platforms to take advantage of the strengths of each platform. Studierstube is an OpenInventor-based component-oriented platform with integration of OpenTracker (Reitmayr and Schmalstieg 2001).

2.4.3 Tracking Methods

Accurate tracking with fine resolution and scalability is crucial in developing a successful AR system in robotics and other manufacturing fields. Different types

of tracking methods are available, and each tracking method has its weakness, as discussed in Section 2.4.1. Compared with other types of tracking methods, spatial scanning has the potential to yield non-invasive, accurate, and cost-effective tracking solutions. The most common scenario in a vision-based 3D tracking system is to use one or more cameras and the relevant CV technologies to detect targets in the image and find their position and orientation.

2.4.3.1 Ultrasonic-based Tracking Methods

Ultrasonic devices, which operate based on time-frequency measurement, have been used for seam tracking in robot welding. Mahajan and Figueroa (1997) presented a seam tracking method, with two ultrasonic sensors mounted on the EE just ahead of the welding torch, for the detection of freely curving seams on two-dimensional (2D) surfaces. Kuang and Morris (1999) developed an ultrasonic-based robot tracking system where a transmitter was mounted onto the EE of a robot and eight receivers were arranged locating at the corners of a virtual cube around the robot. It can be used to track the position of the EE in this cube using geometric position-calculation algorithm. The combination of Time-of-Flight (TOF) and Doppler-Distance (DD) has been investigated to improve the tracking performance (Kuang and Morris 2000a). It has been proven that by employing one additional receiver, the ultrasonic speed variation introduced by the environmental noise could be compensated (Kuang and Morris 2000b). Ultrasonic sensors have been equipped for the localization of mobile robots in unstructured environments (Choi and Lee 2008, Lin *et al*. 2008).

2.4.3.2 Laser-based Tracking Methods

Laser trackers have been used in robot applications increasingly as they usually yield high accuracy. One disadvantage of laser trackers is the high cost of the laser devices. Many companies and research groups, such as Leica Geosystems (Switzerland), FARO (USA), API (USA), etc., have developed commercial laser devices that can be used for 3D tracking, but the price is much higher than cameras with acceptable resolution. Bai *et al.* (2003) proposed a laser-based system for robot calibration through tracking the target that is mounted on the EE of the robot. Laser devices used for trajectory tracking can assist operators in robot programming process, as shown in Figure 2.5. Through tracking the input device, the trajectory can be visualized, manipulated and recorded by means of laser projection (Zaeh and Vogl 2006; Reinhart *et al*. 2008). Laser trackers can also be employed as range finders in mobile robots to acquire spatial information of the working environment (Collett and MacDonald 2006; Livatino *et al*. 2010).

2.4.3.3 Vision-based Tracking Methods

Vision-based tracking methods can be divided into two categories, namely, marker-based and natural feature-based (marker-less). Marker-based tracking determines the position and pose of the centre of the camera with respect to a given marker. The shape of the marker can be circular, linear, triangular or square. The use of markers increases the robustness and reduces the computational requirements as the detection of the markers in the entire image sequence is stable and fast. However, occlusion is an issue that cannot be overlooked as virtual objects cannot be augmented onto the markers if the markers are partially

occluded or outside the FOV. Though the accuracy and jitter-free levels are lacking and markers are not desirable in many situations, ARToolKit (Kato and Billinghurst 1999) still remains as the entry point for the development of many AR systems. Many marker-based systems have been developed and they offer better performance (Foxlin and Naimark 2002; Wagner and Schmalstieg 2007).

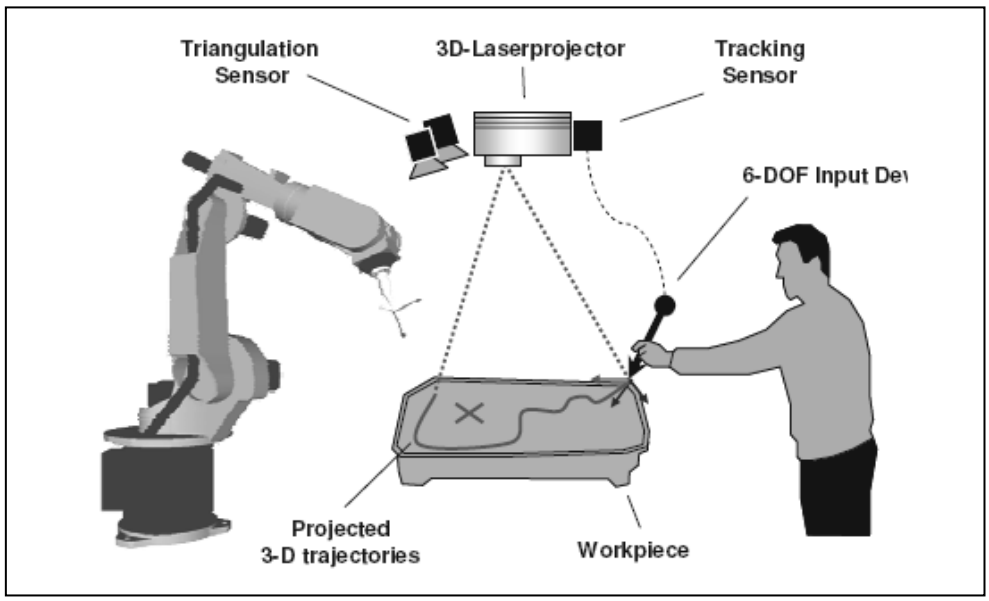


Figure 2.5: Laser tracking method for robot programming (Zaeh and Vogl 2006).

The conditions of an outdoor environment, e.g., poor lighting condition, unknown moving objects, etc., will lead to poor tracking performance when markers are used. Numerous markerless CV-based AR trackers have been proposed either through tracking with known 3D scene structures or tracking with natural features. The 3D scene structures are normally available beforehand, or can be obtained while tracking (Klein and Murray 2007; Klein and Murray 2008) with robust feature tracking algorithms. For natural feature tracking, the main difficulty lies in solving the correspondence problem, or matching of feature points projected in

two or more views. Some of the most promising techniques are the Scale Invariant Feature Transform (SIFT) (Lowe 2004), the Random Trees (RT) (Lepetit and Fua 2006), the Speeded Up Robust Features (SURF) (Bay *et al*. 2006), the Binary Robust Independent Elementary Features (BRIEF) (Calonder *et al*. 2010), the Keypoint recognition based on classification using Naïve Bayesian (Ferns) (Özuysal *et al*. 2007), and the Efficient Second-order Minimization (ESM) (Benhimane and Malis 2007). Such advances of CV algorithms can achieve real-time tracking and matching performance even on devices with limited computational power. These can be used in mobile robots to improve real-time localization and mapping, and exploration of unstructured environments.

2.4.3.4 Hybrid Tracking Methods

Hybrid tracking methods integrate the advantages of two or more methodologies. A hybrid tracking method, combining vision-based and mechanical tracking principles, was presented by Bischoff and Kurth (2006). As shown in Figure 2.6, through hand-eye calibration, the position of the marker with respect to the robot base can be obtained. Using vision-based tracking, the position and pose of the camera with respect to the marker can be calculated. Using mechanical tracking, the coordinate of the robot EE in the robot base coordinate system can be computed. Hence, the position and pose of the camera with respect to the robot EE can be retrieved, and a close loop tracking is achieved. However, this hybrid method has a small tracking range as the operation range of the robot is limited. The tracking system as shown in Figure 2.5 can also be regarded as a hybrid system. The two Infra-Red (IR) cameras are the input devices. The visualization

of the trajectory planning process is handled by the laser system. As the robot is not within the FOV of the cameras and the projection range of the laser system, other methods will be required to achieve the robot registration. Ng *et al*. (2010) proposed a hybrid method for robot tool path planning by fusing information from a camera and a laser range finder. The laser range finder is used to obtain the 3D information of a point of interest in the working environment, and a camera is used to provide the video streams for visualization and virtual contents rendering.

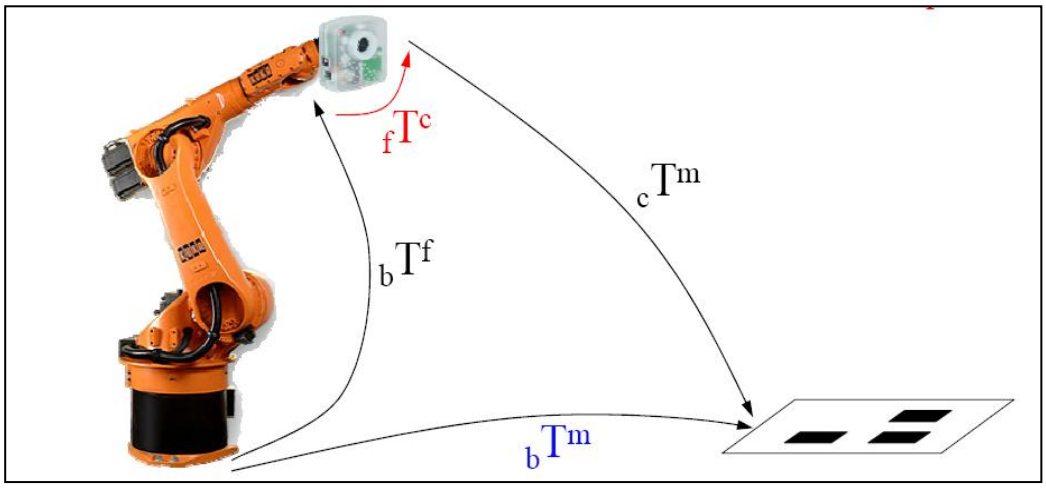


Figure 2.6: A hybrid robot tracking system comprising of vision-based tracking and mechanical tracking (Bischoff and Kurth 2006).

The combination of inertial sensor and marker-less CV algorithms is another commonly used approach for hybrid tracking (Foxlin and Naimark 2003; Ababsa and Mallem 2007; Ribo *et al*. 2004). This can be attributed to the fact that inertial, marker-less CV, unlike ultrasonic or magnetic sensors, are practically source-less, i.e., they do not require a specialized emitter to work. In such systems, particle filters can be used for the fusion of the inertial and marker-less CV trackers (Ababsa and Mallem 2007). There are other approaches combining inertial, GPS

and marker-less CV in a single hybrid tracker, e.g., Reitmayr and Drummond (2006), Fong *et al*. (2009), etc., where the GPS and inertial sensors are used to define an initial search space, and the CV component tracks the natural features, such as points, lines and textures, in the search space.

2.4.4 Registration

Registration is an important issue that currently limits the application of AR in robot programming. It is concerned with the seamless blending of the virtual computer-generated graphics and real entities for visualization. For example, in robot trajectory planning, the virtual robot should be superimposed precisely onto the location of the real robot; otherwise the real robot cannot replicate the planned path accurately, leading to a failed operation in task execution.

Registration methods can be divided into two categories, namely, CV-based registration and sensor-based registration (Azuma 2001). CV-based registration can be achieved through marker-based methods (Bischoff and Kazi 2004), or natural feature-based methods (Yuan *et al*. 2006). Sensor-based registration methods use sensors to determine the position and/or orientation of the real objects that are to be replaced by their virtual duplicates. However, sensor-based methods are often combined with CV-based methods to achieve the required accuracy (Behringer 1999; You *et al*. 1999).

The desirable registration accuracy depends on the intended application. For example, more accurate registration is required for medical applications involving

the overlay of images for surgery, as compared to simple maintenance tasks that usually involve only text annotations. Satoh *et al.* (2006) used an industrial robot to perform quantitative evaluation of the registration techniques in a mixed reality environment. By identifying the evaluation factors, which include registration accuracy, size of space/range being registered, tolerance, integrated cost, etc., several evaluation criteria were defined. It was found that there is a trade-off between the registration system complexity and the registration performance. Hybrid registration schemes always need auxiliary devices or expensive sensors to be attached to the existing systems. It is not always necessary to design a complicated system to achieve the required registration accuracy, especially as the recent achievement in natural features tracking in both indoor and outdoor environments permits more stable and robust registration performance.

2.5 Augmented Reality in Robotics

AR has been applied successfully in many manufacturing processes, such as manufacturing planning (Gausemeier *et al*. 2002), manual assembly (Reinhart and Patron 2003; Yuan *et al*. 2008; Ong and Wang 2011), in-field or remote maintenance (Henderson and Feiner 2009; Zhu *et al*. 2011), CNC machining simulation (Olwal *et al*. 2008; Zhang *et al.* 2010), collaborative design and view management (Shen *et al*. 2009; Shen *et al.* 2010), etc.

In robotics, AR can assist the users in pre-operative planning as well as real-time tasks implementation. It allows the operators to use their natural spatial

processing abilities to obtain a sense of presence in the real world with virtual information. AR offers the possibility to visualize the motions and trajectories of a robot augmented onto the real environment; it enables the users to interact intuitively with the spatial information. Table 2-2 summarizes the major research efforts worldwide in using AR in robotics, tele-robotics and their applications.

2.5.1 Augmented Reality in Robot Path Planning

AR has been applied in various robotics applications, e.g., tele-operations (Rastogi *et al*. 1996) and robot surgeries. Marin *et al*. (2005) presented a mixed reality-based visualization interface where virtual cues can be displayed to the users for overlapped objects isolation and grasping operations conducted at a remote site. The system provides different HRI channels, namely, vocal-based and high-level text-command-based, which enables the users to choose a suitable interaction modality with the robot. Chou *et al.* (2004) introduced a virtual neurosurgical robotic system to assist surgeons in selecting the most suitable robot pose for a puncturing operation. An AR-based cueing method was reported (Nawab *et al*. 2007; Chintamani *et al*. 2010) to assist the users in moving the EE of a real robot using two joysticks. The visual cues, which associate the orientation and translation of the EE with the movement of the joysticks, allow the users to navigate the robot in a tele-operation task intuitively under display-control misalignment conditions. These studies show positive effects of using AR on operator performance in *ad hoc* tele-robotic tasks.

Table 2-2**:** Major AR research in robotics and tele-robotics

Groups/ Projects	Institutes	Area	Methodology/Feature
Ergonomics in Tele-operation & Control Laboratory (Rastogi *et al*. 1996)	University of Toronto (Canada)	Industrial robot	(1) Stereo-vision method (2) Human-tele-robot interaction (3) Virtual tape-measure (4) Stereoscopic graphics and video
Centre for Medical Robotics and Computer Assisted Surgery (Simon *et al*. 1995; Simon 1997; Simon and Kanade 1997)	Carnegie Mellon University (USA)	Medical/surgical robot	(1) Marker-based tracking method (six-DOF) (2) Intra-operative collection of tracking data in image-free systems (3) Shape-based registration (4) Patient-specific 3D template method
da Vinci Surgical System (Tewari *et al*. 2002; Hoznek *et al.* 2002; Hattori *et al.* 2003)	Intuitive Surgical Inc. (USA)	Medical/surgical robot	(1) Master-slave system (2) Optical 3D location sensation and digital video processing (3) Calibration/registration using optical marker
Robotics Intelligence Lab (Wirz and Marin 2004; Marin *et al*. 2005)	Jaume I University, Castellón (Spain)	Net-robotics (educational robot)	(1) Web-based system controlling online robot (2) Predictive display using AR (3) AR assistance on robot programming and monitoring (4) Remote programming (5) Multi-users management (6) Distributed architecture
Inami Laboratory and Tachi laboratory (Tachi *et al*. 2004; Tadakuma	The University of Electro-communications;	Medical/surgery robot;	(1) Real-time sensing (2) Haptic display (3) Synchronized interaction

et al. 2005; Shimizu *et al*. 2006	University of Tokyo (Japan)		(4) Retro-reflective projection for tele-communication with robot (5) Projector-based Tracking System
Anthropomorphic Robot Assistants Project(MORPHA) (Bischoff *et al*. 2002; Bischoff and Kazi 2004; Bischoff and Kurth 2006)	KUKA Robot Group (Germany)	Industrial robot (welding, painting, assembly, *etc.*)	(1) Monitor-based visualization with various simulation options (2) Optical tracking (with marker) and mechanical tracking (without marker) (3) Occlusion model (4) KUKA AR viewer (5) Instantaneous/real-time visual feedback
Augmented Reality in Assembly Planning (Zaeh and Vogl 2006; Reinhart *et al*. 2008)	German research Foundation; Technical University Munich (Germany)	Industrial robot	(1) Laser projection and VST- Visualization (2) Tracking with retro-reflective markers (3) Pen-like input and interaction tool (4) Trajectories planning and editing
AVILUS (AVILUS; Lieberknecht *et al*. 2009)	Federal Ministry for Education and Research (BMBF)	Industrial robot (heavy industry and SMEs)	(1) Integration of CAD/CAM models in Product Lifecycle Management process (2) Tracking of large measuring volume in the size of a factory (3) Tactile interaction metaphor (4) System ergonomics issues (5) Extension to mobile platforms
SMErobot (Schraft and Meyer 2006; SMErobot 2009; Hollmann *et al*. 2010; Neto *et al*. 2010b)	EU initiative	Industrial robot (SMEs)	(1) Robot programming by everyone (2) Multimodal interaction: e.g., tactile, verbal, visual (3) Speech Interaction: better signal to noise ratio (4) PDA integrated with KUKA Teach Wand (5) Touch screen based programming

Some AR-based robotic systems have been reported to address path planning issues, where a number of AR-based methods have been developed to facilitate human-virtual robot interactions. Chong *et al.* (2009) presented a method to plan a collision-free path through guiding a virtual robot using a probe attached with a planar marker. Zaeh and Vogl (2006) introduced a laser-projection-based approach where the operators can manually edit and modify the planned paths projected over a real workpiece using an interactive stylus. Reinhart *et al.* (2008) adopted a similar HRI approach (Zaeh and Vogl 2006) in robot remote laser welding (RLW) applications where the production cycle efficiency for the RLW process has been significantly improved with reduced set up and programming time.

2.5.2 Augmented Reality in EE Orientation Planning

Several approaches have been reported in the literature on the planning of the orientation of the EE of a robot along a path using AR. Ong *et al*. (2010) presented an approach for planning the EE orientations through defining interactively a number of coordinate systems with origins located at the parameterized points of the path. This method enables the user to modify the EE orientations to avoid possible collisions by re-defining the coordinate system at the beginning of the path. Reinhart *et al*. (2008) developed a computer-based approach for planning a smooth path along a weld seam by optimizing the EE orientations. The orientation at each point, except the first and the last, was regulated by two forces that are quadratically proportional to the distances to the two adjacent points.

The application of AR provides the operators with various simulation options during robot planning. However, the present research mainly focuses on geometric path planning problems considering the robot kinematics only. Chong *et al.* (2009) adopted the beam search strategy to find an energy-optimal path through evaluating the amount of pseudo-energy needed by the joints to execute a pick-and-place task. Zaeh and Vogl (2006) and Reinhart *et al.* (2008) projected the planned geometric paths that a robot needs on a real workpiece. The AR-based simulations presented in these systems are mainly conducted to verify the accessibility of the planned paths which are accessible without considering the motion constraints (e.g., joint velocity, joint acceleration, joint torque, etc.). In addition, such paths may not be suitable for translation directly into the robot program codes for execution.

2.6 Summary

This chapter provided a review of the existing approaches for robot programming, robot path planning, as well as AR applications in robot programming and path planning. The existing approaches for robot programming and path planning have been improved with new techniques in recent years, such as PbD, VR, CAD/CAM technologies, multi-modal interactions, etc. The use of AR in robotic systems can enhance the users' understanding of the working environment and facilitate the HRI in various robot tasks. The technical issues, e.g., the display, the real-time and robust tracking, as well as virtual object registration, have been reviewed in detail. Particularly, the issues of using AR in robot path planning

have been identified and discussed. Three research issues, which are intuitive robot path planning and simulation given a pair of start and goal configurations using AR, orientation planning of robot EE using AR, have been explored. However, the incorporation of robot dynamics to facilitate robot programming and trajectory planning in an AR environment has not been considered. The use of AR cues that can assist human operators in the planning of robotic tasks and simulation of the planned trajectories need to be further explored.

Chapter 3. RPAR-II System Description

3.1 Introduction

This chapter presents the RPAR-II system, namely, Robot Programming and Trajectory Planning using Augmented Reality. The hardware components, including a human-robot interaction device, for the setup of the system are introduced. A parametric robot model, which incorporates both the kinematics and dynamics properties of a serial-type robot, is described. An ARToolKit-based tracking method is adopted for tracking the interaction device and virtual robot registration. The relationships between the coordinate frames and transformations defined in the proposed AR environment and the robot kinematics configurations are then established. Lastly, the detailed system configurations are presented.

3.2 System Setup

The setup of the RPAR-II system is shown in Figure 3.1. It includes a serial robot arm mounted with an electrical gripper, a robot controller, a desktop-computer, a desktop-based display, a stereo camera (Figure 3.2(a)), and a hand-held interaction device attached with a marker-cube (Figure 3.2(b)).

3.2.1 Marker-cube

The use of a marker-cube to facilitate robot manipulation has been reported by Bischoff and Kazi (2004), where a marker-cube with six planar markers is attached to the end-effector (EE) of a real robot. It provides users with instantaneous visual feedback of the movements of the EE. For the marker-cube used in the RPAR-II system, four different fiducial markers are attached to the faces of the cube. The advantage of using multiple markers over a single one is that the cube can be tracked even if it has undertaken a relatively large rotation, e.g., 180 °, as there is a known relationship (rigid transformation) between each planar marker and the coordinate frame of the cube.

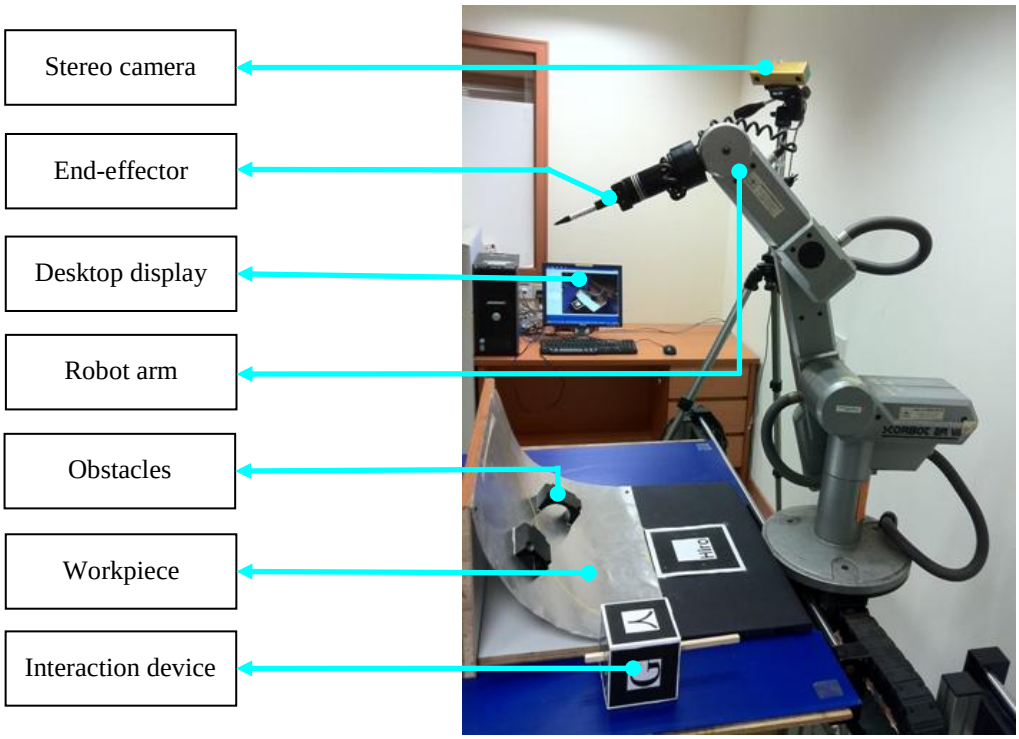
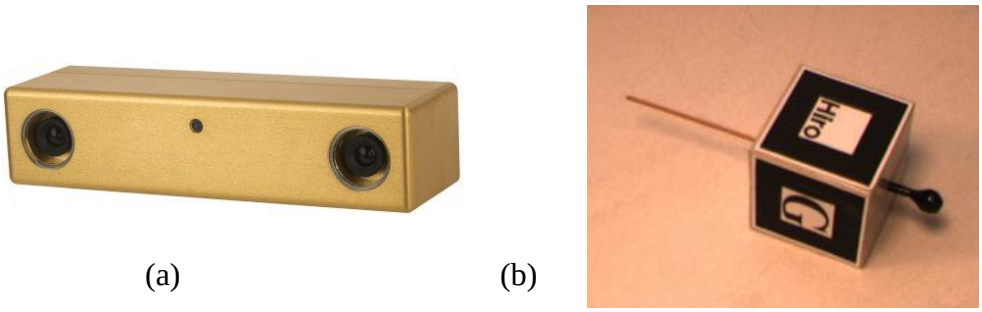


Figure 3.1: The RPAR-II setup.

With careful manipulation of the marker-cube, the camera is able to capture at least one marker in each frame, which is sufficient to obtain the position and orientation of the cube. The marker with the largest *confidence ratio* (Kato *et al*. 1999) is selected to define the coordinate system of the marker-cube.



(a) (b)

Figure 3.2: (a) *BumbleBee* stereo camera; (b) Marker-cube.

Each marker on the cube defines a single coordinate system. Thus, proper calibration is required to define a coordinate system consistent with the alternation of the tracked markers. The calibration procedure is as follows:

(1) Select a dominant marker and define the coordinate system of the interaction device with reference to this marker; and

(2) Obtain the transformations from the rest of the markers to the dominant marker, and assign each transformation to the corresponding marker.

After the calibration, the pose of the cube can be tracked smoothly such that the pose of the interaction device can be well defined. The jerks in the reference frame of the cube, which may be caused by the alternation of the recognized markers, can be reduced. In the proposed system, the coordinate system of the

interaction device is defined to locate at the tip of the probe, as shown in Figure 3.2(b), and the *Z*-axis is set to be the pointing direction of the probe.

3.3 RPAR-II System Configurations

3.3.1 System Framework

The framework of the RPAR-II system is shown in Figure 3.3. The main components of this system are as follows:

(1) The augmented environment consists of the physical entities that exist in the robot operation space, such as the robot manipulator, the working platform, tools, workpiece, etc., and a virtual robot to replicate its real counterpart;

(2) Interaction techniques to guide the virtual robot in robot task planning for point-to-point operations or path following operations, which include the generation of a suitable/relevant solution space, the generation of a smooth path within the solution space, and the determination of the orientations of the EE along the path;

(3) Trajectory optimization techniques to transform a geometric path into time-scale trajectory subject to robot dynamics so that the trajectory can be translated directly into the controller codes;

(4) Interaction simulation where the virtual robot is simulated to perform the planned motion to allow the user to evaluate visually the quality of the resulting trajectory, prior to the translation of the trajectories into robot programs.

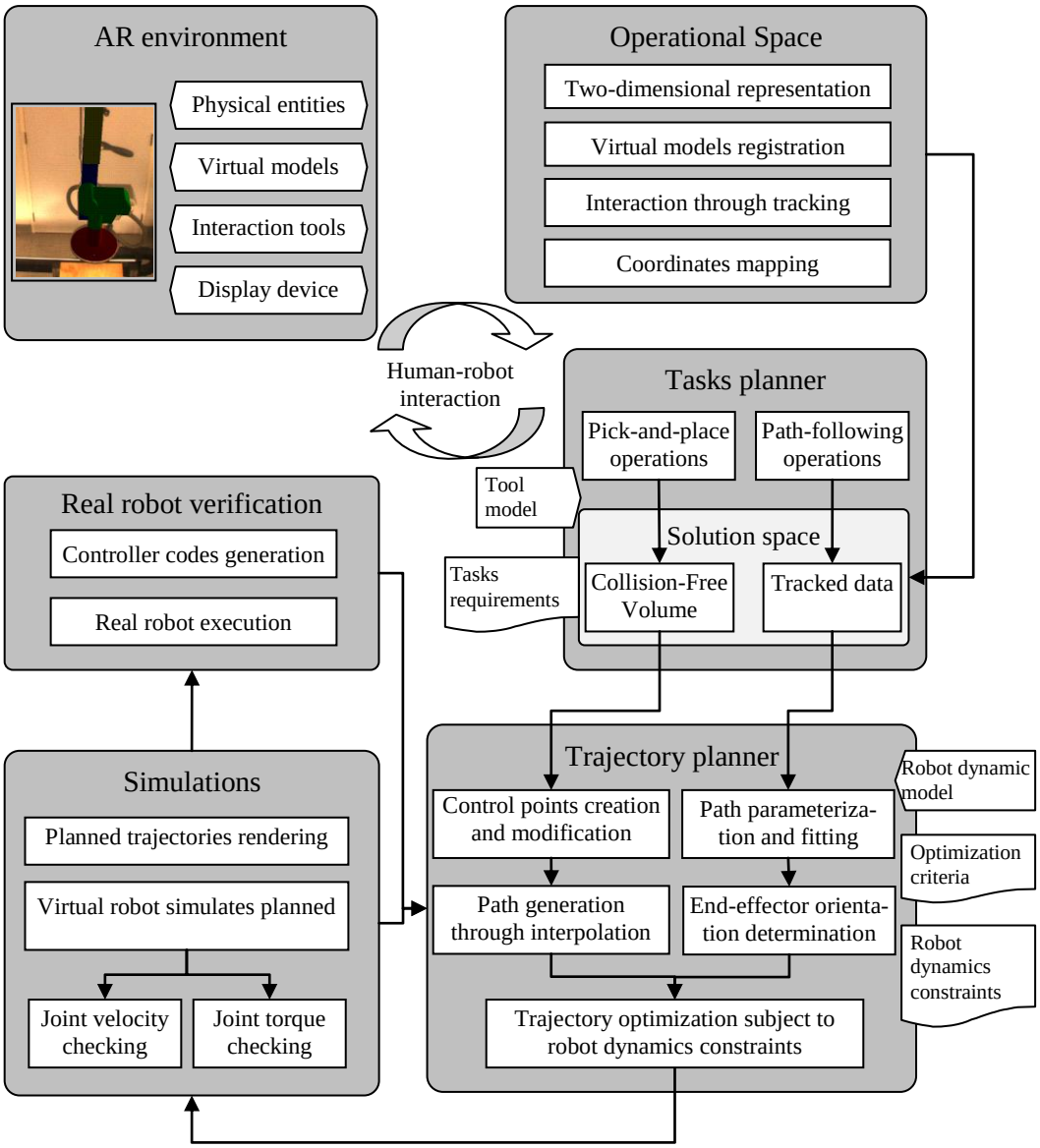


Figure 3.3: Architecture of the RPAR-II system.

It is common for small and medium enterprises (SMEs) to own a robotic system with multiple types of EEs for various tasks. The operating space of a robot is the space that the EE can reach with a full stretch of each link of the robot, and it changes when different EEs are used. The solution space, which is task-dependent, is a sub-set of the operating space. For instance, a collision-free volume (CFV), which is a free space relevant to a given robot task, can be generated through the

demonstration of a path using a virtual robot (Chong *et al*. 2007; Chong *et al*. 2009; Ong *et al.* 2010). The generation of a solution space can be achieved through precise registration and tracking of the interaction device. A user can see the actual environment where the robot will operate when a robot path is demonstrated; therefore, the solution space generated will be collision-free within the operating space of the robot. It can be observed easily that the solution may not be globally optimal since the solution space is a sub-space of the entire workspace. In other words, the proposed methodology is heuristic where the completeness of the algorithms is dependent on the completeness of the solution space with respect to the workspace.

Different demonstration and interaction routines are performed for different tasks. In a robot pick-and-place operation, such as materials handling, a CFV will be first constructed through human demonstrations. By specifying the starting and goal points, a user can create a number of control points within the CFV to generate a smooth path through interpolation. In a robot path following operation, the user needs to first guide the EE of the virtual robot following the desired path/curve for data collection. By defining interactively the orientation of the EE on each control point selected on the path, an orientation profile along the path is generated through interpolation.

Once the geometric path has been obtained, a trajectory is optimized subject to robot kinematics and dynamics constraints, and the task-dependent requirements. The trajectory planned is simulated using the virtual robot model under a control

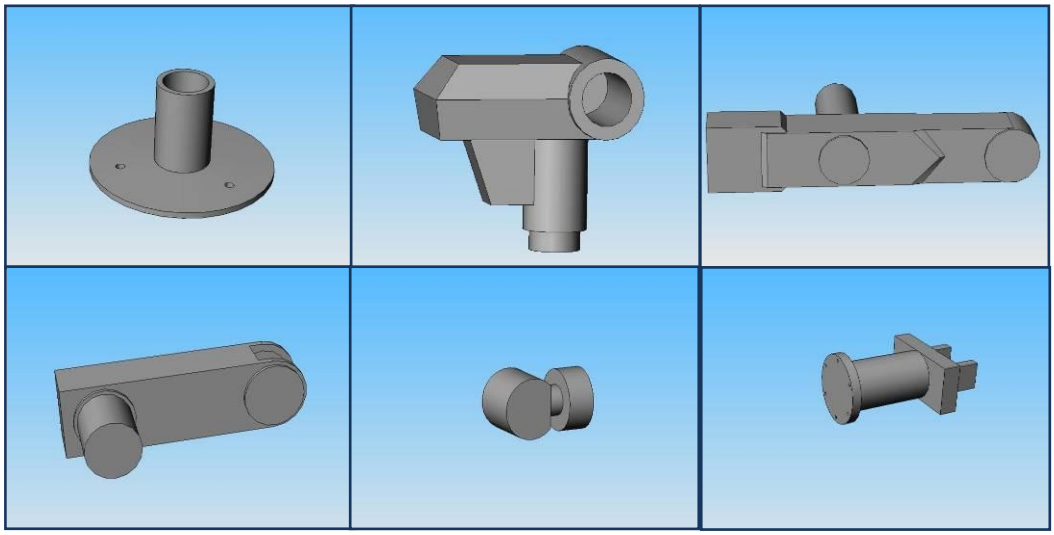
scheme, possibly with tuned control gains, to check the simulated joint velocity and joint torque. The associated information, together with the planned and simulated paths, is overlaid on the real workspace simultaneously to provide the user instantaneous visual feedback for inspection. The trajectory is finally translated into the robot controller codes and verified on a real robot.

3.3.2 Robot Model

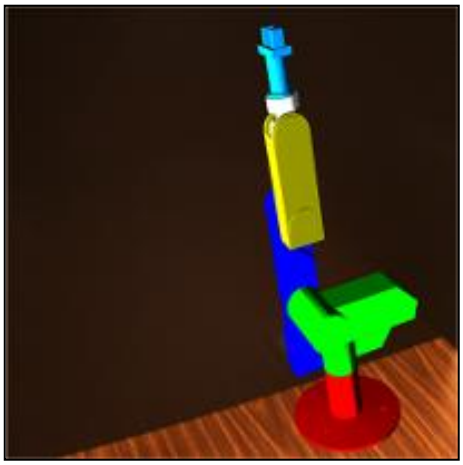
A valid model that represents the kinematics and dynamics properties of a robot will help the users understand the mutual relationship between the torque applied to each joint and the produced motion of the robot. Moreover, a more realistic simulation, i.e., the behavior of the given robot under given initial conditions, applied torques and loads, can be predicted appropriately by utilizing the robot dynamics model. In the proposed system, a Scorbot-ER VII robot is used.

3.3.2.1 Robot Geometric Model

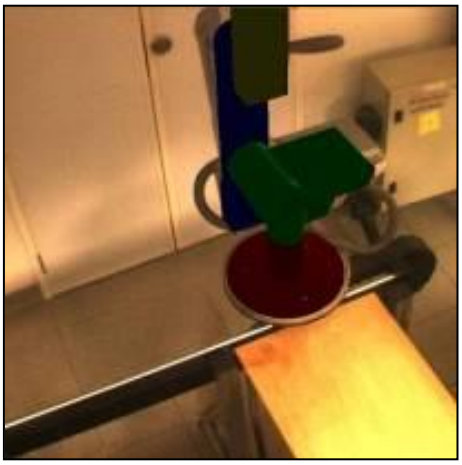
A virtual Scorbot-ER VII robot is modeled using SolidWorks®, as shown in Figure 3.4. The geometries of the coupled joints and links are modeled separately according to the kinematics parameters of the manipulator. The links are built with both functional and geometric features to enhance the visualization effects. Separate components are exported in the .STL format, which contains the geometric information with respect to the corresponding coordinate frame assigned to each link. The virtual robot model can be assembled and augmented onto the real environment, and scaled up or down for various robot configurations.



(a)



(b)



(c)

Figure 3.4: Geometric model of the Scorbot-ER VII manipulator.

3.3.2.2 Robot Kinematics Model

Conventionally, the kinematics configuration of a robot is characterized by its generalized coordinates, $\mathbf{q} = \left[q_1, q_2, .., q_{ndof} \right]$, where *ndof* is the degree-of-freedom (DOF) of the robot. For the Scorbot-ERVII robot, which has five revolute joints, the joint angles are denoted as $\mathrm{q} = \left[\theta_1, \theta_2, ... \theta_5 \right]$. The path for the EE of the robot is achieved by solving the inverse and forward kinematics.

The forward kinematics analysis is straightforward as any configuration of the robot in a joint space can be uniquely mapped to a configuration in the operation space. The classical kinematics solver is based on the Denavit-Hartenberg (D-H) convention (Craig 2005). The inverse kinematics can be solved analytically given the position and orientation of the EE with respect to the base of the robot. Setting the joint ranges enables the kinematics solver to eliminate some of the unwanted configurations caused by joint redundancy, such as the singular configurations at the boundary of the operation space. The EE of the robot can be kept at the last feasible configuration prior to the operation space violation (Chong *et al.* 2007). It is intuitive for a user when moving the EE of the robot along a given Cartesian path to verify whether all the points on this path are reachable, and/or whether the starting and goal points are reachable in the different solutions.

3.3.2.3 Robot Dynamics Model

The dynamics model of a robot is useful for the simulation of a robotic system; various robotic tasks can be examined without the need of a real robotic system. In this research, Recursive Newton-Euler (RNE) equations are adopted to compute the forward and inverse robot dynamics (Featherstone 2000). A user interface has been implemented by which a user can load or modify the robot kinematics and dynamics parameters easily in terms of each link of a serial manipulator, such as the D-H parameters, link mass, centre of gravity (CG), inertia tensor, motor properties, etc., during robot trajectory planning. This allows the users to evaluate the planned performance based on different robot parameters.

3.3.3 Coordinate Frames and Transformations

The coordinate frames and transformations defined in the RPAR-II system are illustrated in Figure 3.5.

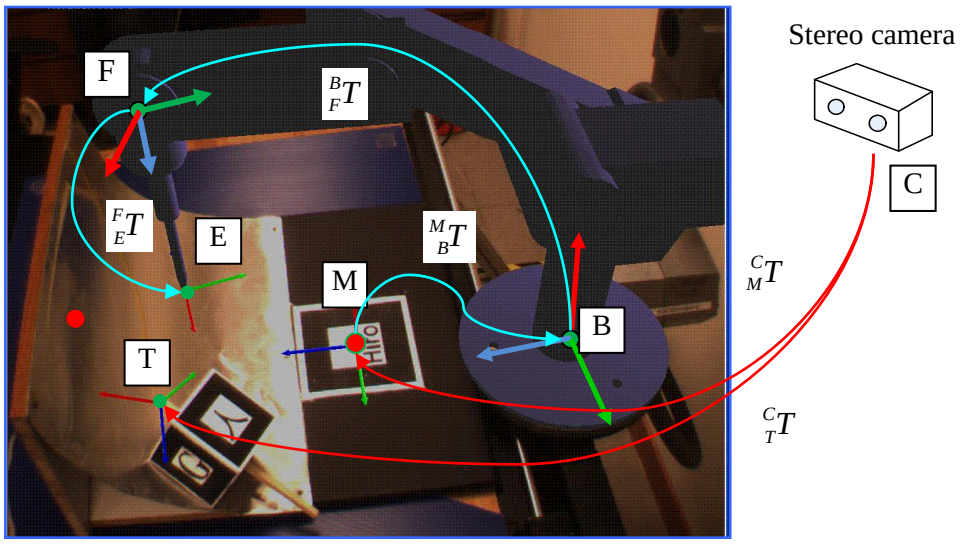


Figure 3.5: Coordinate frames and transformations in the RPAR-II system.

The coordinate frames are namely,

C: *Camera Frame*. This is the universal frame that is defined by the stereo camera for vision-based tracking and registration.

M: *Marker Frame.* This frame is defined at the centre of the marker. It is given by a transformation ${}^{C}_{M}T$, which represents the pose (position and orientation) of the centre of the marker with respect to the camera frame.

T: *Marker-cube Frame.* This frame is attached to the marker-cube and associated with the coordinate frames of four planar markers that are attached on the surfaces of the cube.

B: *Robot Base Frame.* This frame is attached to the base of the robot. The transformation ${}^{M}_{B}T$ describes the base frame with respect to the marker frame, and is used for virtual robot registration. In this research, the base of the robot is fixed to the workspace.

F: *Robot Flange Frame.* The flange frame is defined on the robot flange and described with respect to the robot base frame by the transformation ${}^{B}_{F}T$. It can be represented alternatively by a sequence of joint positions of the manipulator.

E: *Robot End-effector Frame*. The origin of this frame is located at the tool centre point of the EE. For a fixed robot base frame, a general robot movement can be specified by the movement of the frame defined at the robot EE.

W: *Robot Path Frame*. This frame is defined at the path or path segment followed by the EE. It is described by a transformation with respect to the robot base frame.

3.3.4 Virtual Robot Model Registration

In the RPAR-II system, a computer vision-based method is adopted to register a virtual robot over the real robot, as shown in Figure 3.5. The camera is maintained at a fixed position and orientation with respect to the workspace, such that robot registration is only required to be performed once using a fiducial marker. The transformation matrix between the camera and the base of the robot can be stored, and retrieved when the planar marker is removed from the workspace. In this research, a stereo camera is used for tracking and registration. Camera calibration

is carried out to obtain the intrinsic parameters of the camera before it is used in the system. The camera is located in a fixed position with respect to the base of the robot. To enhance the visualization of the complete robot programming process, the camera is placed such that it could cover the operating space of the robot, or at least the required workspace for a given task.

3.3.5 Robot 2-D Workspace

The workspace of a robot is defined as a volume of space which the robot can reach in at least one orientation. It can be represented as a cluster of 3D points in the Cartesian space. There are many research studies on robot workspace modeling and analysis. In this system, however, it is not necessary to calculate the entire workspace of the robot and augment it onto the real scene, as the movement of the robot is related to the work place. For a given z (referenced with the robot base coordinate system), the workspace is reduced to a 2D region reachable by the EE of the robot. This region can be characterized by two variables, namely, the position (represented by z) and the radius (R) of the region, as shown in Figure 3.6, which depicts the boundary (dashed curve in red) of the workspace of a robot from the side view. Considering the operating range of each joint and the length of each link, for a feasible z, the corresponding 2D workspace can be determined as follows:

(1) Locate the given z and determine the region it falls into;

(2) If $z \in [Z_1, Z_{Max}]$ (i.e., within region I), the radius of the 2D workspace is determined by θ_2 (In this case, $\theta_3 = 0$);

(3) If $z \in [Z_{Min}, Z_1)$ (i.e., within region II), the radius is determined as θ_3 (In this case, θ_2 is chosen such that the joint 2 is at the boundary of its operating range);

(4) Determine the angle of the fan-shaped region according to the robot base rotation range;

(5) Register the fan-shaped region onto the real working environment.

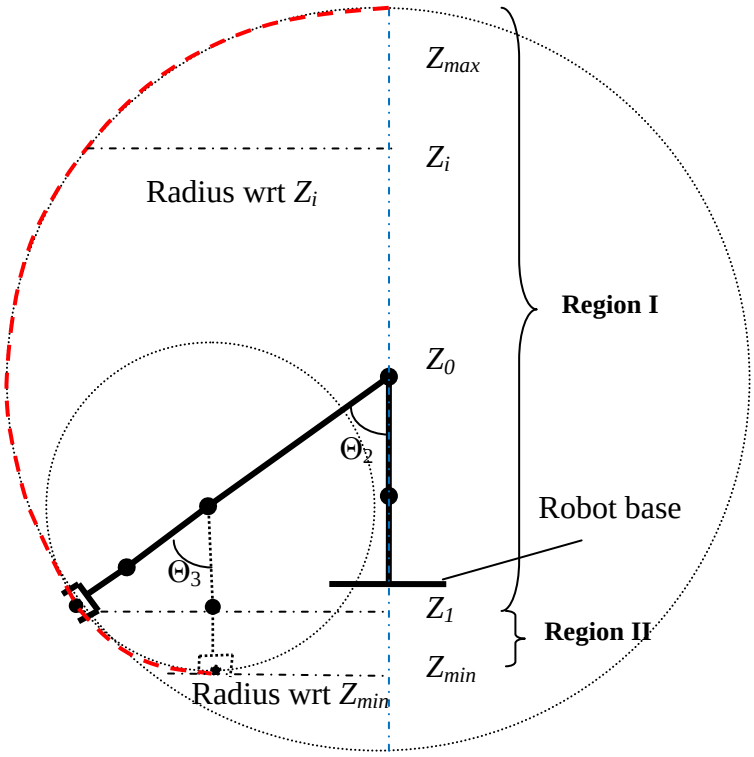


Figure 3.6: Determining the radius of the 2D workspace given its position (in z).

Figure 3.7 shows the registration of a 2D workspace (semi-transparent) onto the real scene. The fan-shaped region enables the user to visualize whether the starting and goal positions of the workpiece are in the reachable range of the EE.

If the goal position is adjustable, the virtual 2D workspace can assist the user in adjusting the goal position such that it is accessible by the EE.

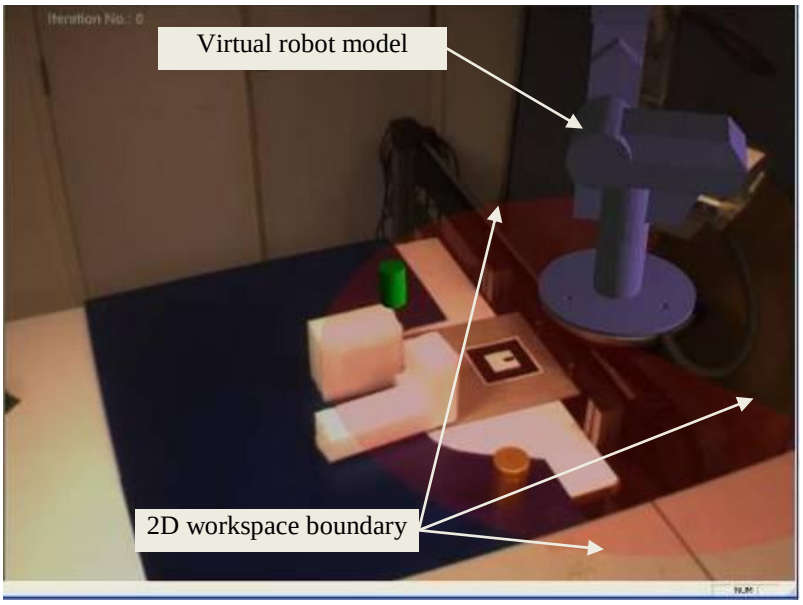


Figure 3.7: The fan-shaped region is the 2D workspace of the robot at the ground level ($z = 0$ referenced in the coordinate system given by the planar marker)

3.4 Summary

Existing robot path planning systems using AR have been developed mainly considering robot kinematics; AR-based simulations are mainly conducted to determine whether the planned path(s) are reachable and collision-free without considering the motion constraints. The system proposed in this research has been developed for interactive robot trajectory planning and simulation incorporating both robot kinematics and dynamics based on the application of AR. In this chapter, an overall system structure has been presented. An AR environment to facilitate HRI has been introduced. The tracking and registration methods to

register virtual objects and interaction techniques used to manipulate the virtual robot model are presented. Despite the limited accuracy in marker-based tracking methods, the users can teach the EE of the industrial robot easily using an interaction device in pick-and-place applications and continuous path following applications. A virtual 2D robot workspace can be rendered in the real working scene to enhance the users' view in the proposed AR environment.

Chapter 4. Interactive Path Planning and Simulation using Augmented Reality

4.1 Introduction

This chapter presents an interactive robot trajectory planning and simulation methodology using AR. The target applications are robot point-to-point tasks, such as material handling, spot welding, etc., where a collision-free and continuous path is required, given the starting and the ending points of the path. The roles of the users are (1) generating the free space relevant to the task, i.e., the Collision-Free Volume (CFV); (2) selecting and creating the control points; and (3) modify the control points where necessary, which includes insertion of new points and deletion of unwanted points. An approximated optimal trajectory profile can be obtained using the convex optimization approach to simulate the virtual robot with a proper control scheme.

4.2 RPAR-II Methodology for Path Planning

Most of the tasks carried out by an industrial robot can be decomposed into one or a combination of (1) pick-and-place operations and (2) path following operations. The focus of this section is on the planning of pick-and-place operations, and the goal is to generate collision-free paths and time-scale trajectories that can be transferred directly into the robot controller codes, taking into account robot

dynamics constraints and user-defined/task-dependent requirements. The proposed approach allows human participation to find the geometric paths that are collision-free and without singularities. In this research, a user demonstrates a given task through guiding and moving a virtual robot from the start point to the goal point. The data sampled and recorded during demonstration will be used to construct a CFV, which is a sub-set of the robot operation space. Instead of searching directly for a collision-free path between the pre-defined start point and goal point (Chong *et al.* 2009), the proposed approach allows the user to create a series of control points interactively within the CFV to form a path. There are two reasons for doing this. First, the search process, which is conducted in the joint space, cannot guarantee that the resulting path will be smooth in the Cartesian space. Secondly, searching for a time-scale trajectory directly within the CFV subject to both robot kinematics and dynamics constraints is a computationally intensive and complex problem deemed difficult to solve and implement. The path is formed through interpolation using the created control points, and this process serves as the first step of the de-coupled approach for robot motion planning.

After mapping this path to the joint space, an optimal time-scale trajectory is obtained by solving the reformulated trajectory optimization problem subject to robot kinematics and dynamics constraints. Finally, the trajectory is used to simulate the virtual robot performing the trajectory in the real environment, for the user to visually evaluate the quality of the resulting trajectory, in terms of the offsets deviated from the planned trajectory, and the joint torques required to

produce the trajectory. If the results are unsatisfactory, the user can choose to repeat the simulation process with different choices of the controller parameters, or the trajectory optimization process with different optimization parameters. The final trajectory can be transferred into the controller codes which can be executed on a real robot. The implementation of the RPAR-II approach is illustrated in Figure 4.1**.**

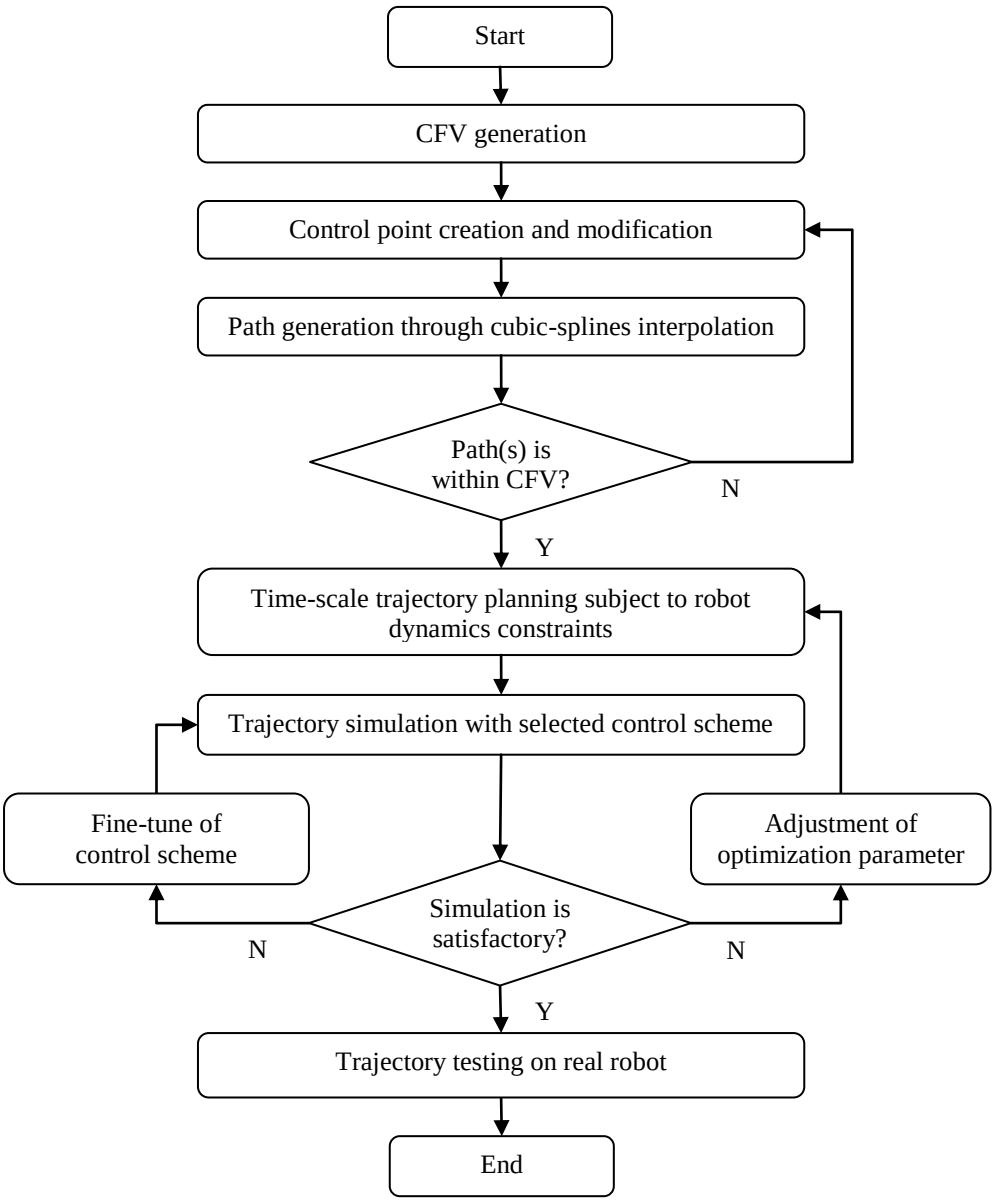


Figure 4.1: Trajectory planning using the RPAR-II system.

4.3 Human-virtual Robot Interaction

In this research, a marker-cube attached on a probe is adopted as the human-virtual robot interaction device, as described in Section 3.2.1. .The interaction device is used to facilitate the following operations, namely, CFV generation, control points creation and control point modification, which includes insertion of new control point(s) and/or deletion of unwanted control point(s).

4.3.1 Collision-Free Volume Generation

In the trajectory planning process, a virtual sphere with a pre-defined radius is attached to the marker-cube and a CFV can be generated by recording the position of the centre of the sphere through tracking the marker-cube. A CFV can be represented by a set of N_0 virtual spheres in Equation (4.1), where c_i is the centre of the sphere, r_0 is the pre-defined radius, and $S(c_i, X_i)$ is the Euclidean distance between c_i and X_i. The coordinate system of the base of the robot is used as the world coordinate system.

$$CFV = \bigcup_{i=1}^{N_0} \{ X_i : S(c_i, X_i) \le r_0 \} \tag{4.1}$$

4.3.2 Control Points Creation

The control points created in the Cartesian space should meet two requirements, namely, (1) they can be reached by the EE of the robot; and (2) the swept model at each control point should be within the CFV. In addition, the control points should

be selected at locations that are adjacent to the critical zone of the volume (i.e., with higher curvature) as shown in Figure 4.2. This increases the chance that the final interpolation between the created points is within the CFV. It can be seen that the control points selection option in Figure 4.2(b) is better than that in Figure 4.2(a).

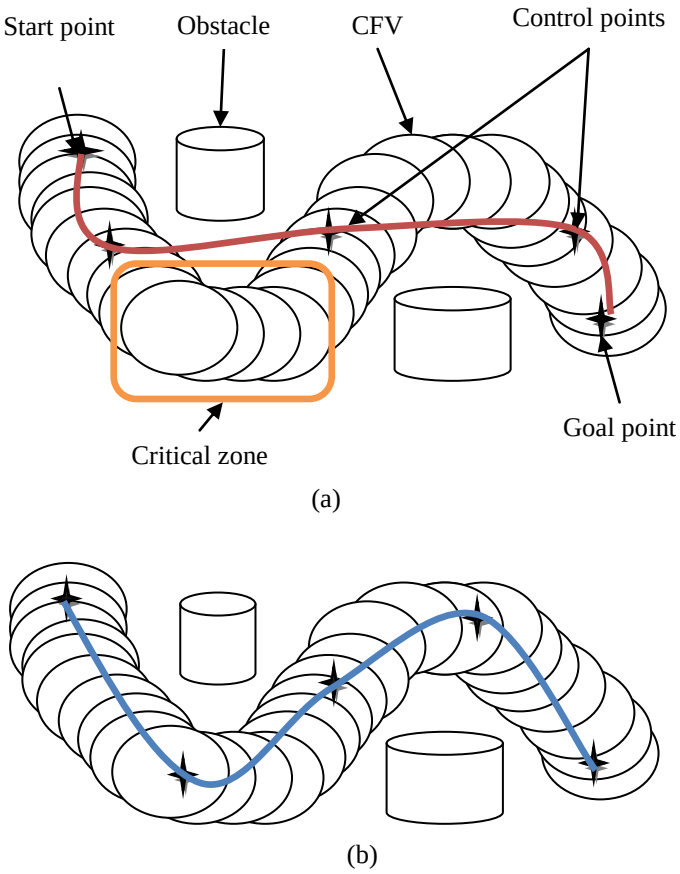


Figure 4.2: Selection of control points within the CFV during path planning.

It is assumed that the virtual object is rigidly attached to the gripper. A swept model enclosing both the EE and the virtual object is used for the CFV check, determines whether the swept model is within the CFV and hence collision-free. A bounding cylinder is used as a swept model in the proposed system. A simple method is used for the CFV check, in which a number of characteristic points on the discretized

surface of the bounding cylinder are verified to check whether they are within the CFV. In the Cartesian space, a cylinder (*BC*) can be defined by four parameters, namely, the radius (*r*), height (*h*), origin (*o*) and axis (*z*) of the cylinder, as given in Equation (4.2).

$$BC = BC\left(o, z, r, h\right) \in \Re^3 \tag{4.2}$$

The bounding cylinders corresponding to the created control points can be defined by Equation (4.3), in which *r* and *h* are the radius and height which are constant for a given bounding cylinder; o_j and z_j are the origin and axis of the cylinder corresponding to the *j*th control point; N_p is the number of control points; and t_j indicates the sequence of the control points to be created.

$$BC = \left\{ BC_j \left(o_j, z_j, r, h, t_j \right) \in \Re^3;\ j = 1, 2, ..., N_p \right\} \tag{4.3}$$

In the proposed system, the axis of the cylinder is the rotational axis of the gripper. The origin of the cylinder is set to locate at somewhere on the segment connecting the origins of the coordinate frames of the gripper and the flange, e.g., 65 mm away from the origin of the coordinate system of the gripper (given that the length of the segment is 231 mm, see Table 6-1). The radius and height of the cylinder are task dependent and associated with the dimension of the object to be grasped, i.e., the bounding volume should be able to enclose the object and the gripper.

4.3.3 Control Point Modification

Once a set of control points has been created, a path that passes through all these points can be automatically generated through interpolation. Although the interpolation process is efficient and fast, it cannot guarantee that the swept model of the EE is always within the CFV along the generated path. If the generated path collides with the CFV, the control point(s) can be modified to re-generate a new path. In practice, it is not necessary to re-generate a CFV and re-create all the control points, as the quality of a CFV is task-independent and affected by the jitters and variations in human demonstration as well as the frame rate achievable in the proposed system (Chong *et al.* 2007). There are other occasions that require control point modification, e.g., deletion of an inadequate point which has been created accidentally by the user, or the insertion of a new control point as in the case illustrated in Figure 4.2(a).

During the control point modification process, it may be difficult for a user to locate a control point in the workspace using the probe attached with a marker-cube. Therefore, a Euclidean distance-based approach is proposed to assist the user in selecting the point of interest in control point deletion or insertion operations. This method computes the distances between the probe and each control point, and associates this value with the corresponding control point. The distances computed are updated automatically when the probe moves in the workspace during the control point modification process. At each time instance, the control point that has the minimum distance to the probe is highlighted to the user as a candidate point that can be selected.

Define the origin of the coordinate system of the interaction device (tip of the probe) as $O_0(x_0, y_0, z_0)$, the *i*th control point as $V_i(x_i, y_i, z_i)$. and the Euclidean distance between O_0 and V_i, $S(O_0, V_i)$, for $i = 1, 2, ..., N_p$, where N_p refers to the number of control points with respect to the world coordinate system. Equation (4.4) gives the definition of a point of interest ($v_{poi}(x, y, z)$ to be selected.

$$v_{poi} : S(v_{poi}, O_0) = \min\{ S(V_k, O_0); k = 0, 1, 2, ..., N_p \} \tag{4.4}$$

The procedures for modification of control points, which includes deletion of a control point V_j and insertion of a point between the control points V_j and V_{j+1}, are as illustrated in Figure 4.3 and Figure 4.4 respectively.

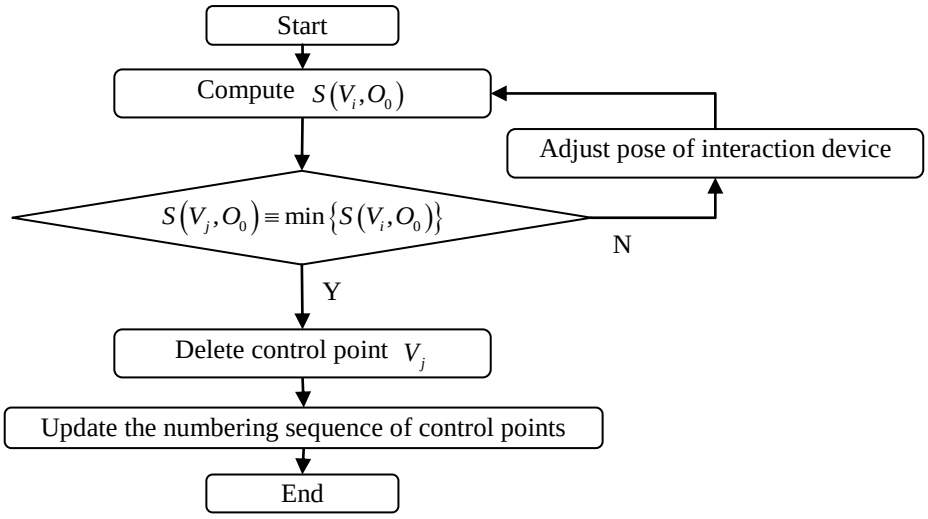


Figure 4.3: Control point deletion procedure.

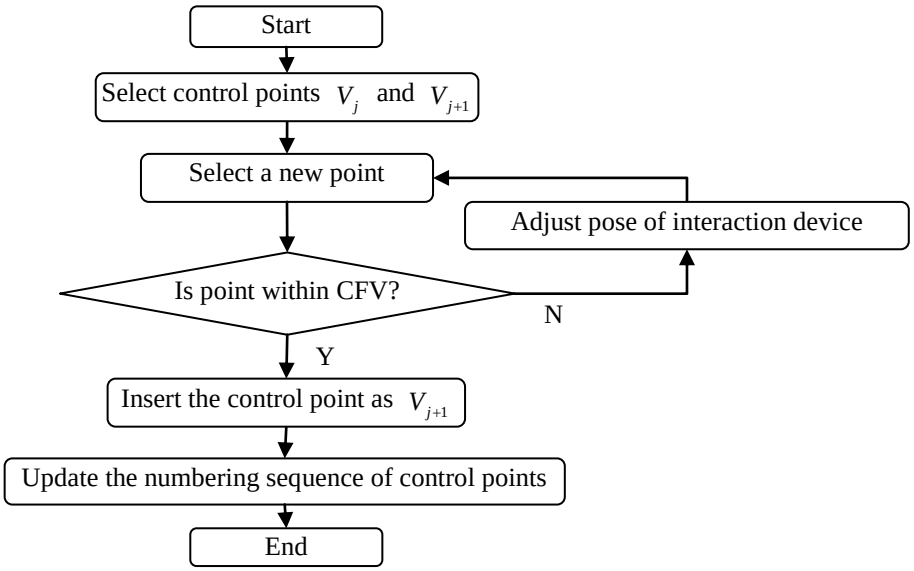


Figure 4.4: Control point insertion procedure.

4.4 Path Interpolation

One of the objectives for path planning in the Cartesian space is to minimize the path length. In this research, given a set of control points, the path can be described in a piecewise linear form where the control points are the knot-points. However, the motion of the EE of the robot following such a path will be jagged at these points. Therefore, a cubic-spline interpolation is adopted to generate a piecewise polynomial path that is second-order continuous at the control points (Craig 2005). The generation of a cubic-spline curve from a list of control points is presented in Appendix A. Particularly, a set of cubic splines are formed with "clamped" conditions, which means the first-order derivatives of the splines at the start and goal points, i.e., the two end tangent vectors, are constrained, such that the motion

of the robot could start and end with a user-defined velocity. In the proposed system, the first-order derivatives at the two end points are set to (0, 0, 0).

The path interpolated in the Cartesian space can be mapped to the joint space through inverse kinematics techniques (Craig 2005). As the control points are created through guiding a full-scaled virtual robot in the robot's operation space, if all the cubic-spline segments are within the CFV, the resulting path can be reached by the EE of the robot without violating the joint limits. Nonetheless, two practical issues need to be considered, namely, the number of control points and the data spacing between the control points.

4.4.1 Number of Control Points

The number of control points is dependent on the task as well as the working environment, and affects the overall curvature of the resulting path. In the proposed approach, a small number of control points will produce a path where one or more segments are likely to be outside the CFV. This problem can be solved by inserting one or more new control points. However, the number of control points should not be more than necessary. There are two reasons. Firstly, the path interpolated will not become smoother and/or with smaller curvature through increasing the number of control points. Secondly, due to the variations and uncertainties introduced during control point creation/modification and the level of accuracy achievable by the tracking system, increasing the number of control points means increasing the overall curvature of the path to be generated. In a less obstacle-clustered working environment, a feasible way for the user is to first select 5-10 control points to

perform the interpolation. If the generated path is not satisfactory, the user can insert new control points progressively. The modification of the control points, as shown in Figure 4.5, is largely dependent on the human operator's capability in the perception of the augmented environment. It is based on their spatial knowledge of the environment to determine whether there should be a deletion or insertion of a control point.

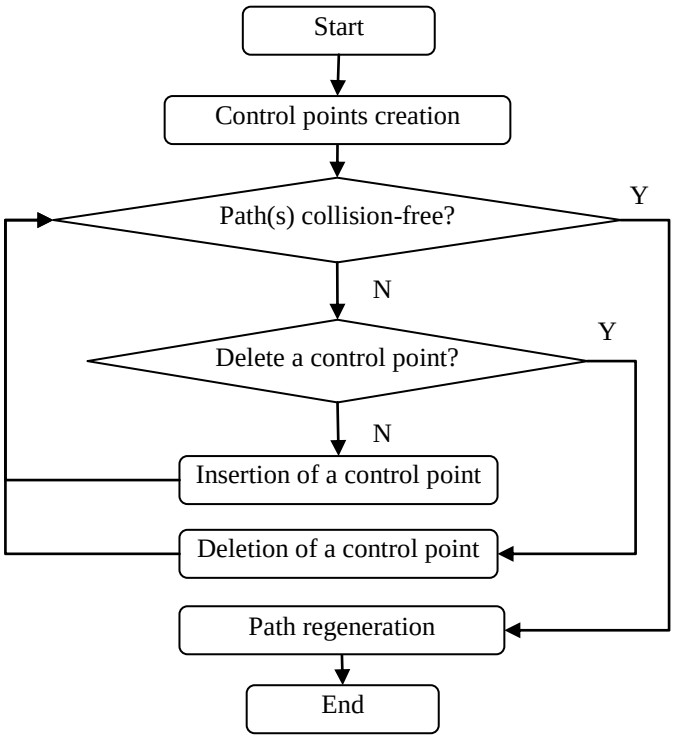


Figure 4.5: Control point insertion procedure.

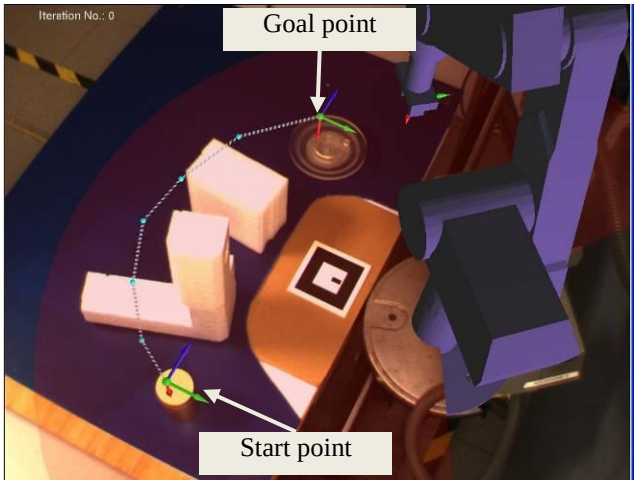
4.4.2 Time Stamps Attached to Control Points

A path is usually interpolated in the time domain. Hence, each control point should be assigned with a unique time stamp indicating the sequence/time that it will be reached. According to the creation procedure presented in Section 4.2.1.3, each of the control points is characterized by its sampling sequence. However, it

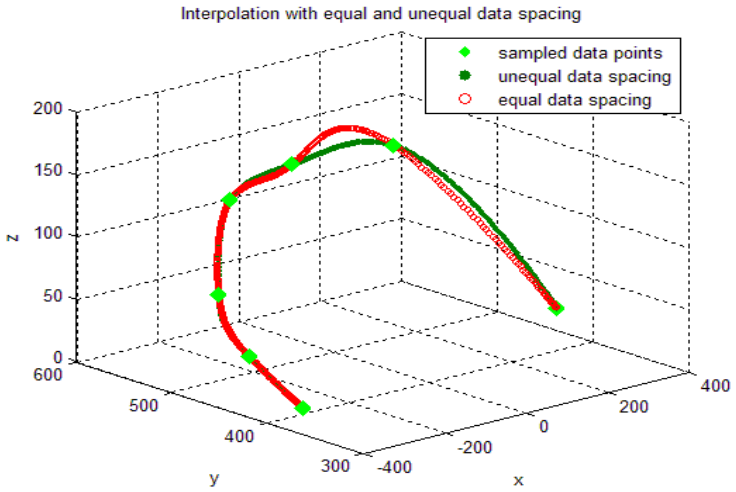
would not be practical to convert the sampling sequence directly into time stamps with equal data spacing for path interpolation. There are two reasons. Firstly, it is unlikely for a user to select a sequence of control points such that they are spaced evenly in the interpolated path. Secondly, it is not necessary to create the points with equal data spacing, as there should be more points in the obstacle-intensive areas and fewer in the less obstacle-crowded areas. Considering the Euclidean distance between two adjacent control points, the normalized time stamp assigned to each control point is proportional to the cumulative distance from the start point to this point. The time stamp for the *j*th control point is given in Equation (4.5), where d_i is the Euclidean distance between the *j*th and (*j-1*)th points, and *N* is the total number of control points (except start point and goal point, which are defined by $t_0 = 0$ and $t_{N+1} = 1$ respectively).

$$t_j = \frac{\sum_{i=1}^{j} d_i}{\sum_{i=1}^{N+1} d_i}, \; j = 1, 2, ..., N \tag{4.5}$$

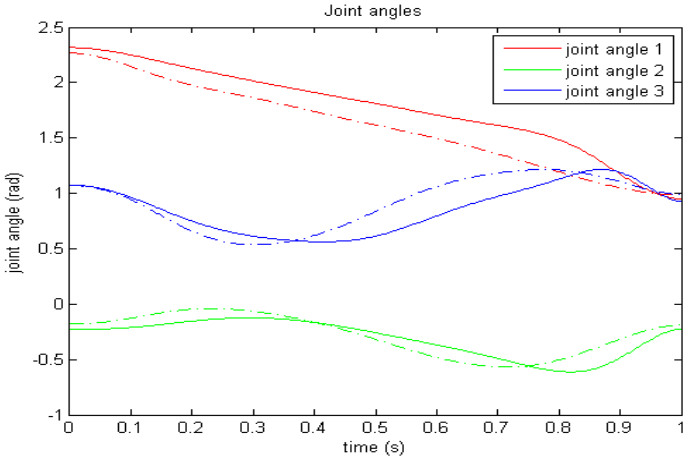
The plots in Figure 4.6 show the influence of the data spacing on the interpolation results. The dashed lines in Figure 4.6(c) represent the plots corresponding to the interpolation with unequal data spacing. It can be observed that the paths interpolated with unequal data spacing are smoother in the Cartesian space (Figure 4.6(b)) and have better distribution in terms of the curvature in the joint space (Figure 4.6(c)).



(a) Start point, goal point and control points in camera's view



(b) Plots of the paths in the Cartesian space



(c) Plots of the paths in the Joint space

Figure 4.6: Path interpolated with equal and unequal data spacings.

4.4.3 Collision Check

During the robot trajectory simulation, an important task is collision detection among all the virtual models. In the RPAR-II system, the *V-COLLIDE* package (V-COLLIDE; Hudson *et al*. 1997) has been adopted to perform the collision check between two virtual models in the *.STL* format. The procedure of collision detection can be divided into two steps, namely, robot self-collision detection; and detection between robot arms and other obstacles existing in the working environment.

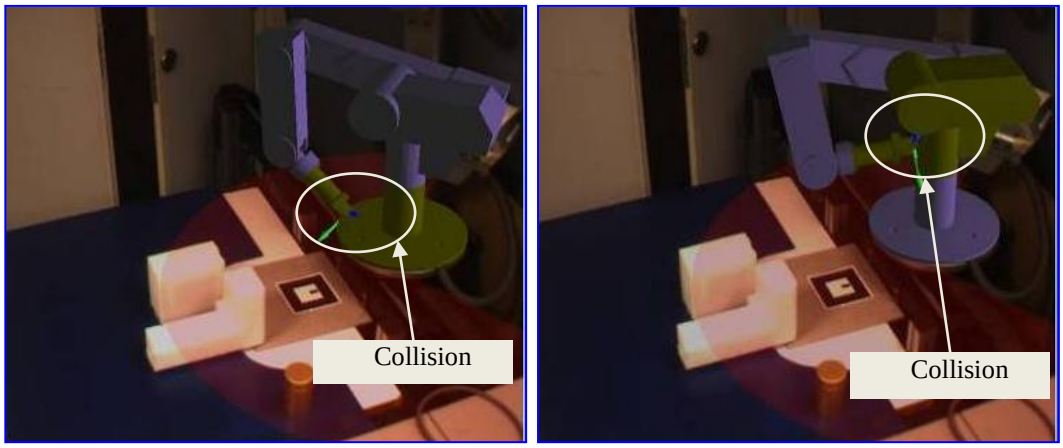
4.4.3.1 Robot Links Self-collision Check

The robot self-collision detection is to check whether the robot links collide with each other when the EE of the robot is moving along the simulated path. In this case, each link will be treated as an independent entity. Since the adjacent links are connected by one joint, this is not considered as a collision. The collision status is stored in a $(N+1)\times(N+1)$ matrix, where N represents the degree-of-freedom (DOF) of the robot. Figure 4.7 shows the robot arms self-collision detection results in two instances. The two links that collide with each other are rendered with different color.

4.4.3.2 Collision Detection between Robot EE and CFV

At each control point along the simulated path, in addition to the robot self-collision detection, the collision detection between robot and obstacles is required. Given the robot model and the obstacle model, detection can be carried out between the EE, other parts of the robot and the obstacles. If an obstacle is

static, only the 3D position of each link needs to be updated along the path for collision check. If the obstacle is mobile, a tracker will be required to track the movement of the obstacle. Such movements should be mapped to the virtual model (e.g., a bounding box) of the obstacle when the EE of the robot is moving along the path.



(a) EE of the robot collides with the robot base

(b) EE of the robot collides with the robot shoulder

Figure 4.7: Robot self-collision detection.

4.4.4 Trajectory Planning

Trajectory optimization in the joint space involves determining a sequence of joint angles that satisfies certain optimization criteria incorporating both robot kinematics and dynamics capabilities. In this research, the trajectory is optimized through solving a convex optimization problem (Verscheure *et al*. 2009), where the path duration is minimized subject to joint torque and joint velocity constraints. The constraints are enforced as log-barrier items to the convex optimization problem, allowing the robotic system to maximize its productivity without having to engage its actuators excessively.

For any *n*-DOF manipulator, the equation of motion with joint angles $q \in \Re^n$ is given as a function of the applied joint torques $\tau \in \Re^n$ (Craig 2005),

$$\tau(q,\dot{q},\ddot{q}) = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_s(\mathrm{q})\mathrm{sgn}(\dot{\mathrm{q}}) + G(\mathrm{q}) \tag{4.6}$$

where $M(q) \in \Re^{n\times n}$ is a positive definite mass matrix; $C(q,\dot{q}) \in \Re^{n\times n}$ denotes a matrix accounting for the Coriolis and centrifugal effects and is linear in the joint velocities; $F_s(\mathrm{q}) \in \Re^{n\times n}$ denotes a matrix of Coulomb friction torques, which is joint angle dependent; and $G(\mathrm{q})$ is the vector accounting for gravity and other joint angle dependent torques.

Let the trajectory sequence be represented in the form $q = \left[q(s_0),...,q(s_T)\right]^T$, which is given by a scalar path coordinate ***s*** that is usually interpreted as the arc length. Its first-order derivative and second-order derivative are given in Equation (4.7),

$$\begin{aligned} \dot{q}(s) &= q'(s)\dot{s} \\ \ddot{q}(s) &= q''(s)\dot{s}^2 + q'(s)\ddot{s} \end{aligned} \tag{4.7}$$

In Equation (4.7), $\dot{s} = ds/dt$, $\ddot{s} = d^2s/dt^2$, assume the trajectory starts at time 0 and ends at time *T*, the path coordinate ***s*** is parameterized as $0 = s_0 = s(0) < s(t) < s(T) = s_T = 1$, a strictly monotone representation

where $t \in (0, T)$; $\boldsymbol{s}$ is discretized into N_s segments evenly with $N_s + 1$ grid points. $\dot{s}$ and $\ddot{s}$ are the pseudo velocity and pseudo acceleration of the trajectory. Defining $a(s) = \ddot{s}$ and $b(s) = \dot{s}^2$ as two optimization variables, the time-optimal trajectory planning is to minimize the path duration *T* subject to the robot dynamics constraints, as being represented in Equation (4.8).

$$E_T = \min T = \min \int_0^1 \frac{1}{\dot{s}} \cdot ds \approx \min_b \sum_{i=0}^{N_s - 1} \left(\int_{s^i}^{s^{i+1}} \frac{1}{\sqrt{b(s)}} \cdot ds \right) \tag{4.8}$$

subject to:

$$\underline{\tau}(s) \le \tau(s) \le \bar{\tau}(s) \tag{4.9}$$

$$b(s) \ge 0 \tag{4.10}$$

$$0 \le s \le 1 \tag{4.11}$$

$$b'(s) = 2a(s) \tag{4.12}$$

$$b(0) = \dot{s}_0 \tag{4.13}$$

$$b(1) = \dot{s}_T \tag{4.14}$$

where

$$\tau(s) = m(s)a(s) + c(s)b(s) + g(s) \tag{4.15}$$

$$m(s) = M(q(s)) \cdot q'(s) \tag{4.16}$$

$$c(s) = M(q(s)) \cdot q''(s) + c(q(s), q'(s)) \cdot q'(s) \tag{4.17}$$

$$g(s) = F_s(q(s)) \cdot \mathrm{sgn}(q'(s)) + G(q(s)) \tag{4.18}$$

$m(s)$, $c(s)$ and $g(s)$ are evaluated from the equation of motion for a manipulator (Craig 2005). Based on the assumption that function $b(s)$, which is associated with the pseudo-velocity of the path, is at least piecewise linear at each segment, Equation (4.8) can be reformulated into a convex optimization problem given in Equation (4.19) (Verscheure *et al*. 2008), where $\Delta s^i = s^{i+1} - s^i = 1/N_s$.

$$E_T = \min_b \sum_{i=0}^{N_s-1} \left(\frac{2\Delta s^i}{\sqrt{b^{i+1}} + \sqrt{b^i}} \right) \tag{4.19}$$

4.4.4.1 Log-barrier Method

In robot trajectory planning, a dominant constraint in robot dynamics is the joint torque limits, i.e., the joint torque that produces the planned path should not exceed the actuator capability. Chan and Dubey (1995) proposed a scheme to avoid the joint limits during trajectory planning. By substituting the joint limits and the planned joint angle with the joint torque limits and the planned joint torques respectively, a penalty function ***P*** to avoid the joint torque limits is given by an averaging sum-log function in Equation (4.20), where $\bar{\tau}_j$ and $\underline{\tau}_j$ represent the upper and lower bounds of joint torque, *ndof* is the DOF of the robot.

$$P = \frac{1}{2*ndof} \sum_{j=1}^{ndof} \left(\log \left(\frac{\left(\bar{\tau}_j - \underline{\tau}_j\right)^2}{4 \times \left(\bar{\tau}_j - \tau_j\right) \times \left(\tau_j - \underline{\tau}_j\right)} \right) \right) \tag{4.20}$$

This function gives higher penalty automatically when the computed torque is close to the torque bounds and the function value goes to infinity when the torque limits

are violated. If the joint torque limits are symmetric, i.e., $\bar{\tau}_j = -\underline{\tau}_j$, Equation (4.20) can be simplified as Equation (4.21).

$$P = \frac{1}{2*ndof}\sum_{j=1}^{ndof}\left(\log\left(\frac{\left(\bar{\tau}_j\right)^2}{\left(\bar{\tau}_j^2-\tau_j^2\right)}\right)\right) \tag{4.21}$$

The joint velocity specifications incorporated in trajectory planning are dependent on the velocity limits of the task and the type of robot, and a lower velocity will be chosen. Suppose their lower and upper limits are symmetrical, i.e., $\bar{\dot{q}} = -\underline{\dot{q}}$, the joint velocity constraints can be transformed into a standardized inequality form in Equation (4.22), which defines a convex set.

$$b(s) \le \frac{\left(\bar{\dot{q}}(s)\right)^2}{\left(q'(s)\right)^2} \tag{4.22}$$

Similar to the treatment applied to the joint torque constraints, the joint velocity constraints can be enforced as another penalty function P^{v} in Equation (4.23).

$$P^{v} = \frac{1}{2\cdot ndof}\sum_{j=1}^{ndof}\left(\log\left(\frac{\left(\bar{\dot{q}}_j\right)^2}{\left(\bar{\dot{q}}_j\right)^2-\left(q'_j\right)^2 b(s)}\right)\right) \tag{4.23}$$

By augmenting Equation (4.19) with the penalty functions in Equation (4.21) and Equation (4.23), which is known as the log-barrier item, the time-optimal trajectory

planning problem is approximated as an unconstrained optimization problem (Boyd and Vandenberghe 2004). The formulation is given in Equation (4.24), where P_i is the penalty, evaluated at the middle point of segment $\left[s^i, s^{i+1}\right]$; κ is the log-barrier parameter.

$$E_{T-\Gamma} = \min \sum_{i=0}^{N_s-1}\left(\frac{2\Delta s^i}{\sqrt{b^{i+1}}+\sqrt{b^i}} + \frac{\kappa}{N_s} P_i \right) \tag{4.24}$$

This formulation is convex, which means it can be solved efficiently with a user-defined threshold within a number of iterations that is bounded by a polynomial of the problem dimensions (i.e., the number of objective and constrained functions). In this research, the solution of the formulation is obtained through solving first-order optimality conditions of this equation. The log-barrier parameter κ is interpreted as the maximum gap between the solutions to the log-barrier approximated problem given in Equation (4.24) and the time-optimal trajectory planning problem given in Equation (4.19). With a small value of parameter κ (such as $\kappa = 0.01$), the log-barrier item has negligible influence on the solution to the problem in Equation (4.19). In this case, the resulting joint torque will lead to "bang-bang" actuator behavior, i.e., at any time instant, at least one of the actuators is saturated. On the contrary, a larger κ will produce a sub-optimal yet smoother solution at the expense of moderately longer path duration. This is evident from the fact that the path duration is not proportional to the pseudo-velocity ($t = \sum_{i=0}^{N-1}\left(2\Delta s^i \Big/ \left(\sqrt{b^i(s)}+\sqrt{b^{i+1}(s)}\right)\right)$). With the aforementioned

property, the log-barrier parameter can be used to tune the optimization process such that it yields a more practical solution to implement, in the sense of less aggressive usage of the actuators, on a real robot manipulator.

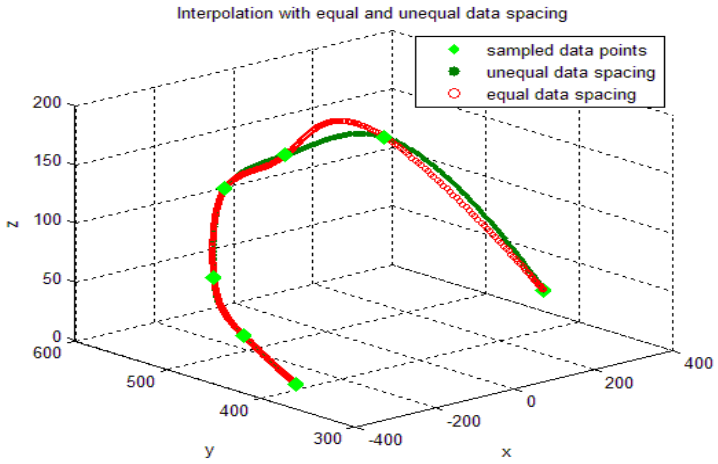
In the proposed system, the log-barrier parameter is not unique. The choice of the parameters can be logarithmically spaced, e.g., 0.01, 0.04, 0.158, 0.631 and 2.51, and each corresponds to a sub-optimal solution to the problem given in Equation (4.24). In many robotic tasks, such as in a robotic assembly line, there may be requirements imposed on path duration, e.g., it should be within a certain range. As the objective of trajectory optimization (as given in Equation (4.8)), the path duration can be used to determine a suitable log-barrier parameter through an iterative process, among some candidate log-barrier parameters.

To solve the unconstrained optimization problem given in Equation (4.24), the optimization variables need to be initialized to fulfill the constraints in Equations (4.9–14). Since **b** is at least twice differentiable and is known at the starting and end points, $\mathbf{b}^0$ can be defined as a parabola with a scalar as in Equation (4.25), given that $\dot{s}_0$ and $\dot{s}_T$ are the pseudo-velocities at the start and goal points. In principle, the pseudo-velocities at the two end points could be zero or non-zero constants according to the task requirements associated with these two points. In the proposed system, they are predefined as zero, indicating that the path velocities at these two points are zero. In Equation (4.25), the coefficient *c* is first chosen to be a positive value such that $\dot{s}_T - \dot{s}_0 - c < 0$, and it is then progressively decreased (while keeping $\dot{s}_T - \dot{s}_0 - c < 0$) until the joint torques constraints are satisfied.

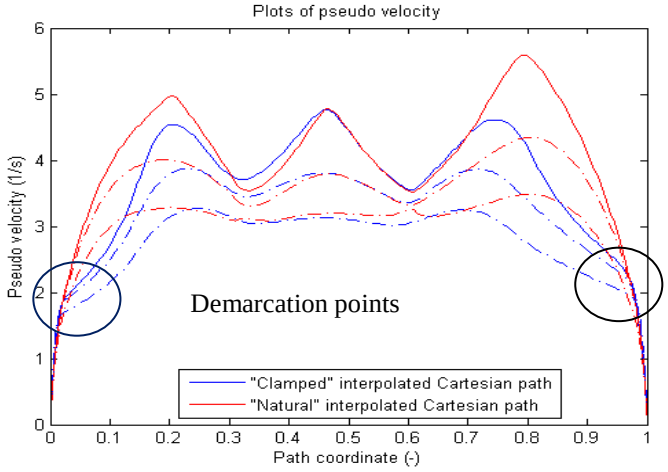
$$\mathbf{b}^0 = \left(\dot{s}_T - \dot{s}_0 - c\right)s^2 + c \cdot s + \dot{s}_0 \tag{4.25}$$

4.4.4.2 Effects of Path Interpolations Schemes on Trajectory Planning

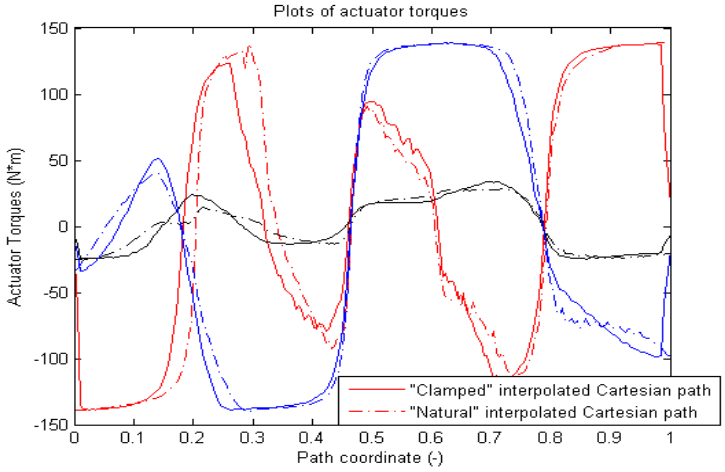
Different path interpolation schemes in the Cartesian space have effects on the solutions to the trajectory optimization, as shown in Figure 4.8. Natural spline interpolation (second derivative at end points are zero) and clamped spline interpolation (first derivative at end points are zero) are used to interpolate the path in the Cartesian space. Figure 4.8(a) shows that the plot of the path from the natural spline interpolation is almost the same as the one from the clamped spline interpolation. However, Figures 4.8(b) and (c) show the differences between the solutions to the trajectory optimization problem between these two interpolation schemes. The optimized actuator torques for the path from natural interpolation show that the planned motion is started with one of actuators is almost saturated (dashed line in red, Figure 4.8(c)); while for the path from clamped interpolation, the optimized torques start around zero, and one of them is progressively (or rapidly) increased to the saturation status (torque limits) (the solid line in red, Figure 4.8(c)). The resulting behavior of the actuator could explain the existence of two demarcation points in the plots of pseudo velocity.



(a) Plots of interpolated paths in the Cartesian space



(b) Plots of optimized pseudo velocity (κ = 0.01 (solid line), 0.16, 0.64 (dashed lines))



(c) Plots of optimized actuator torques (κ = 0.01)

Figure 4.8: Different path interpolation schemes in the Cartesian space.

4.5 Trajectory Simulation

During the trajectory simulation process, the planned trajectory is rendered in the real work space (or on the real workpiece). Figure 4.9 shows a flowchart for the simulation of the planned path.

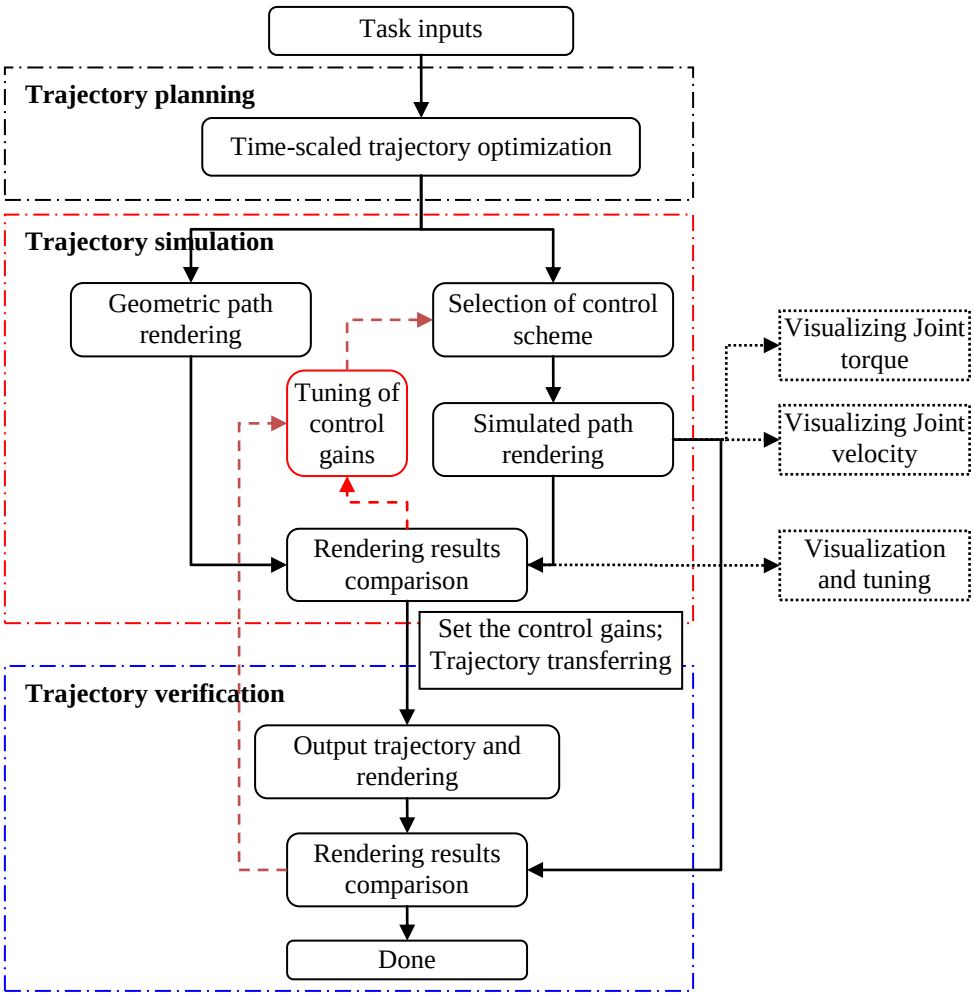


Figure 4.9: Flowchart for trajectory simulation and verification.

To simulate a planned path, a control scheme should be selected according to the capabilities of the robot controller. Generally for a given industrial robot, the

control scheme has been hardcoded in the controller, and the controller parameters can only be modified through the robot programming UI provided by the manufacturers. A set of estimated robot dynamics parameters of the three links of the Scorbot-ER VII robot, i.e., the robot shoulder, robot upper-arm and robot forearm, are considered for trajectory simulation. These parameters are the parameters that have been used during the trajectory optimization process.

4.5.1 Rendering of Trajectories

Once the time-scaled trajectory has been determined, it can be simulated with the virtual robot model by applying the control scheme adopted in the proposed system. The simulated trajectory can be visualized in two ways with different rendering schemes, namely,

(1) Rendered directly in the real working environment (Figure 4.10(a)), or

(2) Uploaded to the virtual robot and augmented gradually while the virtual robot simulates the planned motion (Figure 4.10(b)).

4.5.2 Optimization Constraints Check

In the proposed system, a PD control scheme is adopted to simulate the virtual robot performing this trajectory. A set of PD control gains has been defined initially and an interface is developed allowing the control gains to be adjusted manually. During simulation, the users can visually evaluate the trajectory as both the planned trajectory and the simulated trajectory are rendered on the working environment. Alternatively, the simulated trajectory can be rendered gradually as the EE of the virtual robot moves.

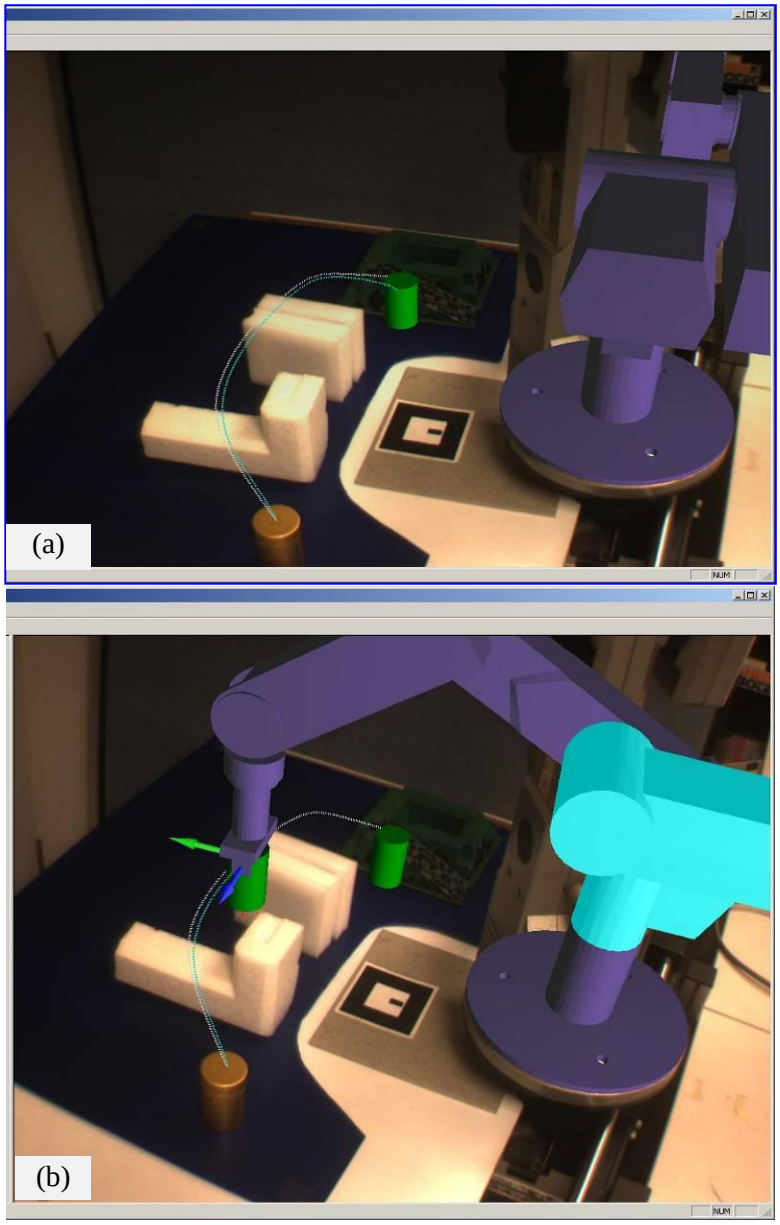


Figure 4.10: Two ways of rendering simulated trajectories in the real scenes.

During the simulation process, the normalized torque of each joint is computed and compared along the trajectory to check whether the joint torque constraints are violated. The normalized torque for joint *i* is defined as Equation (4.26), where

$\tau(i)$ is the actual joint torque of joint *i* computed according to the selected control gains, and $\tau_{MAX}(i)$ is the joint torque bound for joint *i*.

$$\tau_u(i) = \left| \frac{\tau(i)}{\tau_{MAX}(i)} \right| \tag{4.26}$$

The normalized joint torques is evaluated at each sample time during simulation. According to Equation (4.26), if $\tau(i) \in (0,1]$, which means the computed joint torque satisfies the joint torque constraints, the link which is associated with the joint with the largest $\tau(i)$ among all the joints is highlighted to indicate that it is the link most likely to deviate from the planned trajectory. Alternatively, if $\tau_u(i) > 1$, the simulation will be paused and the virtual robot will be halted in the current pose. The user can tune the control gain associated with joint *i*, and run the simulation again from the start point. In practice, due to the coupling effects in the robot dynamics model, it may be difficult to advise the users on the tuning of the control gains to obtain a desirable simulated path. However, there are some cues that can roughly indicate the parameter to be tuned. For instance, the planned trajectory is approximately time-optimal, which means the joint velocity is relatively high, hence the derivative control gains may affect the output of the control system the most. If the simulated trajectory is found to have shifted away parallel to the planned trajectory, then the proportional control gains need to be adjusted as these parameters dominate the drift of the output of the control system.

In general, a number of iterations are required to reduce the deviation between the planned and simulated trajectories to a predefined threshold.

4.5.3 Trajectory planning with tuned log-barrier parameter

A suitable set of control gains should be able to (1) reduce the deviation between the simulated trajectory and the reference trajectory, and (2) ensure the computed joint torques within the joint torques bounds while the virtual robot model moves along the simulated trajectory. In practice, these two conditions are in conflict with each other (i.e., larger control gain may be required for smaller trajectory deviation but may lead to violation of joint torques bounds). In this case the trajectory needs to be re-planned using a more conservative log-barrier parameter or an adjusted trajectory cost model. By doing this, the required torques for each joint will be reduced, though the path duration will be increased. After trajectory re-planning, the control gains need to be tuned again based on the procedure introduced in Section 4.5.2.

4.6 Summary

This chapter presented a methodology for interactive robot path planning and simulation based on the application of AR. The incorporation of robot dynamics and intuitive AR-based interface has facilitated the transformation of the planned trajectories into task-optimized executable robot paths. An interaction device attached with a marker-cube is used to assist the users in the guidance of the virtual robot in the workspace, as well as in the creation of a number of control

points within the surrounding environment, given a pair of starting and goal configurations. A Euclidean distance-based method has been developed to assist the users in the modification of control points, e.g., insertion and/or deletion of a control point, in the case that the geometric path generated from the available control points is not satisfactory. Two practical issues with regards to the path interpolation have been considered, namely, the number of control points and the data spacing among the control points. A log-barrier approximated optimization scheme is implemented to transfer the geometric path into a time-scale trajectory rapidly taking into account the robot actuators constraints. An AR-based interface for trajectory simulation has been developed to provide the necessary cues and options for tuning the controller parameters. In particular, both the control parameters and log-barrier parameter can be adjusted to achieve a more desired time-scale trajectory complying with the robot actuator capabilities. A case study of the approach proposed in this chapter will be presented in Chapter 6.

Chapter 5. Orientation Planning of Robot EE using Augmented Reality

5.1 Introduction

This chapter presents a methodology for planning the orientation of the EE for an industrial robot based on the application of AR. The targeted applications are those where the EE is constrained to follow a visible path with unknown position and model, at suitable inclination angles with respect to the path. The roles of the human are: (1) to demonstrate a visible path, where the sample data will be used to obtain a parameterized path model, (2) to select and modify the control points on the path model, and (3) to define the orientation of the EE associated with each control point so as to generate a ruled surface representing the path. An approximated time-optimal trajectory, which is determined subject to robot actuators and joint velocity constraints using convex optimization techniques, is implemented to simulate a virtual robot performing the planned motion, allowing the users to visually evaluate the trajectory planning process.

5.2 Ruled Surface

A ruled surface is a surface that can be swept out by moving a line in space. It has a parameterization of the form defined in Equation (5.1).

$$\mathbf{x}(u,v) = \mathbf{b}(u) + \upsilon\delta(v) \tag{5.1}$$

In Equation (5.1), $\mathbf{b}(u)$ is the ruled surface directrix (also called the base curve), and $\delta(v)$ is the director curve. The straight lines themselves are called rulings (Weisstein 1997).

The ruled surface has been used in robot path planning, which can represent the continuous motion of the robot configuration vector (Zha 2002), or the trajectory of linear tool (Tsai and Stone 2009), in Cartesian space. In these representations, the directrix $\mathbf{b}(u)$ can be defined by the prescribed positions passed by the tool center point (TCP), the rulings $\delta(v)$ is a normal vector, the orientation of the EE when travelling along the path, υ, which is normally a constant, is the length of the ruling, as illustrated in Figure 5.1. In this research, similar representation of ruled surface is adopted in orientation planning of robot EE based on the application of AR.

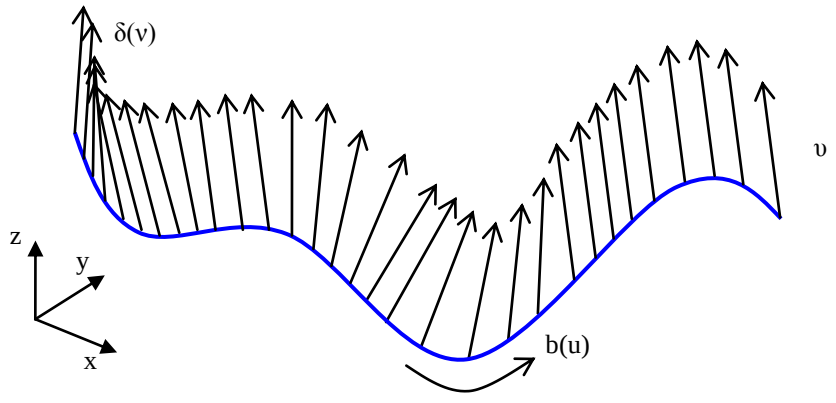


Figure 5.1: Ruled surface.

5.3 RPAR-II Methodology for EE Orientation Planning

The proposed approach for planning the orientation of the robot EE in a path following task is illustrated in Figure 5.2. The target applications are those where the EE is constrained to move along a visible 3D path on a curve surface, which position is unknown, within a pre-defined orientation range with respect to the coordinate systems defined on the curve. First, a user traces the spatial curve using an interaction device and records its positional data for curve learning (Ong *et al*. 2010). Second, a CFV, which is a sub-space of the robot operational space, is generated using the swept volume of a sphere attached to the interaction device. Third, the user selects a sequence of control points on the output curve, and specifies the orientation of the EE at each control point with respect to a universal coordinate frame, such as the robot base frame. A smooth path of the EE can be generated through orientation interpolation associated with the output curve. For an axis-symmetrical EE, the orientation profile can be depicted as a ruled surface, which takes the output curve as the directrix and the orientations of the EE as the rulings. After mapping this path to the joint space, a time-scale trajectory is determined through solving a reformulated trajectory optimization problem subject to the joint velocity constraints and the joint torque constraints. This trajectory is simulated on the real workpiece using the virtual robot so that the user can visually check the collisions that are likely to occur between the EE and the obstacles in the workplace, and evaluate the quality of the resulting trajectory. The final trajectory is compiled into controller codes for further verification on a real robot.

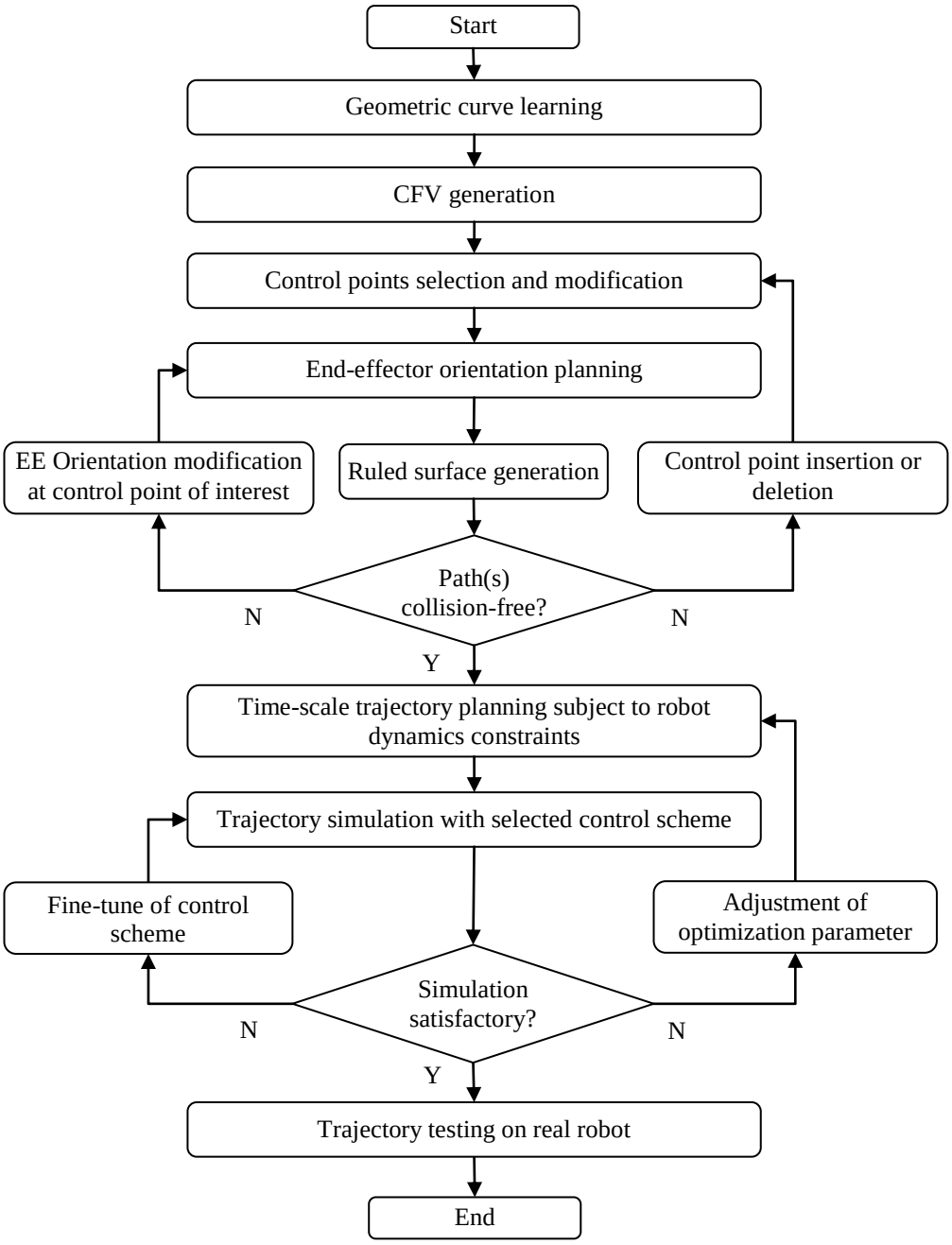


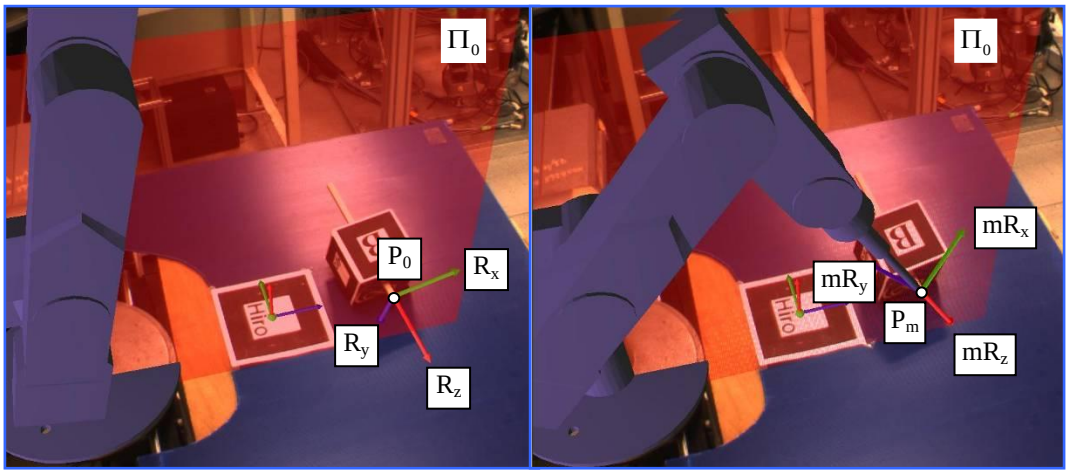
Figure 5.2: Procedure for EE orientation planning using the RPAR-II system.

5.3.1 Interaction Device

In this research, a marker-cube attached on a probe, which offers six degree-of-freedom (DOF) tracking capability, is adopted as an interaction device to

facilitate human-virtual robot interaction in robot path planning. It is used to guide the EE of a virtual robot of a Scorbot-ER VII type manipulator. This type of robot has a reduced wrist configuration, i.e., it lacks one DOF (yaw); therefore, a pose tracked via this interaction device should be carefully mapped to an alternative pose that permits valid inverse kinematic solutions.

It is assumed that the axis of the axis-symmetrical EE attached to the Scorbot-ER VII manipulator is aligned with the fifth axis of the robot. Given a position of the EE tracked by the marker-cube, the possible orientation of the EE with respect to the robot base frame ***B*** is co-planar (plane Π_0), as shown in Figure 5.3. This plane is also perpendicular to the X-Y plane of the robot base frame ***B***. The procedure for mapping an arbitrary pose tracked using the marker-cube to the pose of the EE is presented in Figure 5.4.



(a) An arbitrary pose given by the marker-cube

(b) Resulting pose after mapping and projection

Figure 5.3: Coordinate mapping from a tracked marker-cube.

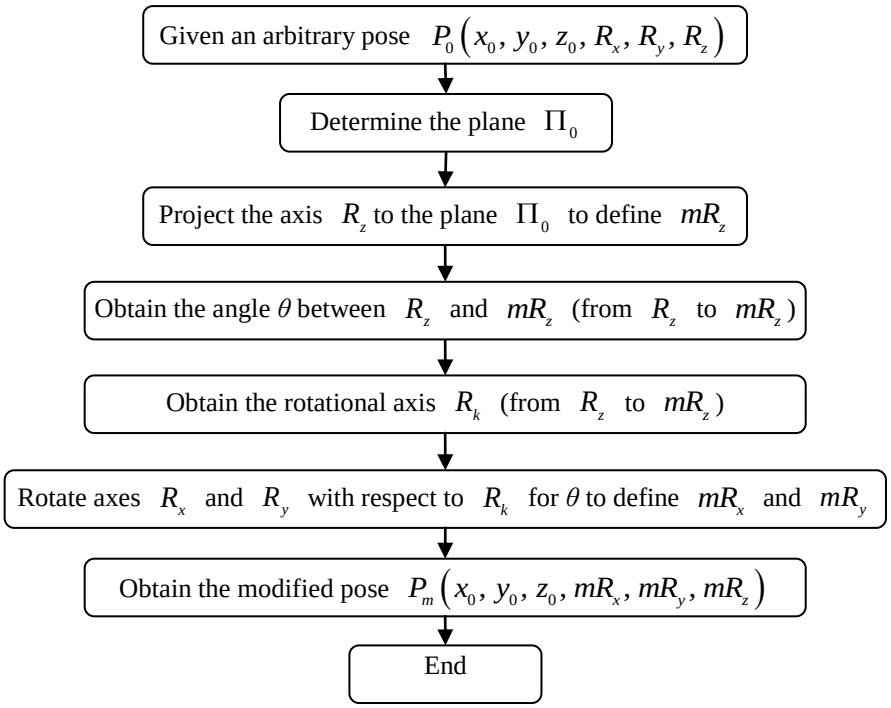


Figure 5.4: Coordinate mapping procedure.

In the path planning process, human-virtual robot interactions involve manipulating the virtual robot around the target path/curve to verify its accessibility, tracing the path for sample data collection, generating a CFV along the output curve, selecting the control points on the output curve and planning of orientations of the EE of the robot at these points. Two modes will be needed here. One is the creation mode where a list of control points is created and the orientations are specified for the robot EE at these points. The second mode is the modification mode where an undesirable control point can be deleted, a point can be inserted, or an orientation defined at any control point can be modified.

5.3.2 Output Curve Model

Given a set of sample data, which is recorded through tracking an interaction device based on human demonstrations in following a reference curve, a parametric curve can be obtained using the re-parameterization and multi-layer neural network techniques (Ong *et al*. 2010). The output curve can be represented by a sequence of parameterized points, as shown in Equation (5.2).

$$P = \left\{ P_i^L \left(i, a_i^L, p_i^L \right) \in \Re^3; i = 0,1,....., N_l \right\} \tag{5.2}$$

In Equation (5.2), i is the index of the parameterized points; a_i^L refers to the auxiliary index which is defined to be proportional to the cumulative distance from the start point to the *i*th point; $p_i^L = \left[x_i, y_i, z_i \right]^T$ is a vector defining the spatial position of the *i*th point with respect to the robot base frame; the total number of parameterized points to form the output curve is N_l+1. P_0^L and $P_{N_l}^L$ refer to the start and the end of the curve. The auxiliary index for each point on the output curve is given in Equation (5.3), where $d_j = \sqrt{\left(x_j - x_{j-1} \right)^2 + \left(y_j - y_{j-1} \right)^2 + \left(z_j - z_{j-1} \right)^2}$, which is the Euclidean distance between the *j*th and (*j-1*)th points.

$$a_i^L = \frac{\sum_{j=1}^{i} d_j}{\sum_{j=1}^{N_l} d_j}, i = 1,2,....., N_l \tag{5.3}$$

The output curve can be re-parameterized into an alternative sequence of points with even interval through interpolation, as given in Equation (5.4).

$$P = \left\{ P_j \left(t_j, a_j, p_j \right) \in \Re^3;\ j = 0,1,....., N_s \right\} \tag{5.4}$$

In Equation (5.4), N_s +1 is the number of the re-parameterized points. The step size for interpolation can be interpreted as the arc/path segment length measured between two adjacent sample points. Therefore, having $t_j = j$ and $D = \sum_{j=1}^{N_l} d_j$, as defined in Equation (5.3), the auxiliary index for the *j*th point in the re-parameterized sequence is defined as $a_j = t_j / N_s$. The re-parameterization is useful for trajectory optimization where the output curve will be re-defined in scalar path coordinate by a sequence of evenly discretized points. Figure 5.5 shows such an output curve model being rendered on the workpiece.

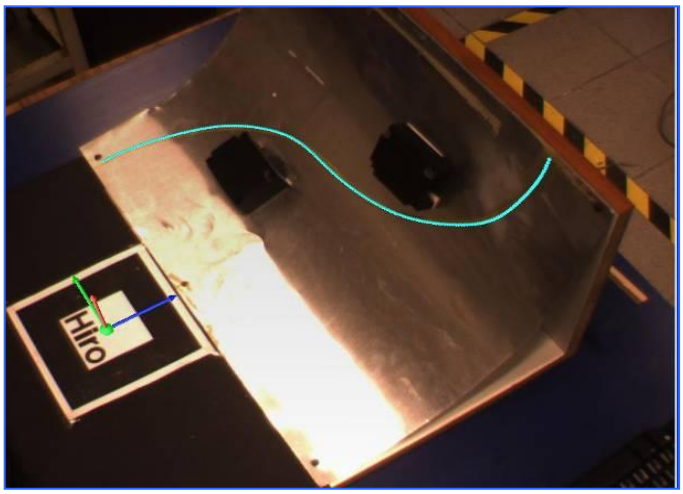


Figure 5.5: Output curve model.

5.3.3 Control Points Creation and Modification

In this section, the Euclidean-distance-based method, as developed in Section 4.3.4, is used to create and/or modify a set of control points. The distances are updated automatically when the marker-cube moves around the curve. At each time instance, the control point that has the minimum distance to the tip of the interaction device is highlighted, as illustrated in Figure 5.6. This enables the user to choose a point intuitively.

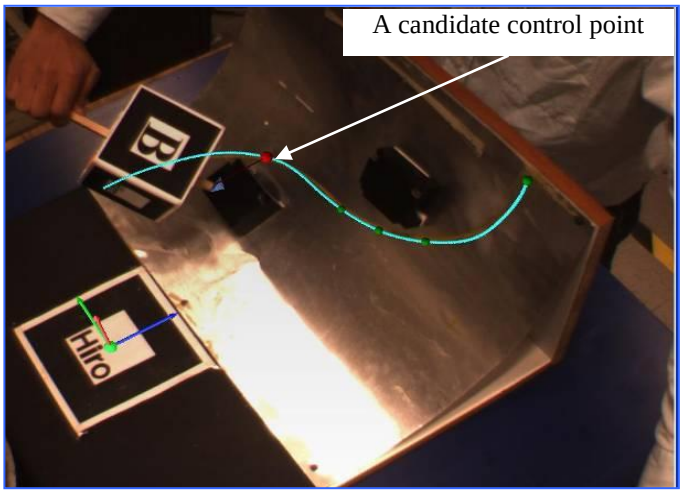
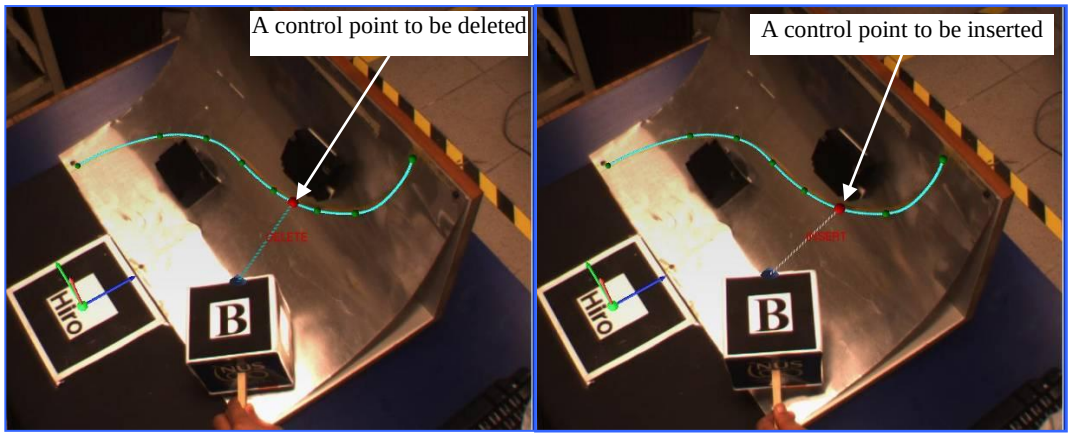


Figure 5.6: Selection of control point on the curve.

The list of control points can be modified in the modification mode. In this mode the Euclidean-distance-based method is applied to select the point of interest for modification, which can be a point from the control point list to be deleted (Figure 5.7(a), where $N_p = N_l$ in Equation (4.4)), or a point from the output curve to be inserted into the control point list (Figure 5.7(b), where $N_p = N_s$ in Equation (4.4)). These operations remove the need to generate a new list of control points when changes are needed.

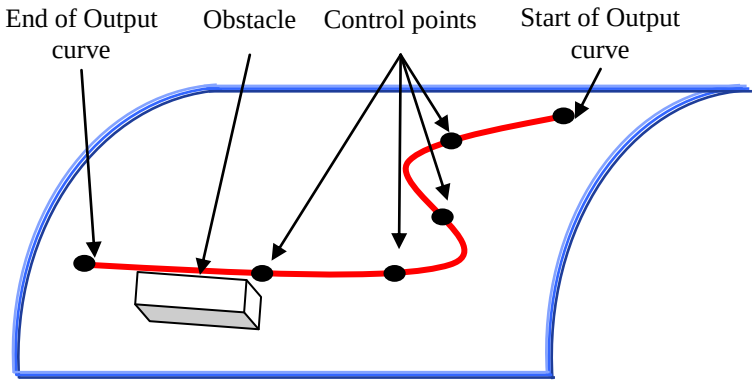


(a) Selection of a point to be deleted (b) Selection of a point to be inserted

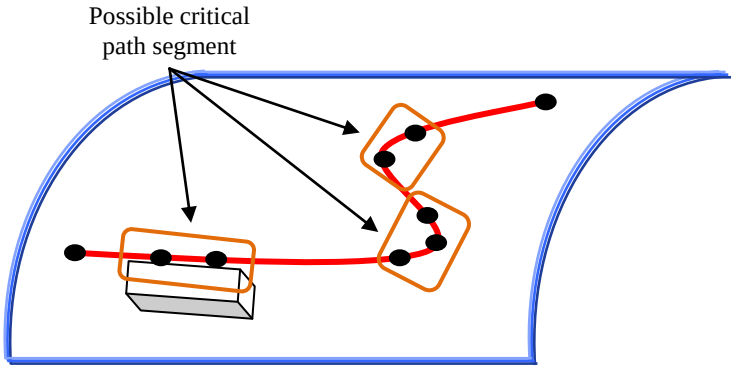
Figure 5.7: Euclidean-distance-based method for point selection in control point modification.

In a general case where a path lies in a curved surface, the location of the control point will affect the geometry and property of the orientation profile to be generated. It would be necessary to have control points at the critical path segment(s), i.e., with higher curvature or in the vicinity of obstacles, as illustrated in Figure 5.8, since these segments are locations where the EE orientations need to be planned more carefully. In addition, it is also undesirable to have an excessive number of control points. A list with more control points may result in a ruled surface being generated in a more controlled manner; however, it is also likely to produce overly constrained ruled surface. In the current system, the final set of control points is determined based on the human operator's understanding of the geometry of the path and the relevant working environment, such as the locations of the obstacles. Alternatively, since the curve model has been obtained, there are existing algorithms for adaptive sampling of curves based on curvature, e.g., the work by De Figueiredo (1995), which can be adopted in the proposed

system. With these initially determined control points, additional control points can be iteratively inserted in the path segments that are adjacent to the obstacles.



(a) Bad choice for control points



(b) Better choice for control points

Figure 5.8: Control points selection.

5.3.4 End-effector Orientation Planning

5.3.4.1 End-effector Orientation at a Control Point

The control points are used to define the orientations of the EE at these points and control the EE orientation profile along the path. Therefore, a data structure associated with each control point, as illustrated in Figure 5.9, is proposed. The data structure is given in Equation (5.5), where N_k is the total number of control points created. CP_1 and CP_{N_k} correspond to the start and end of the output

curve. The parameters defined in the data structure are listed in Table 5-1. Note that $\boldsymbol{\alpha}$ is pre-defined with respect to the normal of the surface where the curve lies, and all the parameters are measured with respect to the robot base frame.

$$CP = \left\{ CP_k \left(k, a_k, p_k, R_k, \alpha_k, o_k, \beta_k \right) \in \Re^6, \ k = 1, 2, ..., N_k \right\} \tag{5.5}$$

Table 5-1: Data structure of the control points

k	Index of control point
a	Auxiliary index of control point
p	Positional component of the coordinate frame defined on control point
R	Rotational component of the coordinate frame defined on control point
α	EE inclination range
o	Vector defines the orientation of the EE
β	Angle between the EE and the Z-axis of the robot base frame

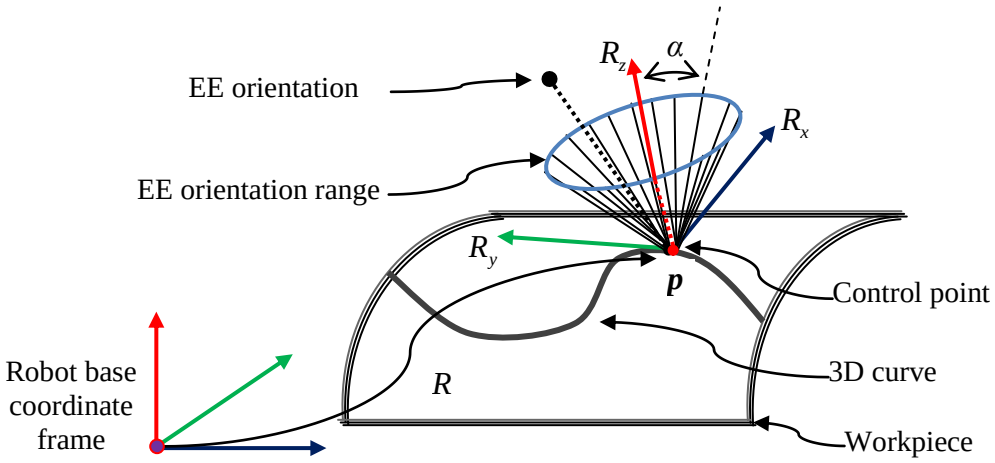


Figure 5.9: Parameters definition associated with a control point.

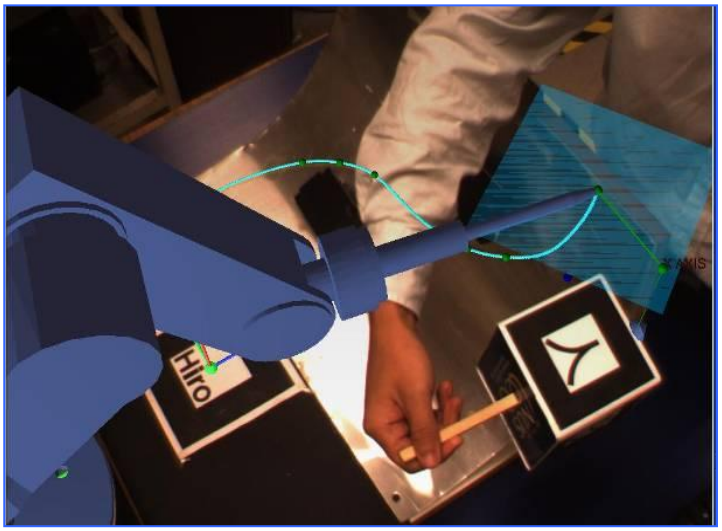
In the control point selection stage, the parameters $\boldsymbol{n}$, $\boldsymbol{a}$ and $\boldsymbol{p}$ can be determined for each point, and $\boldsymbol{\alpha}$, which is a task-dependent parameter, can be set accordingly.

The rest of the parameters can be initialized to be zero and specified later during the EE orientation planning stage.

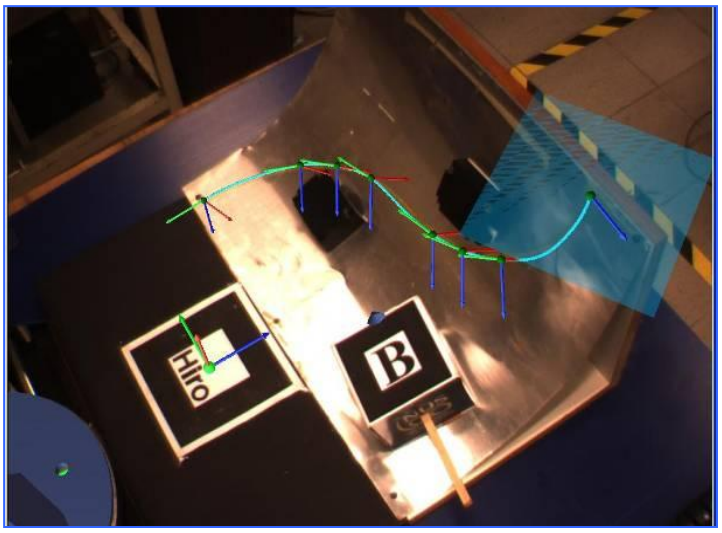
To specify an orientation for the robot EE approaching a control point, a coordinate frame has to be defined first at this point. The definition of a coordinate frame located at the start of the output curve, which is the first control point, has been presented (Ong *et al*. 2010). In this approach, one axis of the coordinate frame (e.g., Y-axis) is set as the direction of the output curve, a user-controlled axis (e.g., X-axis) is defined to project through the tip of the interaction device onto the normal surface perpendicular to the curve direction, and the third axis (e.g., Z-axis) is determined uniquely using the right-hand convention, as illustrated in Figure 5.10(a). The coordinate frames with origins at the rest of the control points can be defined accordingly by applying the transformations reflecting the changes in the curve direction, as shown in Figure 5.10(b), and the parameters $\boldsymbol{R}$ in Equation (5.5) for each control point are defined.

The orientations for the EE at the control points are defined sequentially in accordance with sequence of selection. It can be defined with respect to the coordinate frame at the control point and the robot base frame, and used to define the last two parameters, $\boldsymbol{o}$ and $\boldsymbol{\beta}$, in Equation (5.5). At any instance, the position of the EE is assigned as the position of the current control point while its orientation is controlled through the manipulation of the interaction device. The orientation specified should be accessible by the robot EE and within its inclination range as predefined per control point. In addition, the EE at this

orientation should be within the CFV generated. If an undesirable orientation has been defined, the particular control point associated with this orientation can be located by applying the Euclidean-distance-based method in the modification mode such that the orientation can be re-defined with the assistance of the interaction device.



(a) Coordinate frame definition at the start of the output curve



(b) Coordinate frames permutation to the rest of control points

Figure 5.10: Coordinate frames definition at the control points.

5.3.4.2 End-effector Orientation Interpolation

Once the orientation of the robot EE at each control point has been defined, a *ruled surface*, which takes the output curve as the directrix and the orientations of the EE as the rulings, can be generated through interpolation to represent the orientation profile planned along the curve. At any parameterized point from the output curve, an orientation of the EE can be represented with respect to the robot base frame by a unit vector. Based on the analysis in Section 5.3.1, the orientation of the EE should be in a plane that is uniquely determined using this particular point and the kinematics parameters of the robot model. By observing this property, the angle between the unit vector representing the orientation of the EE and the *Z*-axis of the robot base frame can be interpolated. As the parameterized points of the output curve are generated by taking their normalized accumulative arc/path lengths to the start of the curve as the interpolation parameter, the angles associated with these points should be generated with the same parameter. Thus, the orientations of the EE along the output curve can be represented in Equation (5.6). Consequently, a ruled surface which represents the orientation profile along the output curve is given in Equation (5.7).

$$W = \left\{ W_j\left(t_j, a_j, \beta_j\right) \in \Re^3;\ j = 0,1,....., N_s \right\} \tag{5.6}$$

$$G_{rs} = \left\{ P_j\left(t_j, a_j, p_j\right) + \nu \cdot W_j\left(t_j, a_j, \beta_j\right);\quad j = 0,1,....., N_s \right\} \tag{5.7}$$

In Equation (5.7), $P_j\left(t_j, a_j, p_j\right)$ defines the ruled surface *directrix*; $W_j\left(t_j, a_j, \beta_j\right)$ defines the direction of the *ruling*; ν is the length of the *ruling*,

which is proportional to the distance measured from the tip of the axis-symmetrical EE to the origin of the robot flange frame.

5.3.5 Trajectory Optimization

Trajectory optimization in the joint space involves determining a time history of the planned path that satisfies certain optimization criteria incorporating both robot kinematics and dynamics capabilities. As a dominant constraint, the joint torque that produces the optimized trajectory should comply with the actuator capability; meanwhile, the joint velocity specifications are dependent on the velocity limits of the task and the type of robot, and a lower velocity will be chosen. In this research, a trajectory which minimizes the path duration subject to inequality constraints associated with joint torque and joint velocity is determined through solving a convex optimization problem using the logarithmic barrier method (Boyd and Vandenberghe 2004). The objective function for time-optimal trajectory is represented using the parameters associated with a scalar path coordinate, and the penalty function for each type of inequality constraints are given by an averaging sum-log function, making the constraints implicit in the optimization objective.

5.3.6 Trajectory Simulation

The trajectory obtained can be simulated using a virtual robot under a discrete Proportional-Derivative (PD) control scheme. In the simulation, the joint torque and joint velocity can be inspected to determine whether they violate their limits. A normalized measurement associated with the simulated torque and velocity of each joint is computed and compared along the trajectory. It is defined as in Equation

(5.8), where $\tau_{sim}(i)$ and $\dot{q}_{sim}(i)$ are the simulated torque and velocity of joint *i* according to the selected control gains; $\bar{\tau}(i)$ and $\bar{\dot{q}}(i)$ are their upper bounds respectively.

$$m_u(i) = \max\left\{\left|\frac{\tau_{sim}(i)}{\bar{\tau}(i)}\right|, \left|\frac{\dot{q}_{sim}(i)}{\bar{\dot{q}}(i)}\right|\right\} \tag{5.8}$$

The normalized measurement will be evaluated at each sample time during simulation. According to Equation (5.8), if $m_u(i) \in (0,1)$, both the computed torque and velocity for each joint/link are within their limits. In this case, the link that has the largest torque and velocity among all the links is highlighted to indicate that it is the link most likely to deviate from the planned path. If $m_u(i) > 1$, the computed torque or velocity for joint *i* violates the constraints, the simulation will be paused with the link *i* highlighted. The user can adjust the control gain associated with this particular joint/link, and execute the simulation again. A general rule can be applied for rough tuning of the control gains based on the cues observed during simulation. For instance, in the proposed system, the trajectory achieved is approximately time-optimal; therefore, the derivative control gains may have the largest effect on the output of the control system. If the simulated trajectory is found to have shifted away from the planned trajectory, the proportional control gains will need to be tuned as these parameters dominate the drift of the output.

5.4 Summary

This chapter has presented a methodology to assist the users in planning trajectories and orientations of the EE for a robot along a visible path which position and model are unknown, complying with the robot dynamics constraints. An approach has been developed to map a tracked pose to an alternative pose of the EE to permit a valid robot configuration, considering the reduced wrist configuration of the robot model. Through manipulating the interaction device, a user is able to perform a series of operations, including acquiring a visible curve model, generating a CFV at the vicinity of the curve model, creating a list of control points on the curve, and specifying the orientations of the EE at these points, in order to achieve a smooth and collision-free ruled surface representing the orientation profile of the EE along the output curve. A data structure associated with each control point has been developed. A log-barrier approximated optimization method has been implemented to transfer the geometric path into a time-scale trajectory, subject to the joint torque and joint velocity constraints. The simulation enables the users to preview the optimization outputs, tune the log-barrier parameters and the control parameters to achieve a more desirable trajectory complying with the robot actuator capabilities. The case studies of the approach proposed in this chapter will be presented in Chapter 6.

Chapter 6. Case Studies and Discussions

6.1 Introduction

This chapter presents case studies of the proposed methodologies on path planning for a pick-and-place task, and orientation planning of the robot EE for a path following task. The RPAR-II system was implemented using C/C++ programming language under Visual C++ 2005 environment on a 1 GHz PC. Two external packages are used, namely, *Roboop* which provides robot kinematics and dynamics modeling, and *gnuplot* which provides various plot routines.

6.2 RPAR-II System Graphics User Interface

In the RPAR-II system, a Graphic User Interface (GUI) was developed to assist the users in robot task and path planning using a virtual robot in the real working environment. Two panels were implemented to perform different functionalities at different stages of the planning process, namely, one for virtual robot modeling and another for trajectory planning and simulation.

The panel for virtual robot modeling enables the users to create/modify the parametric models manually for different robot types by incorporating with the *Roboop* package, as shown in Figure 6.1. The right side of the panel is the elements of a data structure defining a single robot link (joint), including joint type (revolute,

prismatic, or fixed), D-H parameters, link mass, centre of mass at respective link, inertia tensor, and motor properties (including motor inertia, gear ratio, viscous friction and dry fraction). The list on the left shows the joint type of each joint of the robot created, and the number of elements in the list indicates the degree-of-freedom (DOF) of the robot.

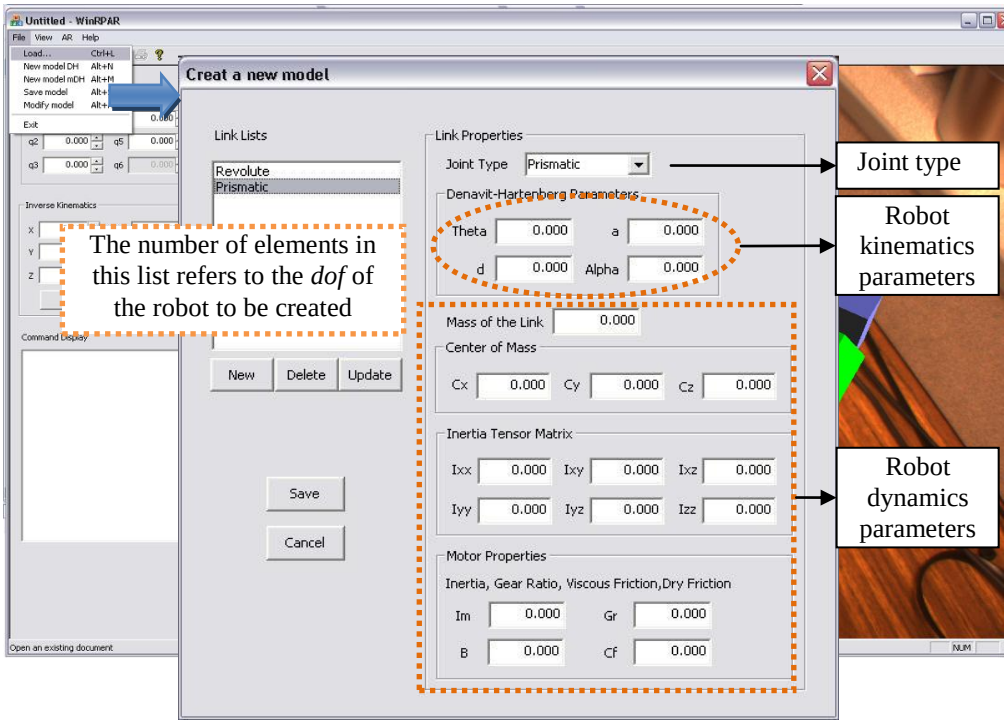


Figure 6.1: Panel for creating a parametric robot model.

The interface for robot parametric modeling has two modes, namely, the creation mode and modification mode. In model modification mode, a known robot model can be loaded into the data structure. By clicking different items in the links list, the respective parameters on the right side will be shown accordingly. These parameters can be modified. The newly created/modified robot model can be exported to a data file in .conf format using this interface for further usage.

Figure 6.2 shows a panel with a number of options for trajectory planning and simulation, including the selection of different log-barrier parameters, the different plot options for planning results analysis, such as the path duration and the energy consumption of actuators, and the tuning options for control gains adjustment.

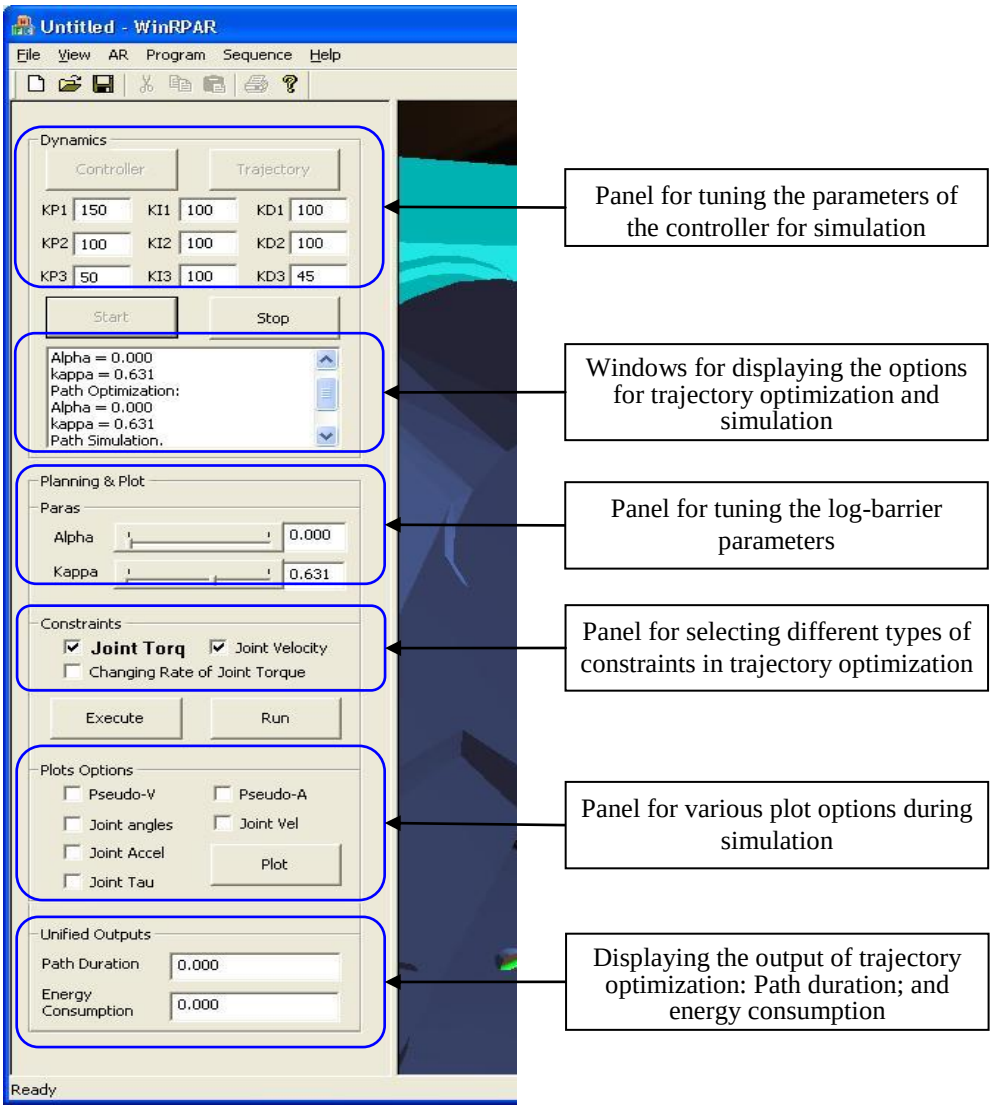


Figure 6.2: RPAR-II system GUI for trajectory optimization and simulation.

6.3 Case Study I: Pick-and-Place Operation

6.3.1 Geometric Path Generation

Figure 6.3 illustrates the process of planning a collision-free path for a robot task, which is to transfer an object from the start point to the goal point. In this example, a total of six control points (excluding the starting and goal points) are created, and each is assigned with normalized spaced time stamp.

(a) An object needs to be moved from the start point to the goal point

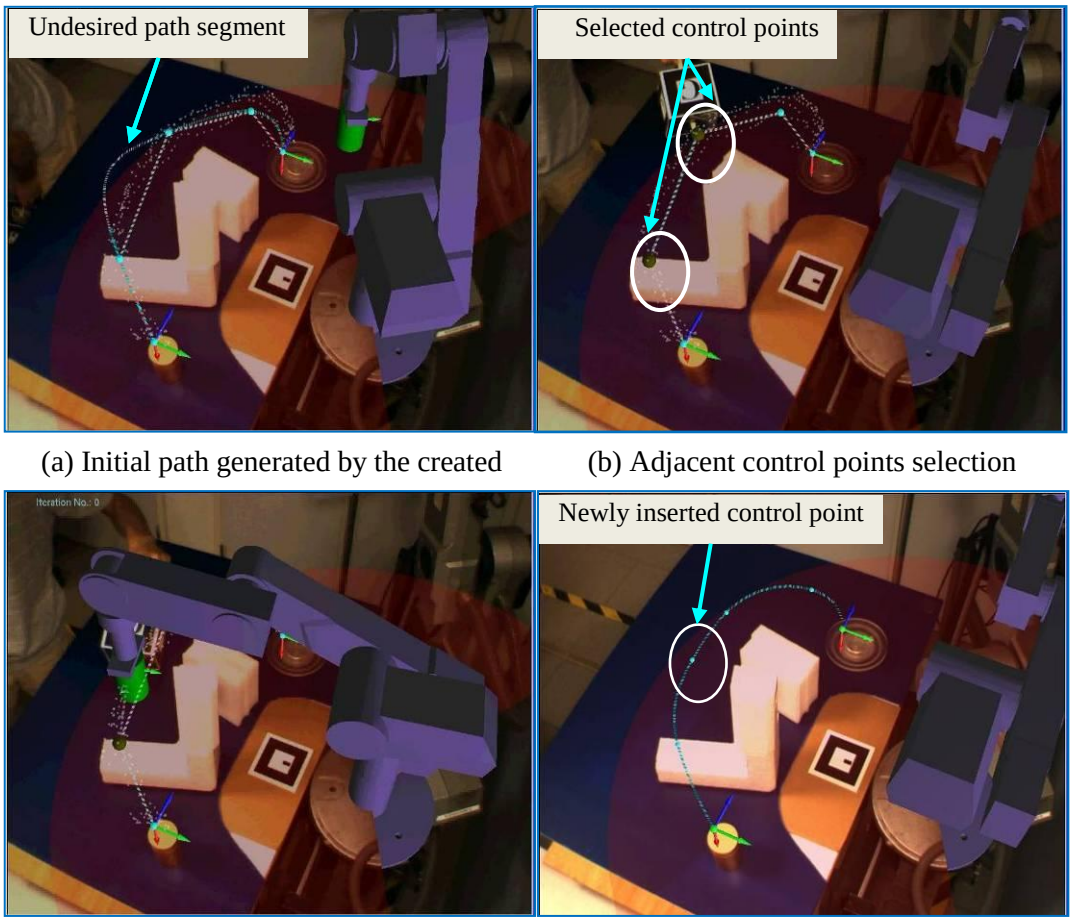
(b) CFV generation

(c) Creation of control points

(d) Geometric path generated by the start point, control points and the goal point

Figure 6.3: Geometric path planning in RPAR-II system.

Figure 6.4 shows a procedure to modify the list of control points when the initial path generated from the created control points is undesirable, e.g., the resulting path generated from these points is outside the CFV. As only three control points have been created (Figure 6.4(a)), one possible way of solving this problem is to insert a new control point that is likely to "drag" this path segment back into the CFV, as shown in Figures 6.4(b)–(c). Figure 6.4(d) shows a path re-generated from the updated control point sequence, and it can be observed that the path has better properties in terms of curvature distribution and smoothness.



(a) Initial path generated by the created (b) Adjacent control points selection

(c) New control point insertion (d) Path re-generation

Figure 6.4: Modification of control points.

6.3.2 Trajectory Planning

Given the geometric dataset of a path, a single thread in the RPAR-II system has been developed for the trajectory planning process. This is to guarantee that the trajectory optimization process would not affect the main thread which handles real-time tracking, registration and rendering. A MFC-based dialog (GUI), as shown in Figure 6.2, is implemented such that the users can select the parameters to construct the different trajectory planning cost model. The trajectory planning results (i.e., path duration and unified energy consumption) can be updated automatically and shown at the "unified outputs" section of the GUI.

In this case study, the path coordinate $\boldsymbol{s}$ is unified, i.e., $s \in [0,1]$, and the step length is set as 0.005 (N_s = 200). Equations (4.13–16) are evaluated at each step along the path to form the coefficients of equations representing the joint torques constraints. The kinematics and dynamics parameters of Scorbot-ER VII model are adopted from reference (Constantinescu and Croft 2000), and shown in Table 6-1 and Table 6-2. The choices of the log-barrier parameter are logarithmically spaced, i.e., 0.01, 0.04, 0.158, 0.631 and 2.51. In this case study, the parameter has been chosen to be 0.631.The optimization variables $\mathbf{b}^0$ are initialized using Equation (4.21) where the pseudo-velocities at the starting and goal points are zero, given an initial value of the constant c = 100.

Table 6-1: Link parameters of Scorbot-ER VII robot using D-H representation

Axis	θ **(°)**	***d*** **(mm)**	***a*** **(mm)**	α **(°)**
1	θ_1	385.5	50	-90
2	θ_2	0	300	0
3	θ_3	-35	250	0
4	θ_4	0	*0*	-90
5	θ_5	231	*0*	0

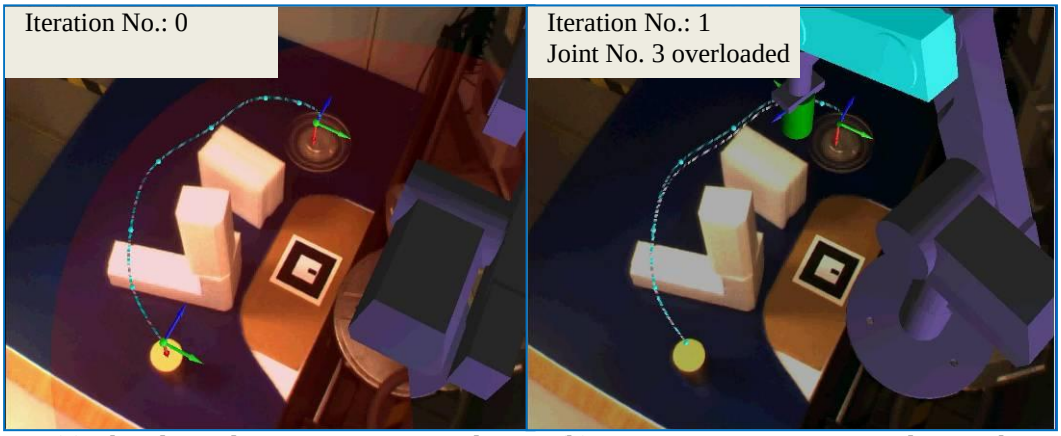
Table 6-2: Dynamics parameters of first three links

Link	**Mass [*kg*]**	$\mathbf{I_x}$**[*kgm*^{2}]**	$\mathbf{I_y}$ **[*kgm*^{2}]**	$\mathbf{I_z}$ **[*kgm*^{2}]**
1	m_1(=0)	I_{x1}=0.00	I_{y1}=0.05	I_{z1}=0.00
2	m_2=6.6	I_{x2}=0.10	I_{y2}=0.60	I_{z2}=0.60
3	m_3=4.2	I_{x3}=0.02	I_{y3}=0.20	I_{z3}=0.30

6.3.3 Trajectory Simulation

The time-scale trajectory obtained was used to simulate the virtual robot carrying out the planned motion for evaluation. The interactive simulation enables the users to preview the planned trajectory (Figure 6.5(a)) and visualize the deviation of the simulated trajectory from the planned one. At each time instance, the link which is most likely to have deviated from the planned path is highlighted in bright colour. Figures 6.5 (b)–(c) show that the computed torques of joints 3 and 2 have violated their limits; such information is displayed on the screen enabling the users to perceive, and the simulation is halted at the current pose respectively. Under these two situations, the derivative gains of joint 3 and joint 2 are adjusted gradually. Figure 6.5(d) shows a satisfactory simulation after the derivative gains have been tuned twice. Nonetheless, if, after a number of iterations (e.g., five iterations), the simulation is still unsatisfactory, the trajectory should be optimized again using a more conservative log-barrier parameter, i.e., a larger log-barrier parameter. The resulting joint torque and joint velocity will become moderately lower, and can be

simulated where the simulated joint torque and joint velocity are within their limits. The solution will be more practical to implement, in the sense of less aggressive usage of the actuators, on a real robot. There will be an iterative trajectory optimization process, as given in Appendix B-2 with a larger log-barrier parameter, and simulation process, until a suitable log-barrier parameter is finally obtained.



(a) The planned trajectory registered over the real working environment

(b) Interactive trajectory simulation: the first trial (unsuccessful)

(c) Interactive trajectory simulation: the second trial (unsuccessful)

(d) Interactive trajectory simulation: the third trial (successful)

Figure 6.5: Trajectory planning and interactive simulation.

6.3.4 Trajectory Implementation on Real Robot

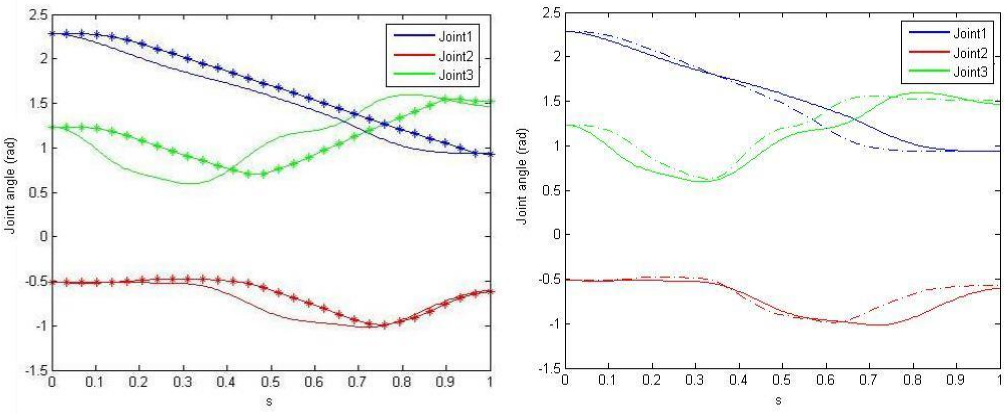
Trajectory planning in the previous sections has been transferred into robot controller codes and executed on a real robot. In this case study, the maximum

planned speed of the EE which is retrieved from the planned time-optimal trajectories, for the given pick-and-place task is around 850 mm/s. According to the robot user manual, the maximum speed of Scorbot ER-VII is 1000 mm/s. so in the robot programs, the robot speed is set to 85, which means the maximum motion speed is set to 850 mm/s.

Figure 6.6(a) shows the robot joint angle profiles of a real robot running the planned motion. Comparatively, Figure 6.6(b) is the profile when running at a scaled-down speed. It can be observed that with an increased running speed, the maximum offset of the actual motion deviated from the planned motion is increased, and the latency at each joint is increased. The latency could be due to the un-tuned control schemes (control gains) when the robot running at different speed levels as a higher running speed requires a faster control response (less response time). In addition, it might due to the coupled robot dynamics effect. Nevertheless, it is consistent with the fact that the robot dynamics will have more influence on the motion when the robot is running at a comparably higher speed.

Figure 6.7 shows the results in the Cartesian space of a real robot running the planned motion at the full planned speed and a scaled-down speed, respectively. The positions of EE are projected onto 3 planes, e.g., X-Y plane (Figure 6.7 (a–b)), X-Z plane (Figure 6.7 (c–d)) and Y-Z plane (Figure 6.7 (e–f)). It can be observed from Figure 6.7 that at a relatively lower running speed, such as 30% of the planned speed, the EE follows the planned motion quite well in the X-Y plane, but it is also clear that there are offsets in the Z direction. This could be due to the

gravity effect of the robot arms (particularly the robot upper-arm and fore-arm which can only move vertically), which has not been compensated in the current PID control scheme applied to the robot controller. In addition, with a higher running speed, the robot motion in the X-Y plane is smoother, despite that the maximum offset is larger. It can be observed from the plots in the Y-Z plane and the X-Z plane that there are more fluctuations/jitters.



(a) Running at full planned speed (b) Running at scaled-down speed (30%)

Figure 6.6: Actual path of the EE in the joint space.

In addition to the coupled robot dynamics and un-tuned robot control scheme, another cause for the offsets of the actual motion deviated from the planned motion could be the improper way of transferring the trajectory to the controller codes. The trajectory is planned with optimization variables associated with the constant path length, and the optimization results (set points) are given based on this path length as a reference. So it is not appropriate to transfer these set points to the controller directly as the controller cannot program the trajectory by setting specific speeds at each running step. An alternative way is to first retrieve efficient set points according to the optimization results in the time domain

(considering the control cycle of the controller), then transfer these points to the controller for execution.

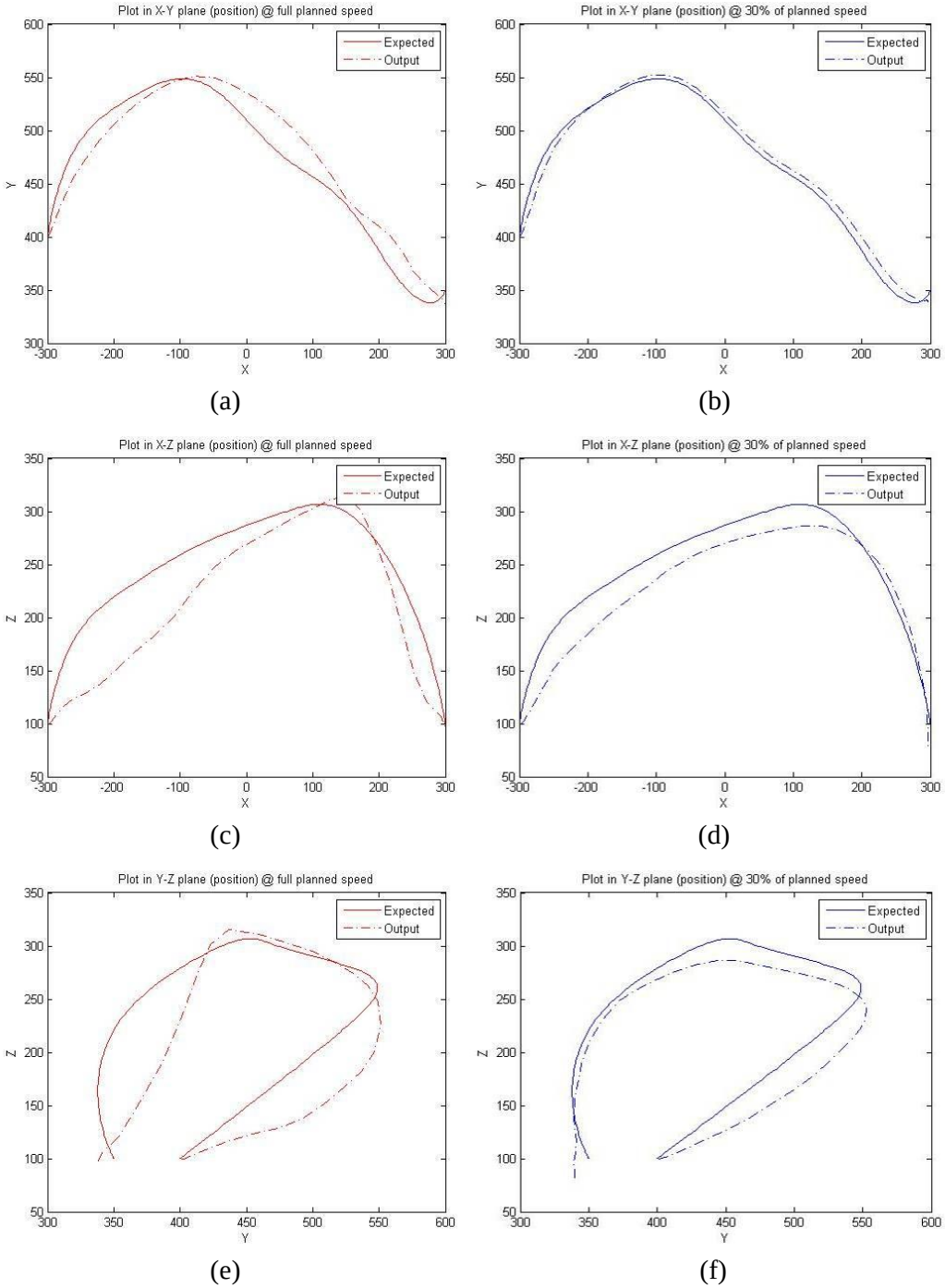


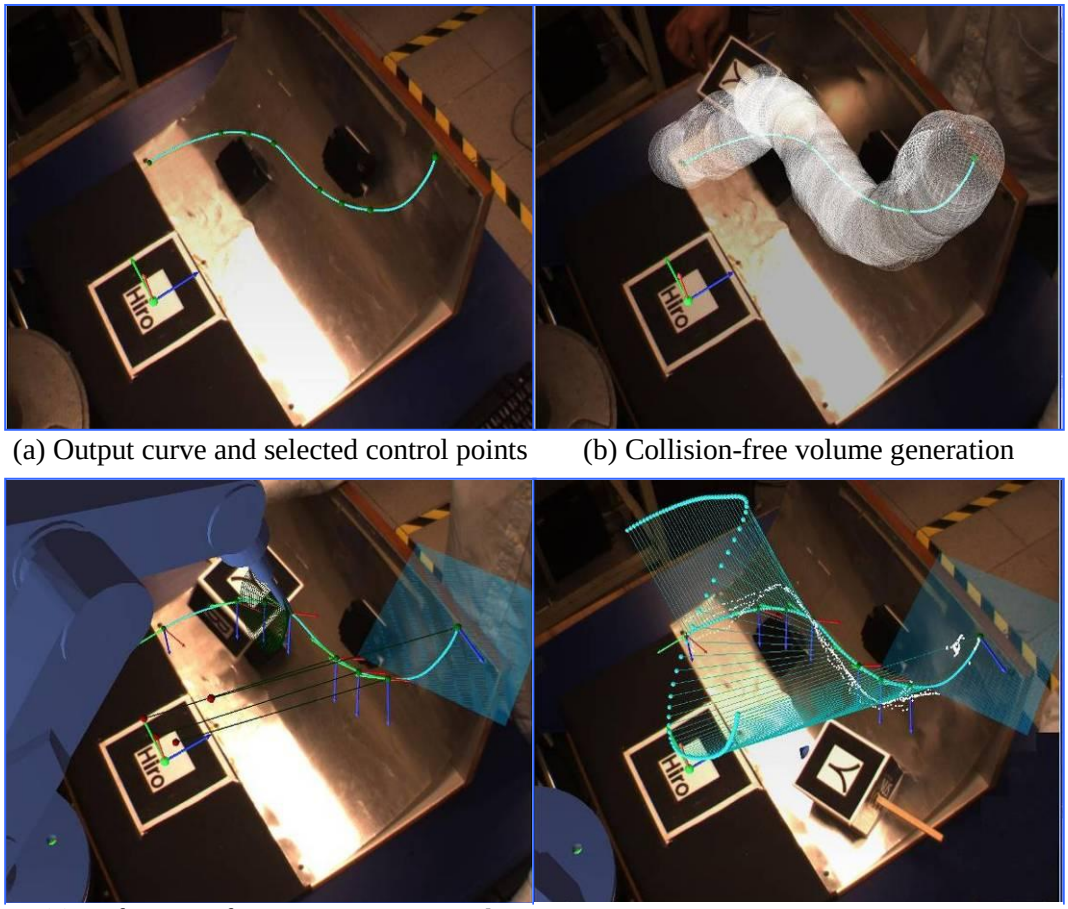
Figure 6.7: Actual path of the EE in the Cartesian space.

6.4 Case Study II: A Spatial *S*-shaped Curve

This section presents a case study to illustrate the proposed methodologies for EE orientation planning. The target path is an *S*-shaped curve which lies on a curved surface, and the trajectory to be planned should allow the robot EE to travel along the path at orientations within an acceptable range with respect to the path.

6.4.1 End-effector Orientation Planning

Figure 6.8 illustrates the procedure for planning a robot task where an axis-symmetrical EE, which is mounted coaxially to the EE holder, is constrained to follow a visible curve. Figure 6.8(a) shows an output curve and the selected control points on the workpiece. The curve model, which can be represented in the form of Equation (5.3), is first obtained through the efficient neural network based learning and parameterization approach, given a set of data collected from human demonstrations in tracing the unknown curve using an interaction device. Next, a CFV is generated along the output curve (Figure 6.8(b)), and the coordinate frame is defined interactively at the first control point; the user proceeds to specify the orientation of the EE at each control point through guiding the virtual robot using the interaction device (Figure 6.8(c)). Lastly, using these orientations and the output curve as inputs, a ruled surface is generated automatically through cubic-spline interpolation (Figure 6.8(d)). The orientation profile is interpolated with the same interpolation parameter as the output curve, such that the orientation interpolated can be associated with the points re-parameterized from the curve model.



(a) Output curve and selected control points

(b) Collision-free volume generation

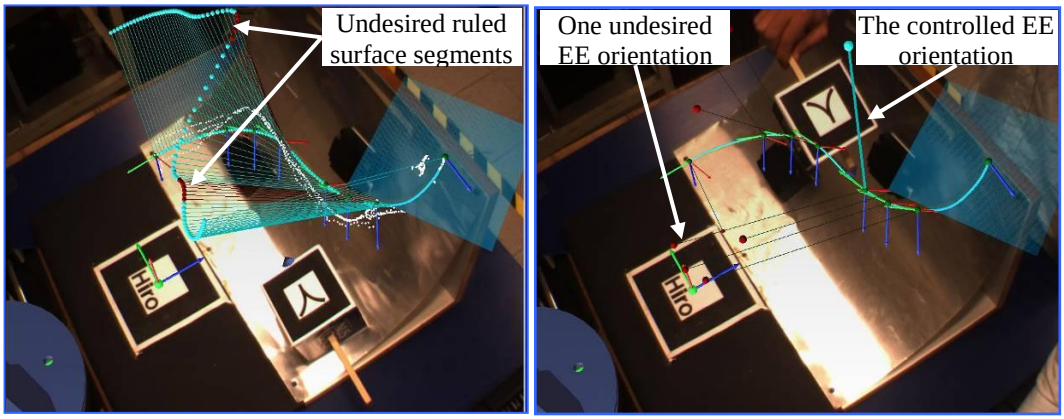
(c) Definition of EE orientation at each control point

(d) Ruled surface generation

Figure 6.8: Robot End-effector orientation planning using the RPAR-II system.

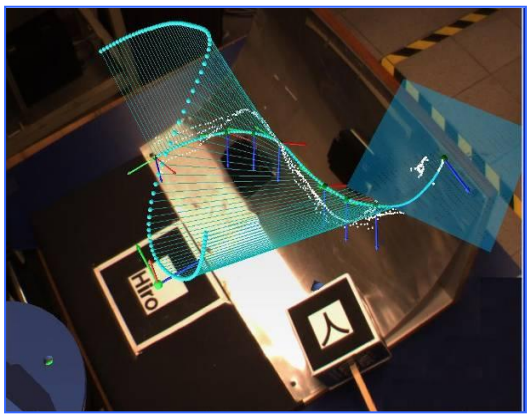
The ruled surface generated from the EE orientations defined at the control points may be undesirable; such as the case shown in Figure 6.9(a) where the EE collides with the CFV when travelling along the output curve at the orientations planned. Figures 6.9(b)–(c) show one way of improving the resulting ruled surface through modifying the orientations of the EE at relevant control points. The user first selects a control point, which belongs to the path segment where the collision occurs, and guides the virtual robot to select an orientation of the EE at

this point (Figure 6.9(b)) to regulate the ruled surface. Figure 6.9(c) shows an improved ruled surface re-generated from the updated control point list and the associated orientations of the EE. Nevertheless, the number of the control points should be maintained within a reasonable range; otherwise it would likely to have overly constrained control points on the EE orientations and produce a ruled surface with excessive variations. The modification operations will be repeated until the ruled surface generated is satisfactory, i.e., collision-free, and smooth with good curvature distribution.



(a) Initial ruled surface generated by the EE orientations defined at control points

(b) Modification of EE orientation defined at a control point



(c) Ruled surface re-generation

Figure 6.9: Ruled surface modification.

If all the rulings of a ruled surface are moved parallel to pass through the origin of a universal coordinate frame (e.g., defined at the TCP), the spherical indicatrix of the rulings can be represented by a bundle of lines concurrent at the origin (Tsai and Stone 2009). If one end of the ruling coincides with the origin, the other end of this ruling will be lying on the surface of the sphere centered at this origin. Figure 6.10 shows the intersections of three ruled surfaces, which are generated from the orientations defined using five, seven and nine control points (including the start and the end of the curve) respectively, with a unit sphere. The curve on the spherical surface associated with nine control points swings back and forth about the one associated with five control points. This means that using more control points, despite of resulting in a ruled surface in a more controlled manner, may yield more directional variation than using less control points. Nevertheless, fewer control points may require more iteration for orientation adjustment in order to generate a collision-free ruled surface.

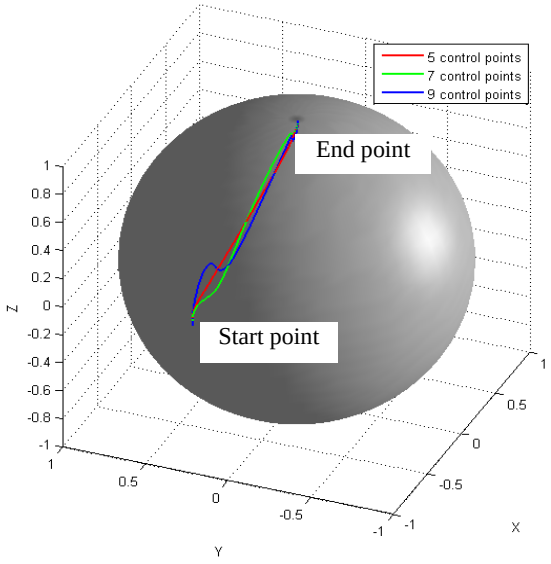


Figure 6.10: Intersections of three ruled surfaces with a unit sphere.

6.4.2 Trajectory Optimization and Simulation

In the proposed system, trajectory optimization is performed in the joint space. The kinematics and dynamics parameters of the Scorbot-ER VII are given in Table 6-1 and Table 6-2. Coulomb friction has also been modeled, which is $1.0\ N \cdot m$ for each link. The torque and velocity limits for each link are set to be $10\ N \cdot m$ and $1.0\ rad/s$ (Constantinescu and Croft 2000). Therefore, the generated ruled surface, as illustrated in Figure 6.8(d), has to be mapped into the joint space after it has been parameterized and represented in path coordinate ***s***. In this case study, similar to Section 6.3.2, the joint torque constraints and the joint velocity constraints are evaluated at each step along the path. The choices of each log-barrier parameter associated with the corresponding constraints are logarithmically spaced, e.g., $\kappa_\tau \in \{0.01, 0.04, 0.158, 0.631, 2.51\}$, $\kappa_v \in \{0.04, 0.158, 0.631, 2.51\}$. A moderate $\boldsymbol{\kappa}$ has been chosen, i.e., $[\kappa_\tau\ \kappa_v]^T = [0.631\ \ 2.51]^T$, to determine an approximately time-optimal trajectory. The optimization variables $\mathbf{b}^0$ are initialized through evaluating the Equations (4.7)-(4.12), (4.19), (4.21), and (4.23) where the pseudo-velocities at the two end points of the curve are zero, given an initial value of the constant ***c*** =50.

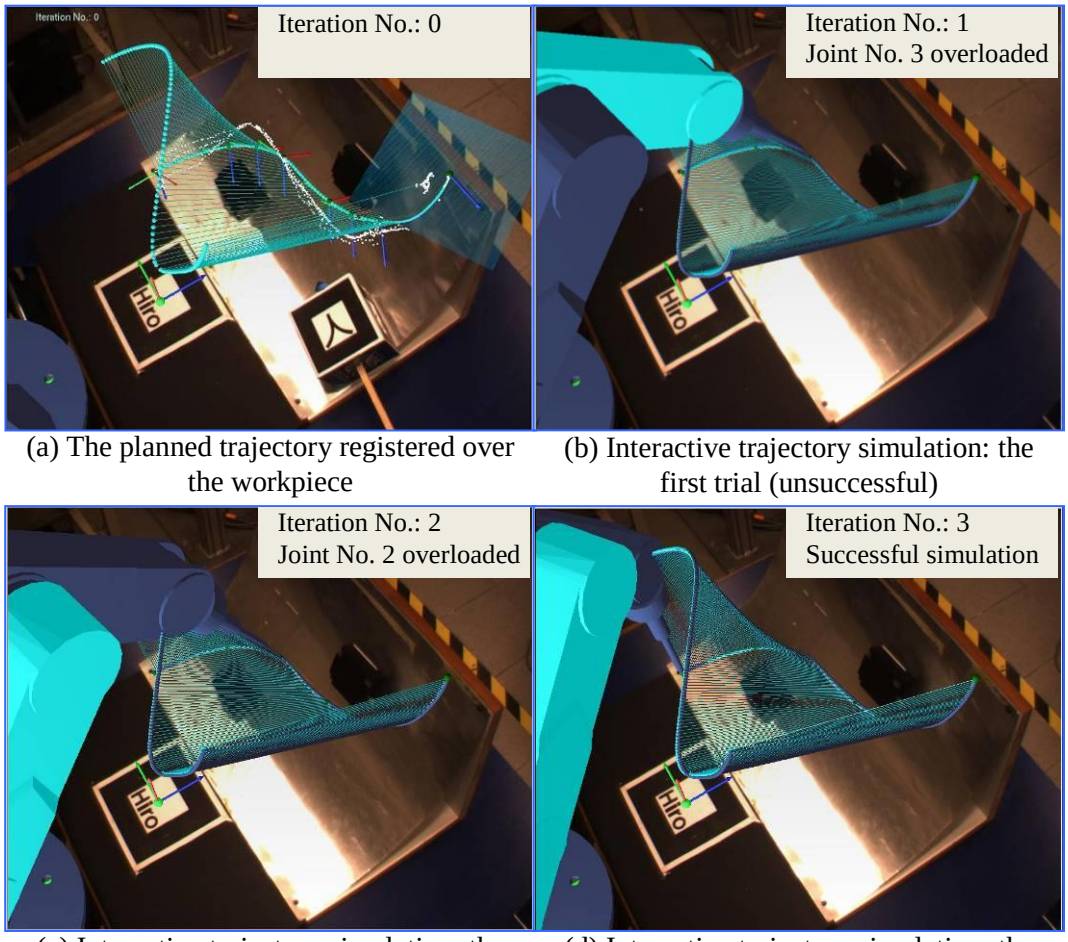
The trajectory planned above has been implemented and evaluated through simulation using a virtual robot. A discrete PD controller has been modeled and initiated with the proportional coefficients 150, 100, 50, and the derivative coefficients 100, 100, 50 respectively. By rendering both the planned and

simulated trajectories, in the forms of ruled surfaces, onto the workpiece (Figure 6.11(a)), the user can inspect the deviations of the simulated trajectory from the planned one. At each time instance, the link which is most likely to have deviated from the planned path is highlighted. Figures 6.11(b)–(c) show that the computed torques of joints 3 and 2 have violated their limits; the system displays this information on the screen enabling the user to perceive the information, and the simulation is halted at the current pose. Under these two situations, the derivative gains of joint 3 and joint 2 are adjusted gradually. Figure 6.11(d) shows a satisfactory simulation after the derivative gains have been tuned twice. If the simulation is still unsatisfactory after a number of iterations (e.g., five iterations), the trajectory can be optimized using more conservative log-barrier parameters.

6.4.3 Accuracy Evaluation

There are two main sources of errors affecting the overall accuracy of the proposed system, namely, curve tracking error due to the specific tracking method adopted, and robot modeling error, i.e., the kinematics and dynamics modeling error of the physical robot. Figure 6.12(a) presents the tracked curve used in the case study and known desired curve (in practice, the desired curve is unknown) in the marker frame $\boldsymbol{M}$ (refer to Figure 3.5), and Figure 6.12(b) depicts the tracking errors in the X-, Y- and Z-directions of the marker frame. The average tracking error is approximately 11.0 mm, given that the camera is installed at 1.5 m away from the workpiece. The error is mainly caused by the ARToolKit tracking method adopted in the process of acquiring 3D data points. In this case study, the tracking error, particularly the error in the Z-direction in camera's coordinate frame, is closely

dependent on the orientation and/or position of the marker-cube when moving along the visible curve. It is worth noting that the EE orientation planning process would not introduce extra errors as the control points are selected from the curve model. To this end, the tracking errors do not severely affect the goal of the research, which is the evaluation of the proposed method for planning the EE orientation along a visible curve.



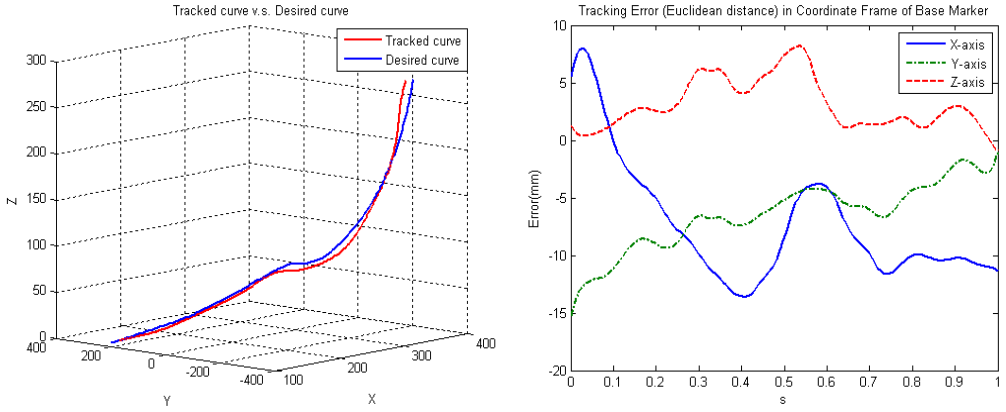
(a) The planned trajectory registered over the workpiece

(b) Interactive trajectory simulation: the first trial (unsuccessful)

(c) Interactive trajectory simulation: the second trial (unsuccessful)

(d) Interactive trajectory simulation: the third trial (successful)

Figure 6.11: Trajectory planning and interactive simulation.



(a) Tracked curve and Desired curve in the base marker frame

(b) Path tracking error in the base marker frame

Figure 6.12: Curve tracking accuracy.

6.4.4 Trajectory Implementation on the Real Robot

The planned and simulated trajectories have been executed on the Scorbot-ER VII robot. It should be noted that the trajectory is planned with optimization variables associated with a constant path length, and the optimization results represented by a set of joint angles are given based on this path length as a reference. Hence, it would be inappropriate to transfer this set of joint angles to the controller directly as the controller cannot program a trajectory by setting specific speeds at each running step. In this case study, an alternative set of joint angles is retrieved from the optimization results in the time domain. Trajectories that have been planned without considering robot dynamics are also implemented on the real robot to show the effect of the robot dynamics. Figure 6.13 shows two flowcharts designed for testing the planned trajectory, namely, Figure 6.13(a) considering robot kinematics and dynamics, and Figure 6.13(b) considering robot kinematics only.

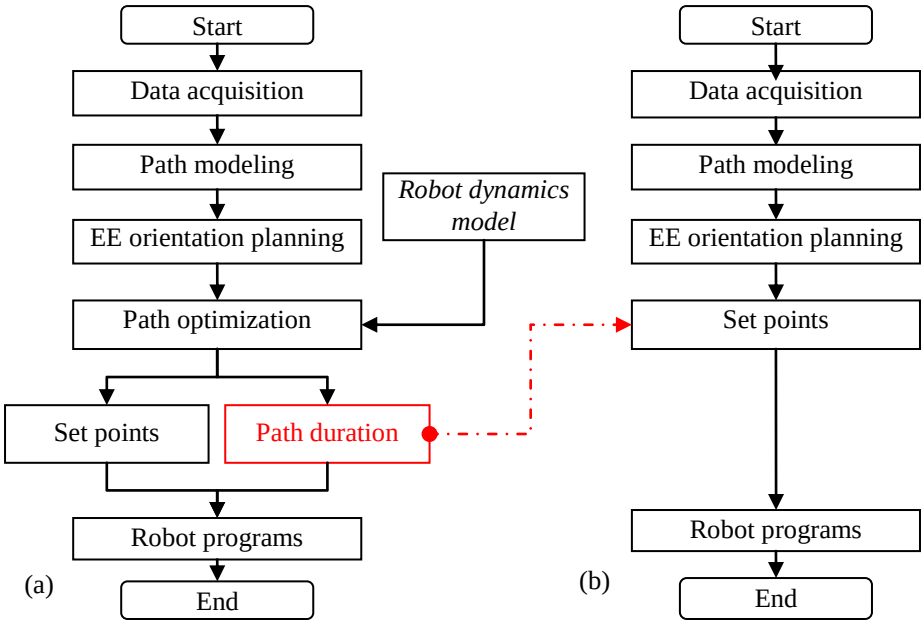


Figure 6.13: Two flowcharts for testing the planned trajectory.

In this case study, the Advanced Control Language (ACL) provided by the robot controller is used to read and translate the trajectories into controller codes that are fed to the controller loop at the controller frequency. A felt-pen is mounted as the axis-symmetrical EE to replicate a robotic arc welding torch. The plots in Figure 6.14 are the actual trajectories of the EE obtained considering both robot kinematics and dynamics constraints, represented in the Cartesian coordinates, as compared with the referenced curve which has been acquired using the curve learning techniques presented in Ong *et al*. (2010). The discrepancies, as shown in Figure 6.14, are caused by a number of sources. They could be inherent from the tracking errors introduced during the curve tracing process using the marker-based tracking techniques, and the robot dynamics modeling errors where only a set of estimated robot dynamics parameters are implemented in the trajectory optimization process. Figure 6.15 shows the actual EE orientation along

the trajectories implemented on the real robot. Figure 6.15(a) represents the joint angles of the 4th joint based on the D-H convention. Similar to Figure 6.10, Figure 6.15(b) is the intersection of a unit sphere with the planned and the actual EE orientation profiles, both are represented in the robot base frame. The deviation exhibited in Figure 6.15(a) could be due to several reasons. Firstly, this joint has exceeded its singular configuration when traveling along the planned path; and secondly, the dynamics of the orientation DOF of the robot has not been considered during trajectory optimization.

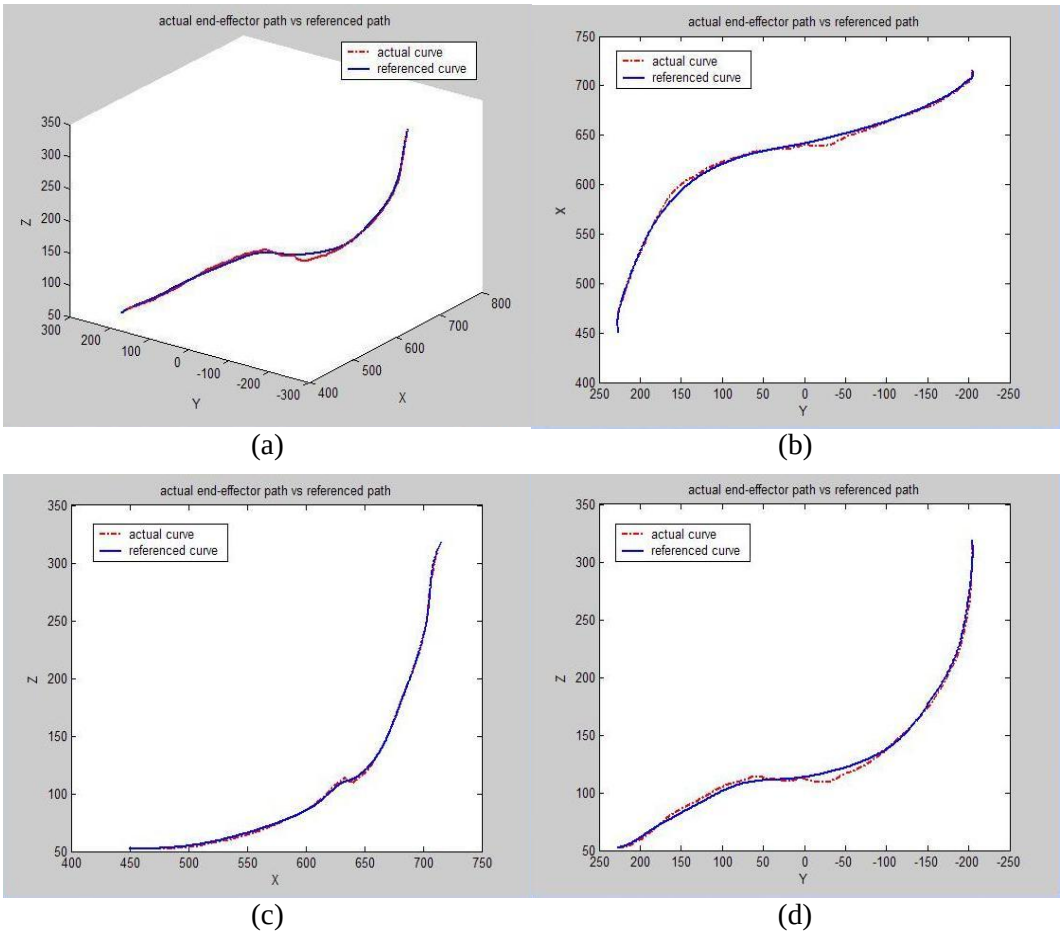
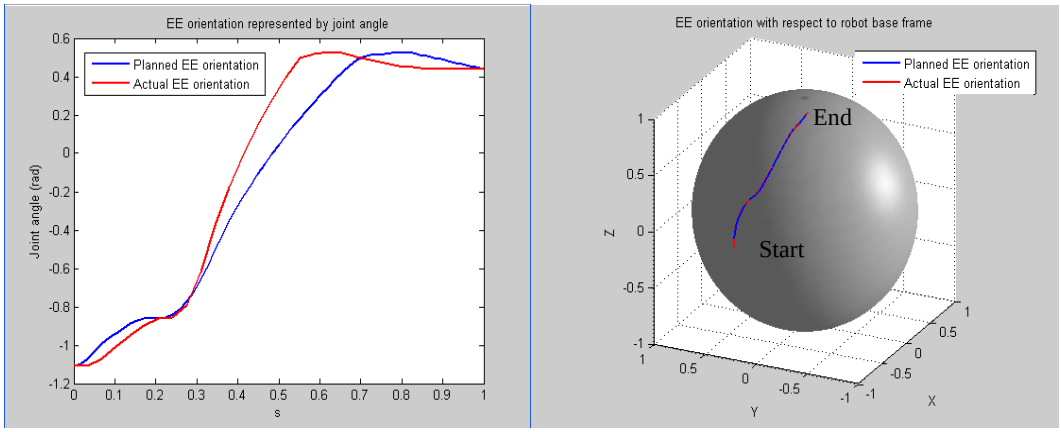


Figure 6.14: Trajectory implemented on real robot.



(a) EE orientation with respect to joint coordinate frame

(b) EE orientation with respect to robot base frame

Figure 6.15: EE orientation on real robot.

Figure 6.16 shows the actual trajectories of the EE obtained considering robot kinematics only, represented in the Cartesian space, compared with the referenced curve. The path duration is the same as the trajectory illustrated in Figure 6.14. The set points are sampled along the curve with equal Euclidean distance, such that the speed of the EE moving along the cure is constant. It can be observed in Figure 6.16 that the actual motion of the EE swings back and forth with respect to the reference curve. In addition, larger discrepancies, as compared with Figure 6.14(a), can be perceived. Hence, it can be concluded that the trajectory which is obtained by incorporating robot dynamics, even a set of estimated robot dynamics parameters, will outperform the trajectory that determined without considering robot dynamics effects.

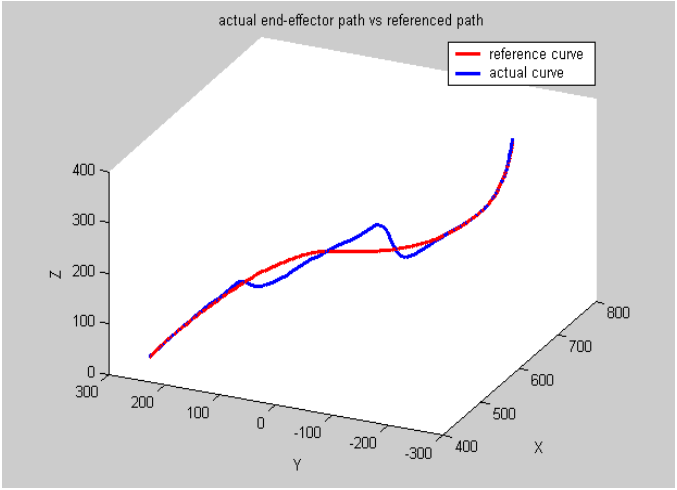


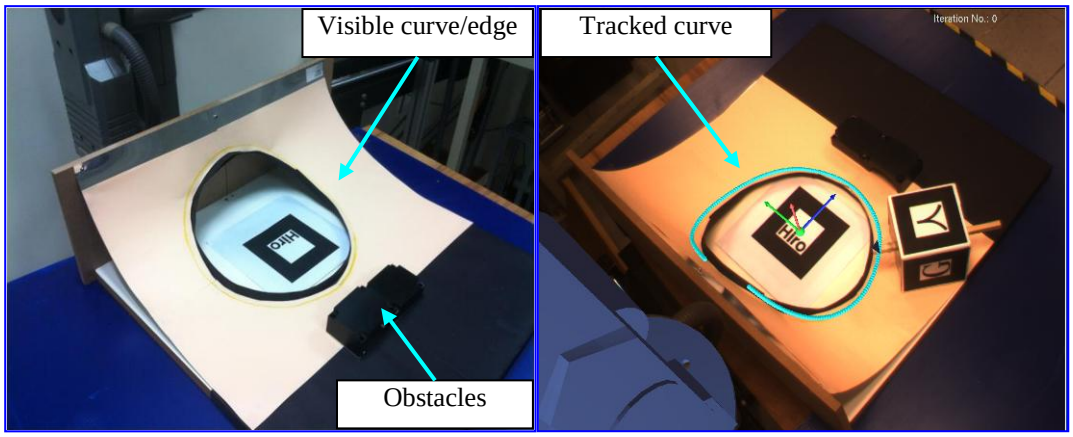
Figure 6.16: Implementation of trajectory without considering robot dynamics.

6.5 Case Study III: A Spatial Circular Curve

This section presents another case study to showcase the proposed method for EE orientation planning. The target path is a circular curve, as shown in Figure 6.17(a), where the orientation of the EE needs to be planned appropriately to avoid the obstacles as well as the edge along the visible curve. This case study is designed to emulate robot operations, such as robot arc welding, gluing, etc.

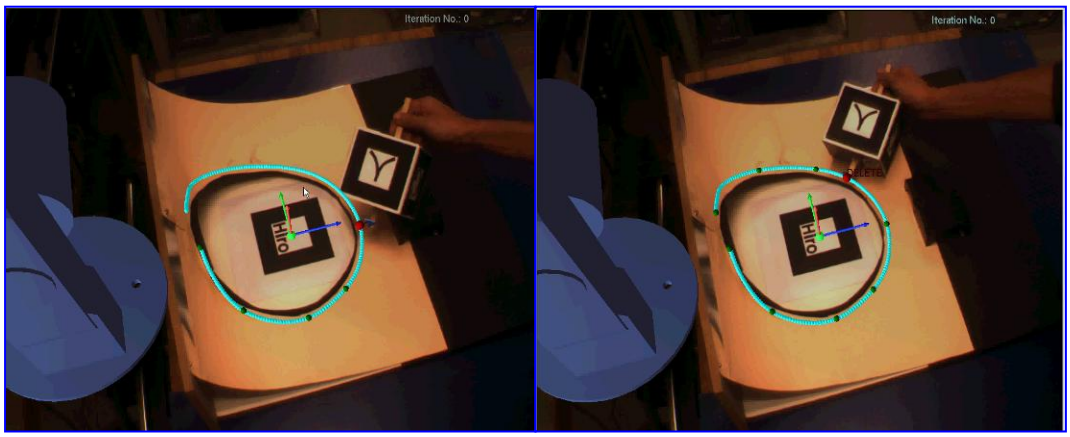
Figures 6.17-19 illustrate the process of using the proposed approach in planning the orientation of the EE along a circular curve. In particular, Figure 6.17(b) registers the tracked curve onto the real workpiece. The procedure for control point selection and modification is depicted in Figure 6.18. Figure 6.19(a) illustrates the determination of the coordinate frame at the first control point. Figure 6.19(b) is the planning of the EE orientation at a control point with respect to the coordinate frame defined at this point. Figure 6.19(c) gives the ruled

surface generated from the orientations defined at all the control points. The setup and selection of the parameters for the trajectory optimization are the same as in case study II. Figure 6.19(d) shows that the planned trajectory is simulated successfully after the robot controller gains have been tuned.



(a) (b)

Figure 6.17: (a) Workpiece for the case study on following a circular curve; (b) Output curve model.

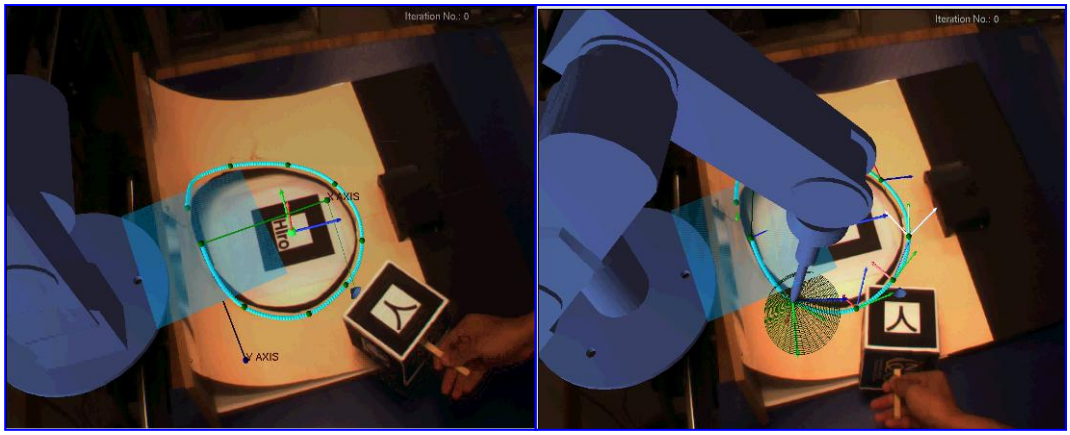


(a) Selection of a list of control points (b) Selection of a control point to be deleted

Figure 6.18: Control point selection and modification

6.6 User Study

This section presents the user study on the proposed AR-based interface. Ten researchers, seven male and three female researchers, from the Mechanical Engineering Department were invited to conduct the experiments. All participants are not familiar with robotic systems, particularly in robot path and task planning, while eight of them have experience in the use of AR-based systems.



(a) Coordinate frame definition at the start of the output curve

(b) Definition of EE orientation at each control point

(c) Successful ruled surface generation

(d) Successful trajectory simulation

Figure 6.19: EE orientation planning and simulation.

A training session of approximately 15 minute is first given to the users to learn the use of the interaction device to guide the EE of the virtual robot, and be familiarized with the procedure designed for EE orientation planning, including control point selection, path coordinate frame definition, EE orientation determination at each control point, and EE orientation modification. The user study is composed of two parts, namely, a system experiment and a questionnaire-based survey. The questionnaire constitutes two sets of questions. One set of questions, which is to be filled by every participant before the test, is to evaluate the participants' background on their experience in the use of AR-based systems and familiarity with robotic task planning skills. Another set of questions is on the participants' evaluation of the AR-based interface, as well as the use of visual cues as visualization enhancements, upon their completion of each planning task.

Two robot tasks have been carried out for the user study. The first task, which is a robot pick-and-place task, is designed to evaluate of the proposed HRI interface for geometric path planning and generation, and the setup is given as in Figure 6.3(a). In this task, the participants were asked to select a number of spatial points between the starting and goal points and a path will be generated from these points. The second task is a robot path following task, which emphasizes on the performance of the HRI interface in robot EE orientation planning and adjustment. In this task, as illustrated in Figure 6.8(a), the participants were asked to select a series of spatial points on a visible curve, then to define the EE orientations on these points. By doing so, an EE orientation profile along the visible curve can be generated. Each of the two tasks has been carried out in two different conditions,

which can be differentiated by the increasing levels of situational awareness to the working environment, as detailed next.

(1) The limit suite of the functions of the proposed AR-based HRI interface that allows the users to view the real environment and interact with the virtual robot without spatial point modification or robot EE orientation adjustment.

(2) The full suite of the functions of the proposed HRI interface that allows the users to view the augmented environment and perform robotic task planning, EE orientation planning and modification. The planned paths can be simulated and reviewed prior to actual execution.

The first condition can be adopted to mimic the planning process using traditional *teach-in* robot programming method, in which it is difficult or even impossible to modify the selected spatial points during planning process. If a planned path is not successful, the spatial points need to be re-created. Comparatively, the full suite of the proposed method permits the users to adjust the spatial points in case the generated path based on these points is unsatisfactory.

A monitor-based visualization is used to present the augmented view of the working environment as well as the necessary visual cues to the users. The objective and the sequence of each task were explained to the participants. Every participant was first allowed to learn and practice the use of the interaction device in guiding the EE of the virtual robot moving around the workspace, and to get

familiarized with the sequence of the tasks. Before each trial, a CFV has already been generated, and the participants were only responsible for the spatial point selection or EE orientation definition. This is to ensure that under the first condition, each point selected is within the CFV and thus the corresponding robot configuration is collision-free. This mimics the process of spatial point selection in *teach-in* robot programming in which an operator operates the real robot using a teaching pendant.

Since the selected spatial points cannot be modified under the first condition, it would be obvious that the more spatial points are being created, the higher the possibility that the path generated from these points is collision-free. Therefore, under this condition, each participant was asked to select ten spatial points considering the complexity of the work environments as shown in Figure 6.3(a) and Figure 6.8(a). Comparatively, there is no such constraint in the second condition since the spatial points are modifiable in case the generated path has collision with the CFV. Each participant performed four trials, i.e., the two tasks each under two distinctive conditions. For each trial, the time to completion, possible collisions and the task completion rate are measured. A collision is defined as where a path segment is outside the generated CFV, or where the swept model of the EE is outside the CFV when moving along the visible path.

Figure 6.20 shows the time to completion of the pick-and-place task under the two conditions, where the experimental conditions have significant effect on the time to task completion, i.e., approximately 206 seconds (standard error is 45 seconds)

under the first condition, and 125 seconds (standard error is 18 seconds) under the second condition. In addition, under the first condition, only one out of the ten participants was able to select the spatial points yielding a collision-free path at their first attempt. Another three participants completed the task in their second attempt, and the rest needed more than two attempts to create a set of suitable spatial points to form a satisfactory path. It should also be noted that in actual application, it would take more time than in the experiments to create the spatial points, as the user needs to manipulate the real robot arms moving to a series of desired positions and record them accordingly.

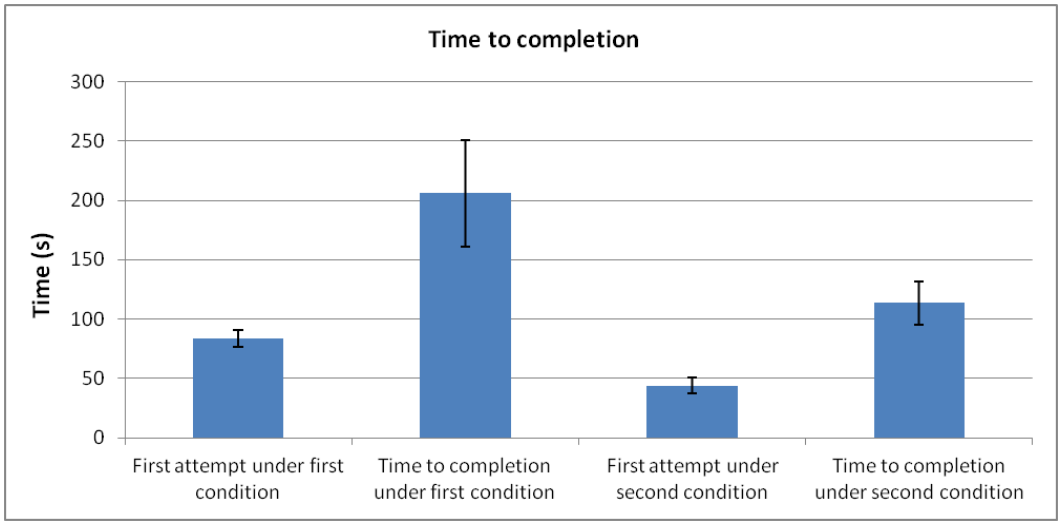


Figure 6.20: Average times to complete the robot pick-and-place task.

For the path following task, similarly, a CFV has been generated in advance around the visible curve. Each participant in the first trial was asked to select ten robot configurations along the visible curve, each consists of both robot EE position and orientation, at which the robot EE is within the CFV. If the EE orientation profile generated from these configurations collides with the CFV, the

ten configurations would need to be re-generated until the resulting orientation profile is satisfactory. In the second trial, the participants can first select some spatial points on the curve, and define the EE orientation associated with each spatial point correspondingly. In case the resulting orientation profile of the robot EE is unsatisfactory, the participant can adjust the EE orientation at the relevant spatial points, or edit the list of spatial points if he/she feels the need to. Figure 6.21 shows the time to completion of the task for EE orientation planning under the two conditions. The average time for completion of the EE orientation planning under the first condition is nearly 605 seconds (standard error is 87 seconds) and about 337 seconds (standard error is 74 seconds) under the second condition. It has been observed that all the participants have failed in their first attempt to plan a satisfactory EE orientation profile along a given path.

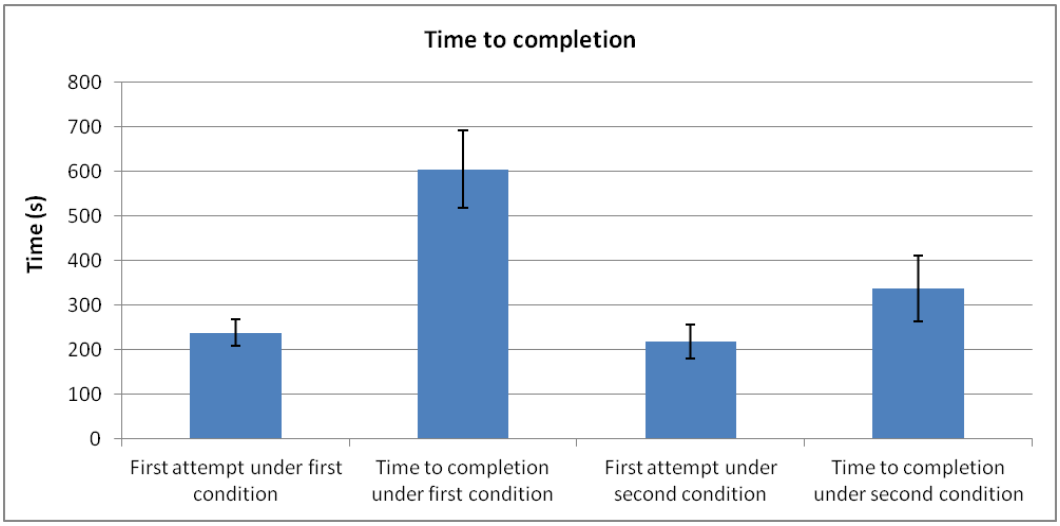


Figure 6.21: Average times to complete the robot path following task.

Figure 6.22 shows the time taken by each participant to complete the procedure designed for robot EE orientation planning along a visible curve. For participants

9 and 10, the time for EE orientation modification was zero, which indicates that the ruled surface generated using the selected EE orientations is satisfactory and there is no need to perform the orientation modification. Figure 6.23 shows the maximum, minimum and average time to achieve each step of the procedure among all the participants.

From the user studies, the participants felt that they were able to interact with the virtual robot in its working environment using the interaction device. In particular, they can achieve the creation, selection and modification of the spatial points quickly and easily in the robotic pick-and-place robot task. They also felt intuitive and convenient to carry out these operations on the visible curve model in the robotic path following task, even though they demonstrated that there were some difficulties in the determination of a suitable EE orientation at each spatial point. Such difficulty may be caused by the misalignment between the virtual EE and the interaction tool as the virtual robot model has a reduced wrist configuration. It is reflected in Figure 6.22 and Figure 6.23 that it takes a much longer time in the definition of the EE orientation than in the selection of the control points. The difficulty may be attributed to the fact that the virtual EE cannot always be aligned with the interaction tool as the virtual robot model has a reduced wrist configuration. In addition, they reported that it was time-consuming to perform the tasks under the first condition, as it required them to remember the unsatisfactory segments on the previous generated path or orientation profile and redefine the spatial points or orientations carefully in the neighborhood of these segments. As demonstrated in planning the robot path following task, particularly

in the first attempt of each participant, most collisions occurred at the path segments that were adjacent to the obstacles.

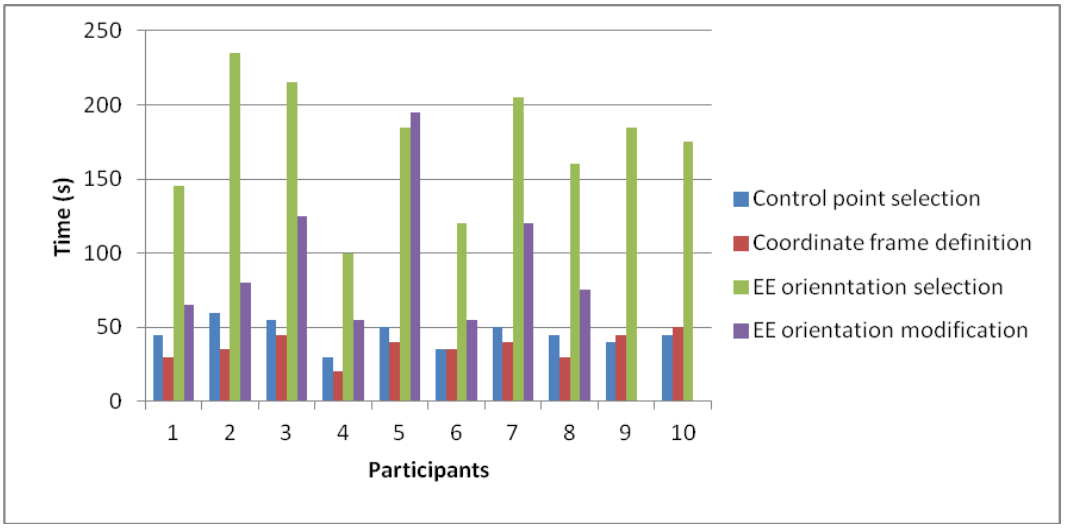


Figure 6.22: Individual time of each participant to complete the procedure for EE orientation planning under the second condition.

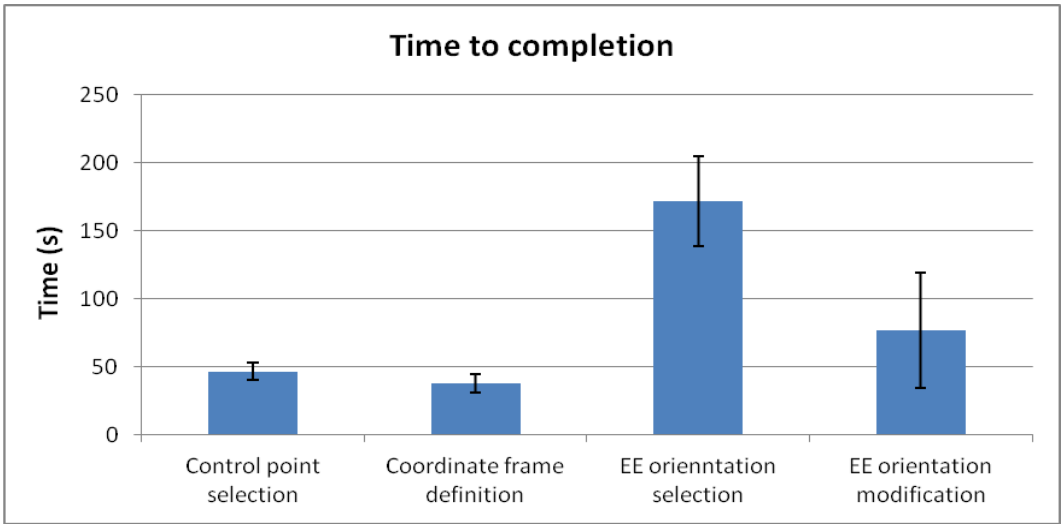


Figure 6.23: Average times to complete each step in EE orientation planning.

With regards to the visual cues and feedback presented on the monitor screen, the participants rated that it has helped them understand the planning process and

correction operations. Particularly under the second condition, they felt that the use of such virtual contents as cues makes it easy and flexible in their interaction with the virtual environment that facilitates the completion of path and EE orientation planning tasks. However, they felt distracted that the virtual EE disappeared when the interaction device moves out of the working range of the robot. The participants who are using AR systems for the first time tended to occlude the base marker with the marker-cube or move the marker-cube out of the field of view of the camera, making the virtual robot disappear from their views. In addition, they experienced distractions and fatigue as they needed to alternate their attentions between the perception of the augmented environment through the monitor screen and the manipulation of the interaction device in the real working environment. Meanwhile, the participants tended to focus on the monitor's view when performing the tasks. This will to some extent lead to improper guidance of the virtual robot as it is not easy for the operator to perceive the depth information from the display on the monitor. The use of a head-mounted display instead of the monitor could solve these issues and thus improve the performance of the proposed method significantly.

The results from the user study have suggested some advantages of using the proposed AR-based HRI method over the conventional *teach-in* method in which a teaching pendant is normally used to assist the operator in robot task planning. First, inexperienced users are able to learn the method quickly and interact with the virtual robot using the proposed HRI interface. Secondly, the proposed interface facilitates faster robot programming and path planning. Thirdly, the Euclidean

distance-based method allows the users to select a spatial point of interest easily for insertion or deletion. The use of visual cues increases the intuitiveness of the AR-based HRI, and guides the users during the interaction with the virtual robot, e.g., spatial point selection and modification, as well as EE orientation definition and adjustment, in planning a given robotic operation. In the pick-and-place task, the path formed by the spatial points can be updated simultaneously once one or more points are being modified. In the path following task, the EE orientation profile is re-generated immediately once the EE orientation at a spatial point has been modified. This enables the users to be aware immediately of the results of their interactions with the virtual robot and the working environment.

6.7 Summary

This chapter presented the case studies for validating the proposed methodologies for robot path planning, EE orientation planning, as well as simulation based on the applications of AR. The programming environment, implementation tools, and external packages used in this study are introduced. Three case studies have been conducted, one is for finding a collision-free and smooth path, given a pair of start and goal configurations, incorporating the robot actuator capabilities; another two are for orientation planning of robot EE along two 3D curves, namely, an *S*-shaped curve and a circular curve, each on an aluminum sheet.

Chapter 7. Conclusions and Recommendations

The primary objective of this research is the incorporation of AR and robot dynamics in path planning and the transformation of the planned trajectories into task-optimized executable robot paths. It aims to develop a set of methodologies to facilitate intuitive human-robot interactions in robot path planning and orientation planning of the robot EE. By incorporating various simulations in an AR environment, the users would be able to preview and evaluate the motion planned prior to the execution of a task.

7.1 Research Contributions

This thesis has made contributions in the following aspects.

- **An AR-based environment for intuitive robot programming**

An AR-based Robot Programming (RPAR-II) system is proposed to assist users in robot path planning and robot EE orientation planning incorporating robot dynamics through providing an intuitive interface to enrich the interactions between the operators and the robot. A parametric virtual robot is modeled to replicate the configurations of a real robot, and a marker-based tracking scheme using a stereo camera is adopted to register the virtual robot into the real working scene. In this system, an interaction device, which can be tracked through tracking the attached marker-cube, is employed to guide the virtual robot for intuitive

robot task planning in a safe manner. A monitor-based display scheme is employed to provide a broader FOV, to allow the users to be able to visualize the entire operating range of the robot. Detailed system architecture has been developed to describe the various functions of each module of the proposed RPAR-II system.

- **Methodologies for robot trajectory path planning incorporating robot dynamics**

The contribution of the proposed methodologies is the incorporation of AR and robot dynamics in robot programming and trajectory planning and the transformation of the planned trajectories into task-optimized executable robot paths It is computationally intensive and complex to search for a time-scale trajectory directly subject to both robot kinematics and dynamics constraints, given a pair of predefined starting and goal points. Therefore, an approach has been proposed that allows the users to create a number of control points within the CFV to form a smooth geometric path. A method for control point modification has been developed to facilitate the planning of collision-free paths as the swept model of the EE may not be within the CFV along the generated path. In this modification process, a Euclidean-distance-based method has been implemented to assist the user in selecting the point of interest in the control point deletion or insertion operation. It computes the distances between the probe of the interaction device and each of the control points in real time. The point that has the minimum distance to the probe is nominated as the candidate point, is highlighted to differentiate this from other control points. The robot path has been generated

from the set of control points through a cubic-splines interpolation scheme. Two practical issues associated with the properties of the path are discussed, namely, the number of total number of control points, and the time stamps assigned to each control point.

Once the path has been determined, time-optimal trajectory optimization driven by an objective function associated with path duration proceeds with both robot capabilities and user-defined criteria to complete the task. A convex optimization method has been implemented to obtain an approximated time-optimal trajectory, where the objective function is formulated into a convex function and joint torque constraints are reinforced as a log-barrier item to the convex function, allowing the robot system to maximize its productivity without having to engage its actuators excessively. Prior to translation into the robot controller codes, the optimized trajectory will be simulated with a virtual robot and a set of controller parameters can be adjusted interactively during the simulation. These parameters and the trajectory can then be fed to the robot controller as robot programs for execution.

- **Methodologies for orientation planning of robot EE**

The contribution of the proposed methodologies is the use of AR in planning the orientation of robot EE along a visible path complying with the robot dynamics constraints and task requirements. A pose tracked using the interaction device has been mapped to a valid pose for a Scorbot type manipulator, which has a reduced wrist configuration, through inverse kinematics techniques. An AR-based HRI

has been developed to assist users in performing a series of operations for EE orientation planning, i.e., acquisition of output curve model through human demonstrations, creation of a list of control points on the output curve, and determination of the orientations of the EE at these points. The Euclidean-distance-based method has been applied for the selection of the point of interest in the creation of a list of control points, or the modification of the list of points. A data structure is developed to group a set of parameters associated with each control point together. By defining the curve model as the directrix and the orientation of the EE as the rulings, a ruled surface can be constructed to represent a smooth orientation profile of the EE along the output curve. A similar log-barrier approximated optimization method has been implemented to transfer the geometric EE orientation profile into a time-scale trajectory, subject to the joint torque and joint velocity constraints. The case studies show successful implementation of the proposed methodologies in planning a feasible trajectory for a path following task where the orientations of the EE are within the pre-defined range of inclination angles with respect to the surface normal defined along the curve.

- **Methodologies for interactive robot trajectory simulation**

In this system, information visualization during path planning is achieved by augmenting the necessary virtual contents onto the video stream displayed on a monitor. The simulation enables the users to preview the optimization outputs, such as path duration, optimized profiles for joint velocity, joint torque, etc. A normalized measurement associated with the simulated torque and velocity of

each joint is computed and compared along the trajectory. By rendering both the planned and simulated paths to the real working environment, users are able to perceive the possible discrepancy between the two paths instantaneously. It provides useful cues in tuning the controller parameters or log-barrier parameter to achieve a more desirable trajectory complying with robot dynamics capabilities. In addition, the constraints that have been considered during trajectory optimization can be verified, and the usage of each link in the planned task can be monitored.

7.2 Recommendations

A number of areas can be further explored and developed to improve the contributions made in this research.

- **Development of a more comprehensive RPAR-II system**

In this research, a GUI for parametric robot modeling has been developed. A more powerful library of different robot models can be built with the parameters of different robot types. In addition, more suitable interaction techniques, such as voice-based or gesture-based control, can be developed to assist the users in robot programming and path planning in a more natural manner. An alternative interaction device, such as a joystick or a PHANToM, will enable the users to manipulate the virtual robot augmented on the remote site to carry out the planning tasks. It would be useful to integrate a robust robot dynamics estimation module in the proposed system, such that the robot capabilities can be fully

explored during robot trajectory optimization, and the simulation will be more meaningful and closer to the actual behavior of robot when performing the planned task.

- **Improvement on system accuracy**

One limitation of the proposed system is the low level of accuracy achievable due to the use of the ARToolKit-based tracking method. Even though a stereo camera has been used in the system, the errors in stereo depth estimation are still significant. To enhance the accuracy of this system, a more accurate and robust tracking method will be needed. Stereo cameras with unparalleled optical axes will improve the accuracy of the disparity map, yielding better performance in stereo depth estimation. For industrial robot systems which often have restricted workspaces, the combination of camera and time-of-flight sensors, such as laser tracker, will improve the tracking accuracy significantly.

- **Improvement on definition and modification of EE orientation**

According to the user study on planning the EE orientation, there are some difficulties for the users to define, or modify the orientation of the robot EE at a control point. Improvement can be made to develop an easier, more intuitive and non-distracting interface for the users performing EE orientation definition and modification. Given a cone defining the possible orientation range for the EE at a control point, one possible way is to display a 2D grid on the screen representing the base of the cone, where the grid size can be defined by the users. The EE orientation, or the angle with respect to the axis of the cone, will be transformed

into the distance between the corresponding point and the centre of the base of the cone. This will give the users a clearer perception of the planned orientation of the EE within the permissible range.

- **Extension of the current system for other types of robot tasks**

The proposed methodologies have been targeted at two main types of robot tasks, namely, pick-and-place tasks and path following tasks. The system can be extended to suit other types of tasks, such as robot painting, assembly, etc., where both pick-and-lace operations and path following operations may be required for a single task. It can be further developed to assist the users in the selection of an optimal location of a robot, or a workpiece, prior to its final installation, and the determination of a suitable robot configuration among a number of available robots.

Publications arising from this research

Journal publications:

1. H.C. Fang, S.K. Ong, and A.Y.C. Nee, Interactive trajectory planning and simulation using augmented reality, Robotics and Computer-Integrated Manufacturing, 2011, 28(2): 227–237.
2. H.C. Fang, S.K. Ong, and A.Y.C. Nee, Orientation planning of robot end-effector using augmented reality, International Journal of Advanced Manufacturing and Technology, September 2011, under review.
3. H.C. Fang, S.K. Ong, and A.Y.C. Nee, A novel AR-based interface for human-robot interaction and visualization, Robotica, April 2012, under review.

Conference publications:

1. H.C. Fang, S.K. Ong, and A.Y.C. Nee, Robot programming using augmented reality, In Proceeding of the International Conference on CyberWorlds. University of Bradford, UK; 7–11 September 2009, p.13–20.
2. H.C. Fang, S.K. Ong, and A.Y.C. Nee, Robot Path and EE Orientation Planning using Augmented Reality, 45th CIRP International Conference on Manufacturing Systems, Innovation: The Challenge for the Manufacturing of the Future. Athens, Greece; 16–18 May 2012, p.216–223.

Bibliography

Ababsa F.E., Mallem M. (2007). Hybrid three-dimensional camera pose estimation using particle filter sensor fusion. Advanced Robotics; 21(1–2): 165–81.

ABB (2011) The ABB group. http://www.abb.com/, last accessed on 14 October 2011.

ACL (1999). ACL Reference Guide, Eshed Robotec.

Aleotti J., Caselli S. (2005). Trajectory clustering and stochastic approximation for robot programming by demonstration. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada: 2–6 August 2005. p. 1029–34.

Angeles J. Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms. 3rd ed., NY: Springer; 2007.

Argall B.D., Chernova S., Veloso M., Browning B. (2009). A survey of robot learning from demonstration. Robotics and Autonomous Systems; 57(5): 469–83.

Asano T., Asano T., Guibas L., Hershberger J., Imai H. Visibility-polygon search and Euclidean shortest path. In: Proceedings of the Annual Symposium on Foundations of Computer Science. Portland, OR; 21–23 October 1985. p.155–64.

AVILUS. http://www.avilus.de/, last accessed on 14 October 2011.

Azuma, R.T. (1997). A survey of augmented reality. Presence: Teleoperators and Virtual Environments, 6(4), 355–85.

Azuma, R.T. (2001). Augmented reality: approaches and technical challenges. Fundamentals of Wearable Computers and Augmented Reality, Mahwah, NJ: Lawrence Erlbaum Associates Inc.; p. 27–63.

Bai Y., Zhuang H., Roth Z.S. (2003). Experiment study of PUMA robot calibration using a laser tracking system. IEEE International Workshop on Soft Computing in Industrial Applications. SUNY Binghamton, NY; 23–25 June 2003. p. 139–44.

Bay H., Tuytelaars T., Van Gool L. (2006). SURF: Speeded up robust features. In: Proceedings of the European Conference on Computer Vision. Graz, Austria; 7–13 May 2006. p. 404–17.

Behringer R. (1999). Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors. In: Proceedings of Annual International Symposium on Virtual Reality. Albuquerque, NM; 13–17 March 1999. p. 244–51.

Benhimane S. Malis E. (2007). Homography-based 2D visual tracking and servoing. International Journal of Robotics Research; 26(7): 661–76.

Bicchi A., Peshkin M.A., Colgate J.E. (2007). Safety for physical human-robot interaction, In: B. Siciliano, O. Khatib (Eds.), Springer Handbook of Robotics, Berlin, London: Springer; p. 1335–48.

Biggs G., MacDonald B. (2003). A survey of robot programming systems. In: Proceedings of the Australasian Conference on Robotics and Automation. Brisbane, Australia; 1–3 December 2003. p. 27–36.

Bischoff R., Kazi A., Seyfarth M. (2002).The MORPHA style guide for icon-based programming. In: Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication. Berlin, Germany; 25–27 September 2002. p. 482–7.

Bischoff R., Kazi A. (2004). Perspectives on augmented reality based human-robot interaction with industrial robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. Sendai, Japan; 28 September – 2 October 2004. p. 3226–31.

Bischoff R., Kurth J. (2006). Invited talk: concepts, tools and devices for facilitating human-robot interaction with industrial robots through augmented reality. In: ISMAR Workshop on Industrial Augmented Reality. Santa Barbara, CA; 22 October 2006. Available: http://www.ismar06.org/ data/2a-KUKA.pdf, last accessed on 14 October 2011.

Bobrow J.E., Dubowsky S., Gibson J.S. (1985). Time-optimal control of robotic manipulators along specified paths. International Journal of Robotics Research; 4(3): 3–17.

Boyd S. Vandenberghe L. (2004). Convex Optimization. University Press: Cambridge.

Breazeal C., Edsinger A., Fitzpatrick P., Scassellati B. (2001). Active vision for sociable robots. IEEE Transactions on System, Man, and Cybernetics, Part A: Systems and Humans; 31(5): 443–53.

Brogårdh, T. (2007). Present and future robot control development–an industrial perspective. Annual Reviews in Control; 31(1): 69–79.

Brooks R.A., Lozano-Perez T. (1995). A subdivision algorithm in configuration space for find-path with rotation. IEEE Transactions on Systems, Man and Cybernetics; SMC-15(2): 224–33.

Burdea, G.C. (1996). Virtual reality and robotics in medicine. IEEE International Workshop on Robot and Human Communication. Tsukuba, Japan; 11–14 November 1996. p. 16–25.

Calinon S., Guenter F., Billard A. (2007). On learning, representing and generalizing a task in a humanoid robot. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics; 37(2): 286–98.

Calonder M., Lepetit V., Strecha C., Fua P. (2010). BRIEF: binary robust independent elementary features. In: Proceedings of the European Conference on Computer Vision. Heraklion, Greece; 5–11 September 2010. p. 778–92.

Canny J.F. (1988). The Complexity of Robot Motion Planning. Cambridge, MA: MIT Press.

Castle R.O., Klein G., Murray D.W. (2008). Video-rate localization in multiple maps for wearable augmented reality. In: Proceedings of the IEEE International Symposium on Wearable Computers. Pittsburgh, PA; 28 September – 1 October 2008. p. 15–22.

Chan T.F., Dubey R.V. (1995) A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. IEEE Transactions on Robotics and Automation; 11(2): 286–92.

Chaumette F., Hutchinson S. (2006). Visual servo control, part I: basic approaches. IEEE Robotics and Automation Magazine; 13(4): 82–90.

Chaumette F., Hutchinson S. (2007). Visual servo control, part II: advanced approaches. IEEE Robotics and Automation Magazine; 14(1): 109–18.

Chen, H. Sheng W. (2011). Transformative CAD based industrial robot program generation. Robotics and Computer-Integrated Manufacturing; 27(5): 942–8.

Chen J., Zelinsky A. (2003). Programming by demonstration: coping with suboptimal teaching actions. International Journal of Robotics Research; 22(5): 299–319.

Chen P.C., Hwang Y.K. (1998). SANDROS: a dynamic graph search algorithm for motion planning. IEEE Journal of Robotics and Automation; 14(3): 390–403.

Chintamani K., Cao A., Ellis R.D., Pandya A.K. (2010). Improved tele-manipulator navigation during display-control misalignments using augmented reality cues. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans; 40(1): 29–39.

Choi B., Lee J. (2008). Localization of a mobile robot based on an ultrasonic sensor using dynamic obstacles. Artificial Life and Robotics; 12(1-2): 280–3.

Chong J.W.S., Ong S.K., Nee A.Y.C. (2007). Methodologies for immersive robot programming in an augmented reality environment. International Journal of Virtual Reality; 6(1): 69–79.

Chong J.W.S., Ong S.K., Nee A.Y.C., Youcef-Youmi K. (2009). Robot programming using augmented reality: An interactive method for planning collision-free paths. Robotics and Computer-Integrated Manufacturing; 25(3): 689–701.

Collett T.H.J., MacDonald B.A. (2006). Developer oriented visualization of a robot program: An augmented reality approach. In: Proceedings of the ACM Conference on Human-Robot Interaction—Toward Human Robot Collaboration. Salt Lake City, UT; 2–4 March 2006. p. 49–56.

Constantinescu D., Croft E.A. (2000). Smooth and time-optimal trajectory planning for industrial manipulators along specified paths. Journal of Robotic Systems; 17(5): 233–49.

Craig J.J. Introduction to Robotics, Mechanics and Control. 3rd ed., NY: Pearson Education Inc.; 2005.

De Figueiredo, L. H. (1995). Adaptive sampling of parametric curves, In: V, A. Paeth, (Eds), Graphics Gems, New York: Academic Press, Inc.; p.173–8.

DeSouza G.N., Kak A.C. (2002). Vision for mobile robot navigation: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence; 24(2): 237-67.

Duleba I. (1997). Minimum cost, fixed time trajectory planning in robot manipulators. A suboptimal solution. Robotica; 15(5): 555–62.

Enderle, S. (2009). Grape—Graphical robot programming for beginners. Communications in Computer and Information Science; 33: 180–92.

Faverjon B. (1984). Obstacle avoidance using an Octree in the configuration space of a manipulator. In: Proceedings of the IEEE International Conference on Robotics and Automation. Atlanta, GA; 13–15 March 1984. p. 504–12.

Faverjon B., Tournassoud P. (1987). The mixed approach for motion planning: learning global strategies form a local planner. In: Proceedings of the

International Joint Conference on Artificial Intelligence. Milan, Italy; 23–28 August 1987. p.1131–7.

Fong W.T., Ong S.K., Nee A.Y.C. (2008). Methods for in-field user calibration of inertial measurement unit without external equipment. Measurement Science and Technology; 19(8): 085202.

Fong W.T., Ong S.K., Nee A.Y.C. (2009). Computer vision centric hybrid tracking for augmented reality in outdoor urban environments. In: Proceedings of the Symposium on Virtual Reality Software and Technology. Yokohama, Japan; 14–15 December 2009. p. 185–90.

Foxlin E., Naimark L. (2002). Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality. Darmstadt, Germany; 30 September – 1 October 2002. p. 27–36.

Foxlin E., Naimark L. (2003). Miniaturization, calibration and accuracy evaluation of a hybrid self-tracker. In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality. Tokyo, Japan; 7–10 October 2003. p. 151–60.

Friedrich W. (2002). ARVIKA—Augmented reality for development, production and service. In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality. Darmstadt, Germany; 30 September – 1 October 2002. p. 3–4.

Friedrich H., Hofmann H., Dillmann R. (1997). 3D-icon based user interaction for robot programming by demonstration. In: Proceedings of the IEEE

International Symposium on Computational Intelligence in Robotics and Automation. Monterey, CA; 10–11 July 1997. p. 240–5.

Garlan D., Siewiorek D., Smailagic A., Steenkiste P. (2002). Project Aura: toward distraction-free pervasive computing. IEEE Pervasive Computing; 1(2): 22–31.

Gausemeier J., Fruend J., Matysczok C. (2002). AR-planning tool—designing flexible manufacturing systems with Augmented Reality, In: Proceedings of the Eurographics Workshop on Virtual Environments. Barcelona, Spain; 30–31 May 2002. p. 19–25.

Giesler B., Steinhaus P., Walther M., Dillmann R. (2004). Sharing skills: using augmented reality for human-robot collaboration. In: Proceedings of the SPIE-The International Society for Optical Engineering. Denver, CO; 2–6 August 2004. 5291(1): 446–53.

Gnuplot: http://www.gnuplot.info, last accessed on 14 October 2011.

Goodrich M.A., Schultz A.C. (2007). Human-robot interaction: a survey. Foundations and Trends in Human-Computer Interaction; 1(3): 203–75.

Haller M., Brandl P., Leithinger D., Leitner J., Seifried T., Billinghurst M. (2006). Shared design space: Sketching ideas using digital pens and a large augmented tabletop setup. In: Proceedings of the International Conference on Artificial Reality and Telexistence. Hangzhou, China; 29 November – 1 December 2006. p. 185–96.

Hamada K., Hori Y. (1996). Octree-based approach to real-time collision-free path planning for robot manipulator. In: Proceedings of the ACM

International Workshop on Advanced Motion Control. Tsukuba-City, Mie-Pref., Japan; 18–21 March 1996. p. 705–10.

Hattori A., Suzuki N., Hashizume M., Akahoshi T., Konishi K., Yamaguchi S., Shimada M., Hayashibe M. (2003). A robotic surgery system (da Vinci) with image guided function–system architecture and cholecystectomy application. Medicine Meets Virtual Reality; 11: 110–6.

He X.J., Chen Y.H. (2009). Haptic-aided robot path planning based on virtual tele-operation. Robotics and Computer-Integrated Manufacturing; 25(4–5): 792–803.

Henderson S., Feiner S. (2009). Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality. Orlando, FL; 19–23 October 2009. p. 135–44.

Hollerer T.H., Feiner S., Terauchi T., Rashid G., Hallaway D. (1999). Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system. Computers & Graphics; 23(6): 779–85.

Hollmann R., Hägele M., Verl A. (2010). Learning probabilistic models to enhance the efficiency of programming-by-demonstration for industrial robots. In: Proceedings of the International Symposium on Robotics. Munich, Germany; 7–9 June 2010. p. 1–7.

Hopler R., Otter M. (2001). A versatile C++ toolbox for model based, real time control systems of robotic manipulators. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. Maui, Hawaii; 29 October – 3 November 2001. p. 2208–14.

Hoznek A., Zaki S.K., Samadi D.B., Salomon L., Lobontiu A., Lang P., Abbou C.C. (2002). Robotic assisted kidney transplantation: An initial experience. Journal of Urology; 167(4): 1604–6.

Huang Y.K. Ahuja N. (1992). Gross motion planning—a survey. ACM Computing Survey; 24(3): 219–92.

Hudak P., Courtney A., Nilsson H., Peterson J. (2003). Arrows, robots, and functional reactive programming. Lecture Notes in Computer Science; 2638: 159–87.

Hudson T., Lin M.C., Cohen J.D., Gottschalk S., Manocha D. (1997) V-COLLIDE: accelerated collision detection for VRML. In: Proceedings of the ACM Symposium on Virtual Reality Modeling Language. Monterey, CA; 24–26 February 1997. p.119–25.

IFR service robot. (2011). IFR service robot. Available: http://www.ifr.org/service-robots/, last accessed on 14 October 2011.

IFR world robot. (2011). Exclusive summary of world robotics. Available: http://www.worldrobotics.org/uploads/media/2011_Executive_Summary.pdf, last accessed on 14 October 2011.

Jun S., Shin K.G. (1988) A probabilistic approach to collision-free robot path planning. In: Proceedings of the IEEE International Conference on Robotics and Automation. Philadelphia, NY; 24–29 April 1988. p. 220–7.

Kaiser M., Retey A., Dillmann R. (1995). Robot skill acquisition via human demonstration. In: Proceedings of the International Conference on Advanced Robotics. Sant Feliu de Guixols, Spain; September 1995. p. 763–8.

Kambhampati S., Davis L.S. (1986). Multi-resolution path planning for mobile robots. IEEE Journal of Robotics and Automation; RA-2(3): 135–45.

Kanayama Y.J., Wu C.T. (2000). It's time to make mobile robots programmable. In: Proceedings of the IEEE International Conference on Robotics and Automation. San Francisco, CA; 24–28 April 2000. p. 329–34.

Kato H., Billinghurst M. (1999). Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: Proceedings of the IEEE International Workshop on Augmented Reality. San Francisco, CA; 21–21 October 1999. p. 85–94.

Khatib O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation. St. Louis, Missouri, NY; 25–28 March 1985. p. 500–5.

Kim S., Jung S.H., Kim C.H. (1999). Preventive maintenance and remote inspection of nuclear power plants using tele-robotics. In: Proceedings of International Conference on Intelligent Robots and Systems. Kyongju, South Korea; 17–21 October 1999. p. 603–8.

Kiyokawa, K. (2007). An introduction to head mounted displays for augmented reality. Emerging Technologies of Augmented Reality: Interfaces and Design. Idea Group Publishing. p. 43–63.

Klein A. (1987). CAD-based off-line programming of painting robots. Robotica; 5(4): 267–71.

Klein G., Murray D. (2007). Parallel tracking and mapping for small AR workspaces. In: Proceedings of the IEEE and ACM International Symposium

on Mixed and Augmented Reality. Nara, Japan; 13–16 November 2007. p. 225–34.

Klein G., Murray D. (2008). Improving the agility of keyframe-based SLAM. In: Proceedings of the European Conference on Computer Vision. Marseille, France; 12–18 October 2008. p. 802–15.

Kuang W.T., Morris A.S. (1999). Algorithm for robot position tracking using ultrasonics. Electronics Letters; 35(1): 87–8.

Kuang W.T., Morris A.S. (2000a). Ultrasound speed compensation in an ultrasonic robot tracking system. Robotica; 18(6): 633–7.

Kuang W.T., Morris A.S. (2000b). Combined time-of-flight and Doppler ultrasonic tracking system for better performance in robot tracking. In: IEEE Proceedings–Science, Measurement and Technology. Orlando, FL; 12 April 2000. 147(5): 213–8.

KUKA (2011). KUKA automatisering + robots N.V. http://www.kuka.be/, last accessed on 14 October 2011.

Latombe, J.-C. (1991). Robot Motion Planning. Dordrecht: Kluwer Academic.

Lee G.A., Kang H., Son W. (2008). MIRAGE: A touch screen based mixed reality interface for space planning applications. In: Proceedings of the IEEE Virtual Reality Conference, Reno, NV; 8–12 March 2008. p. 273–4.

Lee J. (1995). Dynamic programming approach to near minimum-time trajectory planning for two robots. IEEE Transactions on Robotics and Automation; 11(1): 160–4.

Lego (2011). Lego Mindstorms NXT 2.0, http://mindstorms.lego.com/ en-us/products/8547.aspx; last accessed on 14 October 2011.

Lieberknecht S., Benhimane S., Meier P., Navab N. (2009). A dataset and evaluation methodology for template-based tracking algorithms. In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality. Orlando, FL; 19–23 October 2009. p. 145–51.

Lin C.C., Chuang J.H. (2003). Potential-based path planning for robot manipulators in 3D workspace. In: Proceedings of the IEEE International Conference on Robotics and Automation. Taipei, Taiwan; 14–19 September 2003. p. 3353-8.

Lin C.S., Chang P.R., Luh J.Y.S. (1983). Formulation and optimization of cubic polynomial joint trajectories for industrial robots. IEEE Transactions on Automation and Control; 28(12): 1066–74.

Lin H.H., Tsai C.C., Hsu J.C. (2008). Ultrasonic localization and pose tracking of an autonomous mobile robot via fuzzy adaptive extended information filtering. IEEE Transactions on Instrumentation and Measurement; 57(9): 2024–34.

Livatino S., Muscato G., De Tommaso D., Macaluso M. (2010). Augmented reality stereoscopic visualization for intuitive robot teleguide. In: Proceedings of the IEEE International Symposium on Industrial Electronics. Bari, Italy; 4–7 July 2010. p. 2828–33.

Lowe D.G. (2004). Distinctive image features from scale-invariant key-points. International Journal of Computer Vision; 60(2): 91–110.

Lozano-Perez T. (1983a). Robot programming. In: Proceedings of the IEEE; 71(7): 821–41.

Lozano-Perez T. (1983b). Spatial planning: a configuration space approach. IEEE Transactions on Computers; C-32: 108–120.

Lozano-Perez T. (1987). A simple motion planning algorithm for general robot manipulators. IEEE Journal of Robotics and Automation; 3(3): 224–38.

Mahajan A., Figueroa F. (1997). Intelligent seam tracking using ultrasonic sensors for robotic welding. Robotica; 15(3): 275–81.

Marin R., Sanz P.J., Nebot P., Wirz R. (2005). A multimodal interface to control a robot arm via the web: a case study on remote programming. IEEE Transactions on Industrial Electronics; 52(6): 1506–20.

Meeussen W. (2006). Compliant robot motion: from path planning or human demonstration to force controlled task execution. Ph.D. Thesis, Katholieke Universiteit Leuven.

Montgomery M., Gaw D., Meystel A. (1987). Navigation algorithm for a nested hierarchical system of robot path planning among polyhedral obstacles. In: Proceedings of the IEEE International Conference on Robotics and Automation. Raleigh, NC; 31 March – 3 April 1987. p. 1612–22.

Natonek E., Zimmerman T., Fluckiger L. (1995). Model based vision as feedback for virtual reality robotics environment. In: Proceedings of the IEEE Virtual Reality Annual International Symposium. Research Triangle Park, NC; 11–15 March 1995. p. 110–7.

Navon, R. and A.Retik (1997). Programming construction robots using virtual reality techniques. Automation in Construction, 5(5), 393–406.

Nawab A., Chintamani K., Ellis D., Auner G., Pandya A. (2007). Joystick mapped augmented reality cues for end-effector controlled tele-operated robots. In:

Proceedings of the IEEE Virtual Reality Conference. Charlotte, NC; 10–14 March 2007. p. 263–6.

Naylor A., Shao L., Volz R., Jungelas R., Bixel P., Lloyd K. (1987). PROGRESS—a graphical robot programming system. In: Proceedings of the IEEE International Conference on Robotics and Automation, Raleigh, NC, USA: 31 March – 3 April 1987, p. 1282–1991.

Neto P., Pires J.N., Moreira A.P. (2010a). Robot path simulation: a low cost solution based on CAD. In: Proceedings of the IEEE International Conference on Robotics, Automation and Mechatronics. Singapore; 28–30 June 2010. p. 333–8.

Neto P., Pires J.N.; Moreira A.P. (2010b). High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition. Industrial Robot; 37(2): 137–47.

Newcombe R.A., Davison A.J. (2010). Live dense reconstruction with a single moving camera. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. San Francisco, CA; 13–18 June 2010. p. 1498–505.

Ng C.L., Ng T.C., Nguyen T.A.N., Yang G., Chen W. (2010). Intuitive robot tool path teaching using laser and camera in augmented reality environment. In: Proceedings of the International Conference on Control, Automation, Robotics and Vision. Singapore; 7–10 December 2010. p. 114–9.

Olwal A., Gustafsson J., Lindfors C. (2008). Spatial augmented reality on industrial CNC-machines. In: Proceedings of SPIE – The International Society for Optical Engineering. San Jose, CA; 31 January 2008. p. 680409-1-9.

Ong S.K., Chong J.W.S., Nee A.Y.C. (2010). A novel AR-based robot programming and path planning methodology. Robotics and Computer-Integrated Manufacturing; 26(3): 240–9.

Ong, S.K., Z.B. Wang (2011). Augmented assembly technologies based on 3D bare-hand interaction. CIRP Annals–Manufacturing Technology, 60(1), 1–4.

Ong S.K., Yuan M.L., Nee A.Y.C. (2008). Augmented reality applications in manufacturing: a survey. International Journal of Production Research; 46(10): 2702–42.

OTM (1999). OSHA technical manual, Section IV, Chapter 4: Industrial robots and robot system safety. Available: http://www.osha.gov/dts/osta/otm/otm_toc.html, last accessed on 14 October 2011.

Owen C., Tang A., Xiao F. (2003). ImageTclAR: a blended script and compiled code development system for augmented reality. In: Proceedings of the International Workshop on Software Technology for Augmented Reality Systems. Tokyo, Japan; 7 October 2003. p. 23–8.

Özuysal M., Fua P., Lepetit V. (2007). Fast Keypoint Recognition in Ten Lines of Code. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. Minneapolis, MN; 17–22 June 2007. p. 1–8.

Pan Q., Reitmayr G., Drummond T. (2009). ProFORMA: probabilistic feature-based on-line rapid model acquisition. In: Proceedings of the British Machine Vision Conference. London, UK; 7–10 September 2009. p. 1–11.

Parasuraman R., Sheridan T.B., Wickens C.D. (2000). A model for types and levels of human interaction with automation. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans; 30(3): 286–97.

Pyro (2011). Python Robotics. http://www.pyrorobotics.org/, last accessed on 26 June 2011.

Rastogi A., Milgram P., Drascic D. (1996). Telerobotic control with stereoscopic augmented reality. In: Proceedings of the SPIE – Stereoscopic Displays and Virtual Reality Systems III. San Jose, CA; 30 January – 2 February 1996. p. 115–22.

Reinhart G., Munzert U., Vogl W. (2008). A programming system for robot-based remote-laser-welding with conventional optics. CIRP Annals—Manufacturing Technology; 57(1): 37–40.

Reinhart G., Patron C. (2003). Integrating augmented reality in the assembly domain – fundamentals, benefits and applications. CIRP Annals—Manufacturing Technology; 52(1): 5–8.

Reitmayr G., Schmalstieg D. (2001). OpenTracker – an open software architecture for reconfigurable tracking based on XML. In: Proceedings of the IEEE Virtual Reality. Yokohama, Japan; 13–17 March 2001. p. 285–6.

Reitmayr G., Drummond T. (2006). Going out: Robust model-based tracking for outdoor augmented reality. In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality. Santa Barbara, CA; 22–25 October 2006. p. 109–18.

Ribo M., Brandner M., Pinz A. (2004). A flexible software architecture for hybrid tracking. Journal of Robotic Systems; 21(2): 53–62.

Roboop: http://www.cours.polymtl.ca/roboop, last accessed on 14 October 2011.

Rolland J.P., Davis L.D., Baillot Y. (2001). A survey of tracking technology for virtual environments. Fundamentals of Wearable Computers and Augmented Reality. Mahwah, NJ: Lawrence Erlbaum Associates Inc.; p. 67–112.

Schall, G., E. Mendez, E. Kruijff, E. Veas, S. Junghanns, B. Reitinger, and D. Schmalstieg (2009). Handheld Augmented Reality for underground infrastructure visualization. Personal and Ubiquitous Computing; 13(4): 281–91.

Schmalstieg D., Fuhrmann A., Hesina G., Szalavari Z., Encarnacao L.M., Gervautz M., Purgathofer W. (2002). The Studierstube augmented reality project. Presence; 11(1): 33–54.

Schraft R.D., Meyer C. (2006). The need for an intuitive teaching method for small and medium enterprises. VDI Berichte; 1956: 95.

Schweighofer G., Pinz A. (2006). Robust pose estimation from a planar target. IEEE Transactions on Pattern Analysis and Machine Intelligence; 28(12): 2024–30.

Satoh K., Takemoto K., Uchiyama S., Yamamoto H. (2006). A registration evaluation system using an industrial robot. In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality. Santa Barbara, CA; 22–25 October 2006. p. 79–87.

Shen Y., Ong S.K., Nee A.Y.C. (2009). A mixed reality environment for collaborative product design and development. CIRP Annals–Manufacturing Technology; 58(1): 139–42.

Shen Y., Ong S.K., Nee A.Y.C. (2010). Augmented reality for collaborative product design and development. Design Studies; 31(2): 118–45.

Sheng W., Xi N., Song M., Chen Y., MacNeille P. (2000). Automated CAD-guided robot path planning for spray painting of compound surfaces. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. Takamatsu, Japan; 31 October – 5 November 2000. p. 1918–23.

Sheng W., Chen H., Xi N., Chen Y. (2005). Tool path planning for compound surfaces in spray forming processes. IEEE Transactions on Automation and Engineering; 2(3): 240–9.

Shiller Z. (1994). On singular time-optimal control along specified paths. IEEE Transactions on Robot Automation; 10(4): 56 –71.

Shimizu N., Koizumi N., Sugimoto M., Nii H., Sekiguchi D., Inami M. (2006). Teddy-bear based robotic user interface. ACM Computers in Entertainment; 4(3): 1–14.

Shin K., Mckay N. (1986). A dynamic programming approach to trajectory planning of robotic manipulators. IEEE transactions on automatic control; 31(6): 491–500.

Simon D.A. (1997). Intra-operative position sensing and tracking devices. In: Proceedings of the First Joint CVRMed / MRCAS Conference. Grenoble, France; 20–22 March 1997. p. 62–4.

Simon D.A., Hebert M., Kanade T. (1995). Techniques for fast and accurate intra-surgical registration. Journal of Image Guided Surgery; 1(1): 17–29.

Simon D.A., Kanade T. (1997). Geometric constraint analysis and synthesis: methods for improving shape-based registration accuracy. In: Proceedings of the First Joint CVRMed /MRCAS Conference. Grenoble, France; 20–22 March 1997. p. 181–90.

SMErobot (2009). The European robot initiative for strengthening the competitiveness of SMEs in manufacturing: The robot capable of understanding human-like instructions. http://www.smerobot.org/15_final_workshop/download/presentations/02_New_devices_and_methods_20090507.pdf, last accessed on 14 October 2011.

Steinfeld A., Fong T., Kaber D., Lewis M., Scholtz J., Schultz A., Goodrich M. (2006) Common metrics for human-robot interaction. In: Proceeding of the ACM International Conference on Human Robot Interaction. Salt Lake City, UT; 2–3 March 2006. p. 33–40.

Strobl K.H., Hirzinger G. (2008) More accurate camera and hand-eye calibrations with unknown grid pattern dimensions. In: Proceedings of the IEEE International Conference on Robotics and Automation. Pasadena, CA; 19–23 May 2008. p. 1398-405.

Tachi S., Kawakami N., Inami M., Zaitsu Y. (2004) Mutual telexistence system using retro-reflective projection technology. International Journal of Humanoid Robotics; 1(1): 45–64.

Tadakuma R., Asahara Y., Kajimoto H., Kawakami N., Tachi S. (2005) Development of anthropomorphic multi-DOF master-slave arm for mutual telexistence. IEEE Transactions on Visualization and Computer Graphics; 11(6): 626–36.

Tewari A., Peabody J., Sarle R, Balakrishnan G., Hemal A., Shrivastava A., Menon M. (2002) Technique of da Vinci robot-assisted anatomic radical prostatectomy. Urology; 60(4): 569–72.

Thrun S. (2004) Toward a framework for human-robot interaction. Human-Computer Interaction; 19(1/2): 9–24.

Tsai M.J., Stone D.-J. (2009) Inverse velocity analysis for line guidance five-axis robots. Robotics and Computer-Integrated Manufacturing; 25 (4–5): 736–45.

V-COLLIDE: http://gamma.cs.unc.edu/V-COLLIDE/, last access date: 14 October 2010.

Van Den Hengel A., Dick A., Thormählen T., Ward B., Torr P.H.S. (2009) VideoTrace: Rapid interactive scene modeling from video. ACM Transactions on Graphics; 26(3): 86-1-5.

Verscheure D., Demeulenaere B., Swevers J., De Schutter J., Diehl M. (2008) Time-energy optimal path tracking for robots: a numerically efficient optimization approach. In: Proceedings of the IEEE International Workshop on Advanced Motion Control. Trento, Italy; 26–28 March 2008. p. 727–32.

Verscheure D., Diehl M., De Schutter J., Swevers J. (2009) On-line time-optimal path tracking for robots. In: Proceedings of the IEEE International Conference of Robotics and Automation. Kobe, Japan; 12–17 May 2009. p. 599–605.

Wagner D., Schmalstieg D. (2007) ARToolKitPlus for Pose Tracking on Mobile Devices. In: Proceedings of the Computer Vision Winter Workshop. St. Lambrecht, Austria; 6–8 February 2007. p. 139–46.

Wagner D., Reitmayr G., Mulloni A., Drummond T., Schmalstieg D. (2008) Pose tracking from natural features on mobile phones. In: Proceedings of the IEEE

and ACM Symposium on Mixed and Augmented Reality. Cambridge, UK; 14–19 September 2008. p. 125–34.

Warren C.W. (1989) Global path planning using artificial potential fields. In: Proceedings of the IEEE International Conference on Robotics and Automation, Scottsdale, Ariz, NY; 14–19 May 1989. p. 316–21.

Weisstein, E.W. (1997) Ruled surface. http://mathworld.wolfram.com/ RuledSurface.html, last accessed on 27th April 2011.

Wirz R., Marin R. (2004) Remote programming of an Internet tele-lab for learning visual servoing techniques: a case study. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. The Hague, Netherlands; 10–13 October 2004. p. 4414–9.

Xin M., Sharlin E., Sousa M.C. (2008) Napkin sketch—handheld mixed reality 3D sketching. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology. Bordeaux, France; 27–29 October 2008. p. 223–6.

You S., Neumann U., Azuma R. (1999) Hybrid inertial and vision tracking for augmented reality registration. In: Proceedings of the IEEE Virtual Reality Conference. Houston, TX; 13–17 March 1999. p. 260–7.

Yuan M.L., Ong S.K., Nee A.Y.C. (2006) Registration using natural features for augmented reality systems. IEEE Transactions on Visualization and Computer Graphics; 12(4): 569–80.

Yuan M.L., Ong S.K., Nee A.Y.C. (2008) Augmented reality for assembly guidance using a virtual interactive tool. International Journal of Production Research; 46(7): 1745–67.

Yue S., Henrich D., Xu W.L., Tso S.K. (2002) Point-to-point trajectory planning of flexible redundant robot manipulators using genetic algorithms. Robotica; 20(3): 269–80.

Yun W.M., Xi Y.G. (1996) Optimum motion planning in joint space for robots using genetic algorithms. Robotics and Autonomous Systems; 18(4): 373–93.

Zaeh M.F., Vogl W. (2006) Interactive laser-projection for programming industrial robots, In: Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality. Santa Barbara, CA; 22–25 October 2006. p. 125–8.

Zha X.F. (2002) Optimal pose trajectory planning for robot manipulators. Mechanism and Machine Theory; 37(10): 1063–86.

Zhang J., Ong S.K., Nee A.Y.C. (2010) A multi-regional computation scheme in an AR-assisted in situ CNC simulation environment. Computer-Aided Design; 42(12): 1167–77.

Zhu J., Ong S.K., Nee A.Y.C. (2011) Online authoring for augmented reality remote maintenance, In: Proceedings of the International Conference on Computer Graphics and Imaging. Innsbruck, Austria; 16–18 February 2011. p. 87–94.

Žlajpah L. (2008) Simulation in robotics. Mathematics and Computers in Simulation; 79(4): 879–97.

Zollner R., Asfour T., Dillmann R. (2004) Programming by demonstration: dual-arm manipulation tasks for humanoid robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems. Sendai, Japan; 28 September – 2 October 2004. p. 479–84.

Appendix A. Clamped Cubic Splines Interpolation

Given N control points in the Cartesian space $P_k\left(t_k, p\left(x_k, y_k, z_k\right)\right)$, three cubic spline functions, $sx(t), sy(t), sz(t)$, each corresponds to one dimension of the curve (i.e., in ***X***, ***Y***, and ***Z*** coordinate), can be determined. Each function is defined piecewise by $N-1$ cubic polynomials joined at the points P_k, such that $sx\left(t_k\right)=x_k$, $sy\left(t_k\right)=y_k$, $sz\left(t_k\right)=z_k$. Since the form of each function is similar, one can have an uniform representation of the cubic splines with respect to each dimension, denoted by $s(t)$. Without loss the generality, hereafter the focus is on the determination of the cubic spline function with respect to one dimension, e.g., in ***X*** coordinate.

Let $P_k\left(t_k, p_k\right)$ and $P_{k+1}\left(t_{k+1}, p_{k+1}\right)$ be two consecutive control points. The *k*th cubic polynomial $S_k(t)$ between those points is assumed to be given in Equation (A.1):

$$s_k(t)=A_k\left(t-t_k\right)^3+B_k\left(t-t_k\right)^2+C_k\left(t-t_k\right)+D_k; t\in\left[t_k, t_{k+1}\right] \quad \text{(A.1)}$$

Thus, in the spline $s(t)$, there are totally $4(N-1)$ coefficients A_k, B_k, C_k, D_k for $k=1,2,...,N-1$ that are to be determined. Given that the spline has continuous derivatives at interior control points up to the second order, there are

totally $4(N-1)-2$ equations. Therefore, two additional constraints are required to determine the set of coefficients uniquely. For a clamped spline, the two constraints are posed at the two end control points to define the tangent vectors at these two points.

Define an *N*-dimensional vector whose *k*th element is s_k as $s=[s_1,s_2,...,s_N]^T$; similarly, we have $s''=\left[s_1'',s_2'',...,s_N''\right]^T$ is the vector of second-order derivatives of the cubic function at all the control points. By applying the constraints of continuous derivatives, the relationship between s and s'' can be given in vector form, as Equation (A.2) (Angeles 2007, Chapter 6), where $\boldsymbol{A}$ and $\boldsymbol{C}$ are $(N-2)\times N$ matrices given as Equations (A.3) and (A.4), in which $\alpha_k=\Delta t_k=t_{k+1}-t_k$, $\alpha_{i,j}=\alpha_i+\alpha_j$; $\beta_k=1/\alpha_k$, $\beta_{i,j}=\beta_i-\beta_j$. s'' can be solved through solving the linear system given in Equation (A.2).

$$As''=6Cs \tag{A.2}$$

$$A=\begin{bmatrix} \alpha_1 & 2\alpha_{1,2} & \alpha_2 & & & & \\ & \alpha_2 & 2\alpha_{2,3} & \alpha_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \alpha_{N-3} & 2\alpha_{N-3,N-2} & \alpha_{N-2} & \\ & & & & \alpha_{N-2} & 2\alpha_{N-2,N-1} & \alpha_{N-1} \end{bmatrix}_{(N-2)\times N} \tag{A.3}$$

$$C=\begin{bmatrix} \beta_1 & -\beta_{1,2} & \beta_2 & & & & \\ & \beta_2 & -\beta_{2,3} & \beta_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \beta_{N-3} & -\beta_{N-3,N-2} & \beta_{N-2} & \\ & & & & \beta_{N-2} & -\beta_{N-2,N-1} & \beta_{N-1} \end{bmatrix}_{(N-2)\times N} \tag{A.4}$$

In the case of natural boundary conditions, $s_1''=s_N''=0$. Therefore, the matrix $\boldsymbol{A}$ can be reduced to a $(N-2)\times(N-2)$ tri-diagonal matrix, which can be used to solve s_2'' to s_{N-1}''.

In the case of clamped boundary conditions, $s_1'=h_1$ and $s_N'=h_2$ are the two user-defined constants. Therefore the matrix $\boldsymbol{A}$ can be represented as Equation (A.5), and the right-hand side of Equation (A.2) is given in Equation (A.6). If $h_1=h_2=0$, the matrix $\boldsymbol{C}$ can be given as Equation (A.7).

$$A_{clamped}=\begin{bmatrix} 2\alpha_1 & \alpha_1 & & & & & \\ \alpha_1 & 2\alpha_{1,2} & \alpha_2 & & & & \\ & \alpha_2 & 2\alpha_{2,3} & \alpha_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \alpha_{N-3} & 2\alpha_{N-3,N-2} & \alpha_{N-2} & \\ & & & & \alpha_{N-2} & 2\alpha_{N-2,N-1} & \alpha_{N-1} \\ & & & & & \alpha_{N-1} & 2\alpha_{N-1} \end{bmatrix}_{N\times N} \tag{A.5}$$

$$6Cs_{clamped}=\begin{bmatrix} 3\left(\beta_1(s_2-s_1)-h_1\right) \\ 3\left(\beta_2(s_3-s_2)-\beta_1(s_2-s_1)\right) \\ \vdots \\ 3\left(\beta_{N-1}(s_N-s_{N-1})-\beta_{N-2}(s_{N-1}-s_{N-2})\right) \\ 3\left(h_2-\beta_{N-1}(s_N-s_{N-1})\right) \end{bmatrix}_{N\times 1} \tag{A.6}$$

$$C_{clamped} = \begin{bmatrix} -\beta_1 & \beta_1 & & & & & & \\ \beta_1 & -\beta_{1,2} & \beta_2 & & & & & \\ & \beta_2 & -\beta_{2,3} & \beta_3 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \beta_{N-3} & -\beta_{N-3,N-2} & \beta_{N-2} & & \\ & & & & \beta_{N-2} & -\beta_{N-2,N-1} & \beta_{N-1} & \\ & & & & & \beta_{N-1} & -\beta_{N-1} & \end{bmatrix}_{N\times N} \tag{A.7}$$

Similarly, Equations (A.2) – (A.7) can also be applied to ***Y*** and ***Z*** coordinates respectively.

Appendix B. Log-Barrier Method for Solving Trajectory Optimization Problem

A.1 Optimization Problem Formulation

According to Section 4.4.4.1, the log-barrier approximated formulation for robot trajectory optimization subject to robot joint torque and joint velocity constraints can be represented in the form as Equation (B.1) (Verscheure *et al*. 2008).

$$\Omega = \min_{\mathbf{b}} . \sum_{i=0}^{N_s-1} \left(\frac{2\Delta s^i}{\sqrt{b^{i+1}} + \sqrt{b^i}} + \frac{1}{N_s} \cdot \left(\kappa^\tau \cdot P^\tau \left(\tau \left(b^i, b^{i+1} \right) \right) + \kappa^v \cdot P^v \left(\dot{\mathrm{q}} \left(b^i, b^{i+1} \right) \right) \right) \right) \quad \text{(B.1)}$$

$$P^\tau \left(\tau \right) = \frac{1}{2 \cdot ndof} \sum_{j=1}^{ndof} \left(\log \left(\frac{\left(\overline{\tau}_j \right)^2}{\left(\overline{\tau}_j \right)^2 - \left(\tau_j \right)^2} \right) \right) \quad \text{(B.2)}$$

$$P^v \left(\dot{\mathrm{q}} \right) = \frac{1}{2 \cdot ndof} \sum_{j=1}^{ndof} \left(\log \left(\frac{\left(\overline{\dot{q}}_j \right)^2}{\left(\overline{\dot{q}}_j \right)^2 - \left(q'_j \right)^2 b(s)} \right) \right) \quad \text{(B.3)}$$

In Equation (B.1), Ω represents the objective associated with the optimal solution determined; b is the optimization variable, which is represented by a vector with $(N_s + 1)$ elements, where N_s is the total number of path segments; Δs^i is the length of normalized (discrete) path segment, where $\Delta s^i = 1/N_s$; $\tau \left(b^i, b^{i+1} \right)$ and $\dot{\mathrm{q}} \left(b^i, b^{i+1} \right)$ are the joint torque vector and joint velocity vector evaluated at the midpoint of the path segment $\left(s^i, s^{i+1} \right)$; $P^\tau \left(\tau \right)$ and $P^v \left(\dot{\mathrm{q}} \right)$, as

given by Equations (B.2) and (B.3), are two log-barrier terms converted from the joint torque and joint velocity constraints respectively, in the case $\bar{\tau} = -\underline{\tau}$, in which $\bar{\tau}$ and $\underline{\tau}$ represent the upper and lower bounds of the joint torque, and $\bar{\dot{\mathrm{q}}} = -\underline{\dot{\mathrm{q}}}$, in which $\bar{\dot{\mathrm{q}}}$ and $\underline{\dot{\mathrm{q}}}$ represent the upper and lower bounds of the joint velocity. κ^{τ} and κ^{v} are log-barrier parameters for the joint torque and joint velocity constraints respectively, where $\kappa^{\tau} \geq 0$ and $\kappa^{v} \geq 0$.

A.2 Numerical Procedure for Optimization Implementation

As the objective function given in Equation (B.1) is convex, its globally optimal solution can be solved efficiently by solving the first-order optimality conditions using the Newton method. The input and output of the formulation are given in Table B-1 and Table B-2, where $q'(s) = \partial q(s)/\partial s$ and $q''(s) = \partial q^2(s)/\partial s^2$ have been evaluated when the path coordinate is converted from time ***t*** to arc length ***s***; $W_{tau}(j,j) = 1/\left(\bar{\tau}_j\right)^2$ for $j = 1, 2, \ldots ndof$, a *ndof* by *ndof* diagonal matrix associated with the bound for joint torque limits. The first-order optimality conditions of Equation (B.1) is

$$\mathbf{J}(\mathbf{b}) = \frac{\partial \Omega}{\partial \mathbf{b}} = 0 \tag{B.4}$$

To solve equation (B.4), for each iteration, we aim to solve a series of linear equations, which are the second-order Taylor approximation of the objective

function, as a function of the Newton step (or search direction) $\Delta\mathbf{b}_{(j)}$, as shown in Equation (B.5), in which $\mathbf{J}\left(\mathbf{b}_{(j)}\right)$ and $\mathbf{H}\left(\mathbf{b}_{(j)}\right)$ denote the gradient and Hessian and of the objective function at current iterate $\mathbf{b}_{(j)}$.

$$\mathbf{J}\left(\mathbf{b}_{(j)}\right)+\mathbf{H}\left(\mathbf{b}_{(j)}\right)\cdot\Delta\mathbf{b}_{(j)}=\mathbf{0} \tag{B.5}$$

Table B-1: Input parameters for trajectory planning using the log-barrier method

	Inputs
N_t	Total number of path segments
Δs^i	Discrete path segment, here it is equally spaced, $i=0,1,2,...,N_s-1$
κ^τ	Log-barrier parameter for joint torque constraint, $\kappa^\tau \geq 0$
κ^v	Log-barrier parameter for joint velocity constraint, $\kappa^v \geq 0$
ξ	Tolerance for the convergence of numerical solution (threshold for the Norm of the Jacobian matrix of the optimziation problem)
δ	Step size in backtracking line search
η	Parameter for adjusting step size δ, $\eta \in (0,1)$
γ	Parameter for adjusting the convergence of merit function during backtracking line search
$q(s)$	Joint angle representation with respect to path coordinate *s*
$q'(s)$	First-order derivative of joint angle with respect to path coordinate *s*
$q''(s)$	Second-order derivative of joint angle with respect to path coordinate *s*
$\tau\left(b^i, b^{i+1}\right)$	Joint torque vector evaluated at midpoint of each path segment
$\dot{\mathrm{q}}\left(b^i, b^{i+1}\right)$	Joint velocity vector evaluated at midpoint of each path segment
$\overline{\tau}$	Upper bound for joint torque vector
$\underline{\tau}$	Lower bound for join torque vector
W_{tau}	Diagonal matrix associated with joint torque constraints
$\overline{\dot{\mathrm{q}}}$	Upper bound for joint velocity vector
$\underline{\dot{\mathrm{q}}}$	Lower bound for joint velocity vector
b^0	Initial state of the optimization variable

Let $\varphi_b^k\left(b^k, b^{k+1}\right) = \frac{2\Delta s^k}{\sqrt{b^{k+1}} + \sqrt{b^k}} + \frac{1}{N_s} \cdot \left(\kappa^\tau \cdot P^\tau\left(\tau\left(b^k, b^{k+1}\right)\right) + \kappa^v \cdot P^v\left(\dot{\mathrm{q}}\left(b^k, b^{k+1}\right)\right)\right)$,

the *k*th element of Jacobian is given in Equation (B.6), and the *kl*th element of Hessian Matrix is given in Equation (B.7).

$$\mathbf{J}_k\left(\mathbf{b}\right) = \frac{\partial \varphi_b^{k-1}\left(b^{k-1}, b^k\right)}{\partial b^k} + \frac{\partial \varphi_b^k\left(b^k, b^{k+1}\right)}{\partial b^k} \tag{B.6}$$

$$\frac{\partial \mathbf{J}_k\left(\mathbf{b}\right)}{\partial b^l} = \frac{\partial^2 \varphi_b^{k-1}\left(b^{k-1}, b^k\right)}{\partial b^k \partial b^l} + \frac{\partial^2 \varphi_b^k\left(b^k, b^{k+1}\right)}{\partial b^k \partial b^l} \tag{B.7}$$

Table B-2: Outputs of trajectory planning using log-barrier method

	Output
b_{opt}	Solution to the trajectory optimization problem
tau_{opt}	Optimized joint torque associated with b_{opt}
$\dot{\mathrm{q}}_{opt}$	Optimized joint velocity, $\dot{\mathrm{q}}_{opt}^i = \frac{b_{opt}^i + b_{opt}^{i+1}}{2} \times q'\left(s^i\right)$
$\ddot{\mathrm{q}}_{opt}$	Optimized joint acceleration, $\ddot{\mathrm{q}}_{opt}^i = \frac{b_{opt}^{i+1} - b_{opt}^i}{2\Delta s^i} \times q'\left(s^i\right) + \frac{b_{opt}^i + b_{opt}^{i+1}}{2} \times q''\left(s^i\right)$
t_{opt}	Optimized path duration, $t_{opt} = \sum_{i=0}^{N_s-1} \frac{2\Delta s^i}{\sqrt{b_{opt}^i} + \sqrt{b_{opt}^{i+1}}}$
E_{opt}	Optimized energy consumption, $\mathrm{E}_{opt} = \sum_{i=0}^{N_s-1}\left(\frac{2\Delta s^i}{\sqrt{b_{opt}^i} + \sqrt{b_{opt}^{i+1}}} \times \sum_{j=1}^{ndof} tau_{opt}^{j,i} \times W_{tau}{}^{j,i} \times tau_{opt}^{j,i}\right)$

According to Equation (B.7), the Hessian matrix $\mathrm{H}\left(\mathbf{b}\right)$ is in tri-diagonal form. Hence at iterate $\mathbf{b}_{(j)}$, the Equation (B.5) can be transformed into the form

$$\begin{bmatrix} h_{1,1} & h_{1,2} & & \\ h_{2,1} & h_{2,2} & \ddots & \\ & \ddots & \ddots & h_{N_t-2,N_t-1} \\ & & h_{N_t-1,N_t-2} & h_{N_t-1,N_t-1} \end{bmatrix} \cdot \Delta\mathbf{b}_{(j)} = -\mathbf{J}\left(\mathbf{b}_{(j)}\right) \tag{B.8}$$

The Hessian matrix can be decomposed into two matrices though LU decomposition. The intermediate variables $\mathbf{z}_{(j)}$ can be computed by solving Equation (B.9) through forward substitution, and $\Delta\mathbf{b}_{(j)}$ can be determined iteratively by solving Equation (A.10) through backward substitution.

$$\begin{bmatrix} L_{1,1} & & & \\ L_{2,1} & L_{2,2} & & \\ & \ddots & \ddots & \\ & & L_{N_t-1,N_t-2} & L_{N_t-1,N_t-1} \end{bmatrix} \cdot \mathbf{z}_{(j)} = -\mathbf{J}\left(\mathbf{b}_{(j)}\right) \tag{B.9}$$

$$\begin{bmatrix} I & U_{1,2} & & \\ & I & & \\ & & \ddots & U_{N_t-2,N_t-1} \\ & & & I \end{bmatrix} \cdot \Delta\mathbf{b}_{(j)} = \mathbf{z}_{(j)} \tag{B.10}$$

The Gradient Newton Method for solving the Equation (B.1) can be given as follows:

Given a set of initialized parameters: κ, ξ δ, η, γ,

(1) Initialize a feasible set b^0 such that the joint torque constraints and joint velocity constraints are satisfied;

(2) At current iteration j, evaluate $E\left(\mathbf{b}_{(j)}\right)$; solving the first-order optimality conditions for Equation (B.1) using Newton method:

$$\mathbf{J}=\frac{\partial\Omega}{\partial\mathbf{b}}=0;\quad \mathbf{b}=\left[b_1,...b_{N_t-1}\right]^T$$

(a) If the Norm of Jacobian matrix is smaller than a user-defined threshold, i.e., $\left\|\mathbf{J}\left(\mathbf{b}_{(j)}\right)\right\|_F<\xi$, then $\mathbf{b}_{(j)}$ is return as the optimal solution to the problem in Equation (B.1);

(b) Else continue on (3).

(3) The search direction in iteration j can be determined through solving the linear equations (second-order Taylor approximation):

$$\mathbf{J}\left(\mathbf{b}_{(j)}\right)+\left.\frac{\partial\mathbf{J}}{\partial\mathbf{b}}\right|_{\mathbf{b}=\mathbf{b}_{(j)}}\cdot\Delta\mathbf{b}_j=0=\mathbf{J}\left(\mathbf{b}_{(j)}\right)+\mathrm{H}\left(\mathbf{b}_{(j)}\right)\cdot\Delta\mathbf{b}_{(j)}$$

(4) Set $\delta=1$, a coefficient related to the search step length;

(5) Update optimization variable in the form:

$$\mathbf{b}_{(j+1)}=\mathbf{b}_{(j)}+\delta\cdot\Delta\mathbf{b}_{(j)}$$

(6) Evaluate $E\left(\mathbf{b}_{(j+1)}\right)$;

(7) Adaptive search step length – Backtracking line search scheme (Boyd and Vandenberghe 2004, Chapter 9) to determine the search step length ***t*** through evaluation of certain criterion, such as:

$$E\left(\mathbf{b}_{(j+1)}\right)<E\left(\mathbf{b}_{(j)}\right)+\gamma\cdot\delta\cdot\mathbf{J}\left(\mathbf{b}_{(j)}\right)^T\cdot\Delta\mathbf{b}_{(j)}$$

(a) If the criterion is irritated, $\boldsymbol{t}$ shrinks by a positive scalar η, i.e., $\delta = \delta \cdot \eta$, go to step (5);

(b) Else $j = j + 1$, go to step (2).

Appendix C. AR Robot Programming and Path Planning Questionnaire

Name: _____

Email Address: _____

Date: _____

PART I

Table C-1: Data Collection.

Tasks	Time for training (Min)	Time to complete the task (seconds)	Number of control point selected
Control point selection			
Coordinate frame definition			
EE orientation planning			
EE orientation modification			

PART II

1. Do you have experience of operating an industrial robot?

___A:Yes

_____

_____

_____

___ B: No

2. Do you have experience of conducting robot path planning and task planning, such as in robot arc welding, material handling?

___A:Yes

_____

_____

_____

___ B: No

3. Have you had any experience using any other AR based systems?

___A:Yes

_____

_____

_____

___ B: No

4. Is it easy to fully understand the marker-cube manipulations when using it guiding the virtual robot end-effector? (1 is easy, 5 is difficult)

   1 2 3 4 5

   ☐ ☐ ☐ ☐ ☐

5. Is it easy to manipulate the marker-cube achieving control point selection?

   1 2 3 4 5

   ☐ ☐ ☐ ☐ ☐

6. Is it easy to manipulate the marker-cube achieving control point modification, such as insertion of a point into the control point list, or delete a point from the list?

   1 2 3 4 5

   ☐ ☐ ☐ ☐ ☐

7. Is it easy to manipulate the marker-cube achieving coordinate frame definition at the first control point?

   1 2 3 4 5

   ☐ ☐ ☐ ☐ ☐

8. Is it easy to manipulate the marker-cube achieving EE orientation selection at each control points?

   1 2 3 4 5

   ☐ ☐ ☐ ☐ ☐

9. Is it easy to manipulate the marker-cube achieving EE orientation modification at a control point of interest?

   1 2 3 4 5

   ☐ ☐ ☐ ☐ ☐

10. Is it easy to learn and use this AR-based system for robot path planning?

    1 2 3 4 5

    ☐ ☐ ☐ ☐ ☐

    Comments:

    _____

    _____

    _____

11. Other suggestions toward improving the current system, in terms of display equipment, interaction device, interaction mechanism?

    _____

    _____

    _____