On the Task-Driven Generation of Preventive Sensing Plans for Execution of Robotic Assemblies

SANTIAGO ENRIQUE CONANT PABLOS

Ph.D. DISSERTATION

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY

DECEMBER 2004

On the Task-Driven Generation of Preventive Sensing Plans for Execution of Robotic Assemblies

A Dissertation Presented by

Santiago Enrique Conant Pablos

Submitted in partial fulfillment of
the requirements for the degree of

Doctor of Phylosophy
in the field of
Artificial Intelligence

Thesis Committee:

Katsushi Ikeuchi, The University of Tokyo
Horacio Martinez Alfaro, ITESM Campus Monterrey
José Manuel Sanchez García, ITESM Campus Monterrey
Noel León Rovira, ITESM Campus Monterrey
Olivia Maricela Barrón Cano, ITESM Campus Monterrey

Center for Intelligent Systems
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey
December 2004

Declaration

I hereby declare that I composed this dissertation entirely myself and that it describes my own research.

Santiago Enrique Conant Pablos
Monterrey, N.L., México
December 2004

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey

Graduate Program in
Electronics, Computing, Information and Communications

The committee members hereby recommend the dissertation presented by Santiago Enrique Conant Pablos to be accepted as a partial fulfillment of requirements to be admitted to the **Degree of Doctor of Philosophy in Artificial Intelligence**.

Committee members:

Dr. Katsushi Ikeuchi
Advisor

Dr. Horacio Martínez Alfaro
Advisor

Dr. José Manuel Sanchez García

Dr. Noel León Rovira

Dr. Olivia Maricela Barrón Cano

Dr. David A. Garza Salazar.
Director of Graduate Program in
Electronics, Computing, Information and Communications

Abstract

It is well known that success during robotic assemblies depends on the correct execution of the sequence of assembly steps established in a plan. In turn, the correct execution of these steps depend on the conformance to a series of preconditions and postconditions on the states of the assembly elements and in the consistent, repeatable, and precise actions of the assembler (for instance, a robotic arm). Unfortunately, the ubiquitous and inherent real-life uncertainty and variation in the work-cell, in the assembly robot calibration, and in the robot actions, could produce errors and deviations during the execution of the plan.

This dissertation investigates several issues related to the use of geometric information about the models of component objects of assemblies and the process of contact formation among such objects for tackling the automatic planning of sensing strategies. The studies and experiments conducted during this research have led to the development of novel methods for enabling robots to detect critical errors and deviations from a nominal assembly plan during its execution. The errors are detected before they cause failure of assembly operations, when the objects that will cause a problem are manipulated. Having control over these objects, commanded adjustment actions are expected to correct the errors.

First, a new approach is proposed for determining which assembly tasks require using vision and force feedback data to verify their preconditions and the preconditions of future tasks that would be affected by lack of precision in the execution of those tasks. For this, a method is proposed for systematically assign force compliance skills for monitoring and controlling the execution of tasks that involve contacts between the object manipulated by the robot arm in the task and the objects that conform its direct environmental configuration. Also, a strategy is developed to deduce visual sensing requirements for the manipulated object of the current task and the objects that conform its environment configuration. This strategy includes a geometric reasoning mechanism that propagates alignment constraints in a form of a dependency graph. Such graph codes the complete set of critical alignment constraints, and then expresses the visionand force sensing requirements for the analyzed assembly plan. Recognizing the importance of having a correct environment configuration to succeed in the execution of a task that involve multiple objects, the propagation of critical dependencies allow to anticipate potential problems that could irremediably affect the successful execution of subsequent assembly operations. This propagation scheme represents the heart of this dissertation work because it provides the basis for the rest of the contributions and work. The approach was extensively tested demonstrating its correct execution in all the test cases.

Next, knowing which are the tasks that require preventive sensing operations, a sensor planning approach is proposed to determine an ordering of potential viewpoints to position the camera that will be used to implement the feedback operations. The approach does not consider kinematic constraints in the active camera mechanism. The viewpoints are ordered depending on a measure computed from the intersection of two regions describing the tolerance of tasks to error and the expected uncertainty from

an object localization tool. A method has been posed to analytically deduce the descriptions of inequalities that implicitly describe a region of tolerated error. Also, an algorithm that implements an empirical method to determine the form and orientation of six-dimensional ellipsoids is proposed to model and quantify the uncertainty of the localization tool. It was experimentally shown that the goodness measure is an adequate criterion for ordering the viewpoints because agrees with the resulting success ratio of real-life task execution after using the visual information to adjust the configuration of the manipulated objects.

Furthermore, an active vision mechanism is also developed and tested to perform visual verification tasks. This mechanism allows the camera move around the assembly scene to recollect visual information. The active camera was also used during the experimentation phase.

Finally, a method is proposed to construct a complete visual strategy for an assembly plan. This method decides the specific sequence of viewpoints to be used for localizing the objects that were specified by the visual sensing analyzer. The method transforms the problem of deciding a sequence of camera motions into a multi-objective optimization problem that is solved in two phases: a local phase that reduces the original set of potential viewpoints to small sets of viewpoints with the best predicted success probability values of the kinematically feasible viewpoints for the active camera; and a global phase that decides a single viewpoint for each object in a task and then stitch them together to form the visual sensing strategy for the assembly plan.

Dedication

To my wife **Jaqueline**.

To my kids **Santiago**, **Gerardo**, and **Jackeline**.

To my parents **Santiago** and **Odila**.

Thanks for all your unconditional confidence, support, patience, and encouragement.
You were my main motivation for pushing through this work.

Acknowledgements

I would first like to express my gratitude to my advisor Dr. Katsushi Ikeuchi for his unconditional support, advise and encouragement during the course of this research. Discussions with him, his insights and vast experience greatly helped me to find the right directions. I thank him for giving me the opportunity to explore many ideas and for having an open door and open mind whenever I needed it. I was fortunate to be accepted by him as a Visiting Scholar at Carnegie Mellon University (CMU) and as a Visiting Research Associate at The University of Tokyo (UT). His help and support during my internships were invaluable. Sensei, thank you very much!

I am also indebted to my local supervisor Dr. Horacio Martínez for his advice, encouragement and guidance. I also thank my dissertation committee members Dr. Noel León, Dr. José M. Sánchez, and Dr. Olivia M. Barrón for their careful reading of this dissertation and their valuable feedback regarding this manuscript.

I thank to Dr. Francisco Cantú, Dr. Rogelio Soto, and Dr. Fernando Jaimes for their support, patience, and constant stimulation to continue and finish my doctoral studies, while a student and member of the current Center for Intelligent Systems and former Center for Artificial Intelligence.

I recognize the financial support of ITESM and CONACyT, without it, it would have been impossible to realize this dream. Special thanks to the Robotics Institute at CMU and the Institute of Industrial Sciences at UT for allowing me to use its excellent computational resources and facilities to develop this research.

I would like to express my gratitude to my fellows and friends at CMU: thanks to Antonio Diaz, Octavio Juarez, George V. Paul, Yoichi Sato, Imari Sato, Hiroshi Kimura, Kohtaro Ohba, Mark Wheeler, Prem Janardhan, Harry Shum, and Yunde Yiar. Thanks also to my fellows and friends at UT: Masataka Kagesawa, Yoichi Sato, Imari Sato, Ko Nishino, Kentaro Kawamura, Keiko Motoki, Tomotaka Saito, Tomoyuki Horiuchi, Hajime Ohno, Hirohisa Tominaga, Koichi Ogawara, Yoshiko Matsuura and Jun Takamatsu. Special thanks for his friendship and attentions to Hiroshi Kimura. My internships would have been much more difficult without them. I cherished their company and friendship.

A very special thank to my "Gran Compadre" Hugo Terashima who was always there for me. His encouragement, advise, and unconditional support helped me to persevere and finish this work. He was always interested and cheering me up. His help has been priceless.

I would also like to express my gratitude to my fellows and friends at the CSI: Olivia Barrón, Manuel Valenzuela, Leonardo Garrido, Eduardo Uresti, Ramón Brena, José Luis Aguirre, Nora Aguirre, Doris García, Arturo Galván, Carlos Cant and Leticia Rodríguez. Their friendship during all these years has been very important to me. I also like to thank my fellow and friend José Luis Gordillo for introducing me to the scientific world, advising me, and putting me in the right research path.

Last but certainly not least, my biggest gratitude and appreciation goes to all members of my family: my wife, my kids, my parents, my siblings, my nephews and nieces, and my parents and siblings in law, for their love, patience, confidence, support, and help for these many years and for the years to come.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The interest in systems that automatically generate robot programs for assembly tasks is growing due to the wider use of industrial robots in assembly applications [69]. This interest has been motivated by the desire to reduce costs in small batch manufacturing applications and to reduce time to reach rapidly changing markets.

The successful execution of assembly plans by robots is of fundamental importance in modern manufacturing industries. In most of the cases, such success is reached by surrounding the assembly cell by hard automation devices that guarantee the conditions assumed by the plan. This devices increase the cost and reduce the flexibility of the system.

No matter how many factors were taken into account during the automatic assembly planning process, the failure of its blind execution and otherwise, the cost of the hard engineering required to ensure its success, remains one of the most motivational facts for the development of new approaches and techniques to cope with real life uncertainty and errors during assembly plan execution. The roll of the hard engineering is precisely to reduce or eliminate the uncertainty and errors that, otherwise, would modify the state of the system in an unexpected way and produce failure.

In an ideal world, *uncertainty* – a level of variability in an action's outcome or measurement result – can be ignored. If uncertainty is absent, the assembly planner can define tasks that guarantee success and its verification during execution is not required. Unfortunately, ideal worlds only exist in simulated environments, where most of the task planners do their job. Robot plan execution outside the simulated environment had to deal with real world uncertainty. Uncertainty transform real plan execution in an error-prone task.

Commonly, to reduce the requirements of hard automation devices and still succeed in making a robot execute an specific assembly plan, the sequence of operations generated by the task planner (whether human, computational, or some combination) has to include an intermingled sequence of assembly-action/sensory-feedback/adjusting-action commands, so that the assembly execution advance could be monitored and deviations from the plan (if any) could be prevented or corrected [6].

There are several proposals for coping with uncertainty using sensory information during assembly planning and plan execution [64]. Every one working under a set of specific assumptions and restrictions. From them it can be concluded that the development of domain-independent planners is currently beyond the scope of our technology and that task-oriented domain knowledge is required to generate successful plans.

1.2 Problem Statement and Context

Fully automated assembly task planning suffers of combinatorial explosion which makes it inappropriate in most realistic situations. Kavraki et al. [54] proved, that the problem of automatically generating assembly sequences is NP-complete even in the two-dimensional case. Adding uncertainty and the requirement of including sensory-feedback operations does not make it better. To deal with this problem, several clues and constraints have to be inserted into the system to reduce its complexity at a manageable size. Natural constraints come from geometry and physics of a domain-specific knowledge; artificial constraints are simplifying assumptions that together with heuristics work as focusing hints to reduced sets of partial or full potential solutions.

An assembly plan describes a sequence of steps that have to be performed by a robot in order to assemble a group of parts. In this dissertation these parts are modeled as polyhedral objects that are manipulated by a robot arm, also referred as *manipulator*, one at the time.

The manipulation process of an assembly part by the robot is composed, basically, by a series of transit, grasping, transfer and ungrasping operations. A transit operation is an assembly step where the manipulator moves without carrying an object. A grasping operation is an assembly step where the manipulator takes control over an object by grabbing it with is end-effector (*gripper* or *hand*). This object is subsequently known as the *manipulated object*. A transfer operation is an assembly step where the manipulator moves carrying the manipulated object. Finally, a ungrasping operation is an assembly step where the robot release the manipulated object loosing its control. The mating process among the assembly components is performed during the transfer operations. The motion of an object can be performed in free-space (moving without touching other objects) or in contact-space (moving while touching other objects). The type and source

of sensing data will depend on the category in which an object's motion falls.

The *configurations of the parts* – their position and orientation – specified in a nominal assembly plan are expected to be reached while performing transfer operations. During some of these assembly steps, the manipulated object enters in *contact* – a state or condition of touching or of being in immediate proximity – with some stationary objects. These objects are subsequently known as *environmental objects*. It is also expected that the fixed condition and pose of the environmental objects is maintained during and after the execution of such steps. A convenient way to take into account the robot actions during transit operations is considering its hand and fingers as mating parts that generate temporal *contact states* – configuration of contacts between the assembly components – when grasping new objects. In this way, the robot's gripper can be considered as the manipulated object and the object to be grabbed as an environmental object in an insertion operation.

It is well known that success during robotic assemblies depends on the correct execution of the sequence of assembly steps established in a plan. In turn, the correct execution of these steps depend on the conformance to a series of preconditions and postconditions on the states of the assembly elements and in the consistent, repeatable, and precise actions of the assembler (for instance, a robotic arm). Unfortunately, the ubiquitous and inherent real-life uncertainty and variation in the work-cell, in the assembly robot calibration, and in the robot actions, could produce errors and deviations during the execution of the plan. In those cases, either the planned configuration of the manipulated object is not accurately achieved or the configurations of some environmental objects are modified by the physical interaction with the manipulated object, during the execution of an assembly step.

The roll of hard automation devices, such as fixtures, automatic feeders, re-orienting devices and ad-hoc grippers and tools, is, precisely, to reduce the uncertainty with respect to the location and configuration of the assembly elements. In this way, the conditions expected in an assembly plan are assured. Planning, constructing and using such devices is a very interesting and complex problem by itself. Furthermore, its presence and use increase the work-cell cost and resistance to change, then reducing the work-cell's flexibility. However, it is not always possible nor convenient to eliminate them from a work-cell, because uncertainty would introduce the possibility of errors and deviations from absolute expected conditions.

Alternatively, the inclusion of sensors and sensing strategies could allow to eliminate some of these automation devices by getting information about sources of uncertainty. Sensors used to monitor the assembly plan execution, in conjunction with a relaxation in the requirement of absolute positioning, transforms a *conformant process* – a sensorless process – into a *contingency process* – a conditional process including sensing actions [80] – that adapts to detected divergences from the plan and react to them. The price to

pay for the earned flexibility is a reduction on the reliability of the plan execution.

Strictly speaking, uncertainty affects all the tasks, however, every assembly operation has a capacity to tolerate certain amount of error: objects offer some resistance to be moved, insertion slot configurations include some clearance for insertion, the robot arm allows some passive compliance, etc. Then, not necessarily all assembly operations require of sensing verification.

This dissertation presents an approach to the automatic planning of sensing strategies to cope with potential problems to be caused by real-life uncertainty and errors during assembly task execution. The approach assumes the use of force feedback operations to perform force compliance tasks, determine visual sensing requirements, and implement visual sensing strategies to detect and prevent (or at least reduce) the possibility of failure during the execution of the assembly operations. Several computer programs that implement the introduced methods were developed. The problem starts with a *nominal assembly plan*, i.e. a sequence of assembly tasks that would execute successfully in the absence of uncertainty, and propose a visual sensing strategy that maximize the possibilities of success of the tasks by characterizing, modeling, and minimizing the uncertainty from different physical sources with respect to the task tolerances.

1.3 Research Questions

The main motivation for this dissertation was the discovery that assembly is not a trivial endeavor for a robot. The assembly plan and assembly execution by a robot requires of a trade-off between the reliability and the flexibility of the solutions. Then, it is worth a try finding new approaches to avoid failure or at least increase the possibilities of success in the robotic execution of an assembly operation while reducing the requirements of hard automation.

The questions which motivated and guided the research presented in this dissertation are the following:

- Why does a robot fail to execute an assembly plan?
- What is the roll of hard engineering devices such as fixtures in the successful execution of assembly plans?
- Is it possible to eliminate some of these devices by using sensing?
- What are the consequences of eliminating some of these devices?
- Do all the assembly operations need of sensing?

- What are the most difficult assembly tasks for a robot?
- Is it enough with using sensing in these tasks?
- Which other tasks require of sensing?
- Which type of sensory information is needed for a task?
- What sensory information is needed for task's sensing?
- Which are the assembly elements that contain such information?
- When is it recommended to use vision?
- What visual information is required?
- How is this information obtained?
- What is the best visual sensing strategy to get such information?
- How can the selected strategy be implemented?
- What kind of vision sensor to use?
- How is this sensor controlled during sensing?
- What to do in the case of detecting an error or deviation from the assembly plan?

1.4 Solution Overview

This dissertation presents an innovative approach to the automatic planning of sensing strategies to help in the successful execution of assembly plans by robots. Figure 1.1 depicts a graph that describes the solution strategy followed during the development of this research. The contributions of this work were constructed around four main elements: a sensing analyzer, a sensor planner, a design of an active vision mechanism, and a sensing planner. The figure also shows the chapters of this document where each of the elements are described.

1.4.1 Sensing Analysis

The sensing analyzer determine which tasks require of sensing operations, what sensing information needs to be gathered by the sensors, and from which objects could this information be obtained. As shown in Figure 1.1, the sensing analyzer recibes as input the nominal assembly plan specification and the CAD models of the assembly parts, and

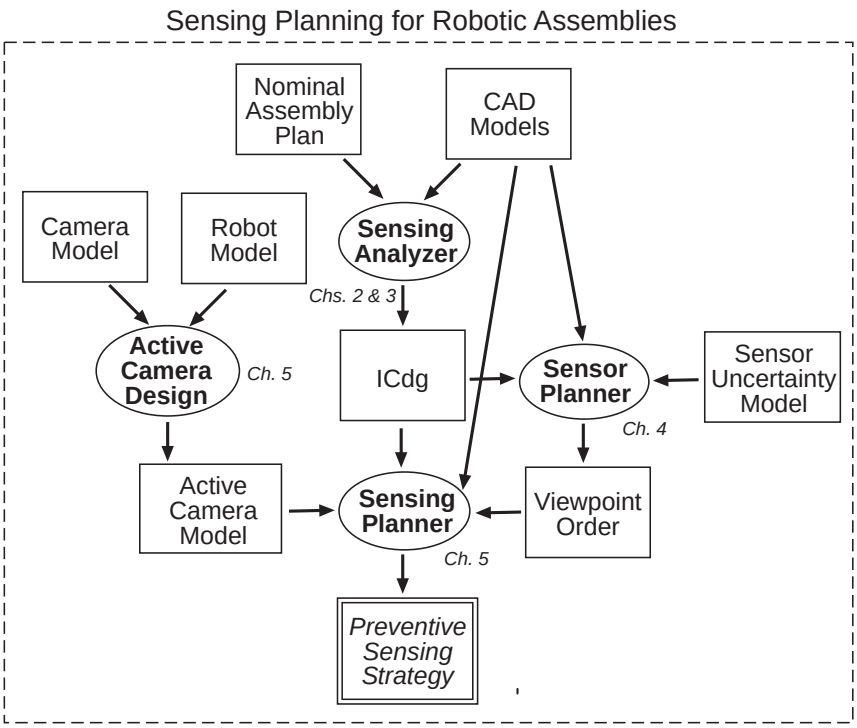


Figure 1.1: Overview of Solution Strategy.

produces as output a sensing requirements specification. The most important output is a coded form of the sensing requirements in the Insertion and Contact dependency graph (ICdg), which is detailed in Chapter 3.

The criteria used to decide which tasks require of sensing operations is that of reaching specific contact-state relations among mating objects. It is realized that the nature of mating operations is very important, specially those which resulted of what is known as *fine-motion planning* (planning of jam-free sliding motions to achieve a goal configuration of a set of spatial relationships between assembly parts [16]), e.g. peg-and-hole operations. These tasks have a low-tolerance to errors during their execution and are considered the most difficult tasks for a robot. Such assembly operations usually require of adaptive behaviors in accordance with a planned maneuvering strategy.

To have success in a fine-motion operation a series of preconditions have to be met. To fulfill such preconditions, additional tasks could require of using sensing feedback operations, e.g. sensing tasks are recommended to ensure that expected insertion configurations are obtained for future insertion operations. Then the recognition of fine-motion tasks ,during the sensing analysis, fires up a backward reasoning process to recognize new operations where sensing verification is needed.

The type of required sensing information depends on the purpose of the task, the time in which it is obtained, the intention of the information, and the available sensors. In

this dissertation, the sensing strategies assumes the use of force and vision sensors. The use of force feedback data is recommended during the task execution to detect force patterns due to contacts while performing guarded and compliant motions. The use of vision feedback data is recommended before some specific tasks in order to verify preconditions and, if necessary, adjust the configuration of manipulated objects in a preventive fashion.

The nominal assembly plans for this dissertation are restricted to binary plans, which are also linear and sequential, that describe a totally-ordered sequence of assembly steps (see Appendix B for definitions of assembly terms). The sensing analysis module determine when is recommendable to use visual sensing operations and force sensing skills through an analysis of the contact state formation process. The type of assembly operations in such plans can be recognized and classified by the type of contact state transition that produces as result of added or reduced contacts with the environment.

This thesis assumes the existence and use of force control mechanisms that implement certain force compliance skills. Actually, to accomplish some of the experiments a limited force control program was developed to execute guarded motions using a wrist sensor. The sensing analysis module determine when to use such sensing operations, but do not enter in further details about how to actually implement more elaborated sensing skills.

The main contributions of this dissertation are in the planning of visual sensing strategies. Vision is employed to deduce the configuration of objects in a scene from intensity images. Vision is utilized to discover conditions on the configuration of the objects that would preclude reaching some projected contact relations. The sensory information is used to deduce, through a geometric reasoning process, the relative constraining relations of some objects with respect to other currently assembled objects or with objects to be assembled in future tasks.

Since the assembly is carried out in an structured and known environment following a pre-planned sequence of steps, the job of the visual sensing operations is reduced to object localization, and more precisely, to object pose refinement. The plan prescribe a pose where each object is expected to be found. The visual tasks can not completely avoid pose uncertainty of the observed objects, then the computed configurations include certain amount of error. The sensing analyzer determine the most important pose parameters from the recognition of the critical dimensions of the task that generated the sensing requirements. Then this information is used to compose a criterion for deciding the configuration of the sensor.

The objects that contain the visual sensing information are those that participate in insertion operations and those involved in assembly operations where a manipulated object enters in contact with multiple objects in the environment. Every task in which an object participates would potentially add constraints in its pose relations with respect

to pre-assembled objects and the environment. To avoid undo-redo operations, an object has to satisfy all the constraints resulting from the sensing analysis of the full plan.

1.4.2 Sensor Planning

The sensor planner determine the goodness of a finite set of viewpoints to position and orient a single camera. The camera is used to get intensity images for localizing the assembly parts containing critical information for verifying the preconditions of assembly operations. As shown in Figure 1.1, the sensor planner recibes as input information from three sources: the ICdg, the CAD models of the assembly parts, and the sensing uncertainty models quantified for each potential viewpoint. It outputs a viewpoint implicit order described by the computed goodness measure.

For this work, a visual sensing strategy is defined as a sequence of sensor configurations to be performed before an assembly step that requires of preventive vision. This sequence is supposed to describe the best viewpoints and sensor parameters to use for observing the objects that could affect the immediate or future success of the execution of assembly steps.

A good sensor configuration is defined as a set of parameter values that minimize the pose estimation errors, with respect to a set of critical dimensions of a task. Instead of only minimizing the uncertainty of the pose estimation, the criterion used to evaluate the different alternatives of sensor configurations maximize a predicted success probability (Psp) value. A Psp value is computed by quantifying the portion of the sensor's uncertainty representation that fall inside a region of tolerated error for an assembly operation. The sensor's uncertainty is obtained by quantifying an approximated uncertainty model representation. Such quantification is realized from a series of empirically realized pose refinement experiments over target objects. The region of tolerated error for an assembly operation is represented by a set of inequalities obtained from the analytical description of the relation of geometric features in contact of the participant objects in a mating operation. Both representations are described in the same parametric space.

In general, there are an infinite number of sensor configurations that could be used to determine the pose of an object. However, in this work a generate-and-test approach to sensor planning was chosen. Then, the number of alternative sensor configurations was limited to a discrete number corresponding to the number of triangles in a tessellated sphere (a *viewing sphere*) around the target object in its center. The size of the sphere and the number of tessels is predefined. The sensor is localized in the center of each tessel and is oriented looking through the center of the sphere. The Psp measure is computed for each sensor configuration and every alternative is sorted based in a Psp descending order. Before deciding the best configuration, a feasibility analysis

has to be done following the resulting order. This feasibility analysis will verify if a sensor configuration satisfy a set of kinematic constraints imposed by an active camera mechanism.

1.4.3 Active Camera Design

An active vision design was developed to implement the visual feedback operations for robotic assemblies. Since the selected visual locatization tool localize 3-D objects over single intensity images, a camera-in-hand was realized. As shown in Figure 1.1, the active camera design used as input the models of the camera and the robot that carries it. The design outputs the active camera model which is instantiated by executing a series of calibration processes. A detailed description is given in Chapter 5.

The active camera mechanism is composed by a pan-tilt camera fixed to a manipulator's hand. The camera can rotate horizontally (*pan*) and vertically (*tilt*) to adjust its viewing direction. For this dissertation, the manipulator that carries the camera was a Cartesian robot arm of five degrees of freedom, but only three of them, those that translate the camera where used. The other two are used only to define an initial attitude for the camera. The pan-tilt camera allows a reduced range of rotation with respect to horizontal and vertical rotation axes. The manipulator has a well defined parallelepiped workspace. These features define the kinematic constraints that are used to decide the feasibility of reaching a predefined viewpoint during the sensing planning stage.

Some of the most important and hard jobs on building an active camera mechanism is that of calibrating the system for every possible viewpoint. In this thesis, the viewpoints are fixed with respect to the object to be observed, but the object configuration can change due to uncertainty, which means that the configuration of the viewpoints can not be fixed with respect to the camera mechanism, and this precludes the possibility of calibrating the system for every viewpoint off-line. The solution implemented in this work was modeling the active camera mechanism in a way to adjust its calibration parameters dynamically.

The camera calibration model includes a set of intrinsic and extrinsic parameters defined as a function of the mechanical and photometric characteristics of the sensor and its pose with respect to a coordinate system of reference. In general, the calibration model is only an approximation to reality, and consequently, calibration is proposed and solved as an optimization problem. As a result, usually, any change in the sensor state requires of a full re-calibration, if this is not done, the new calibration data would include additional error in its measurements. To minimize this effect while avoiding to perform a full recalibration procedure, the calibration schema followed in this thesis was one that for a non-calibrated viewpoint, the values for the intrinsic parameters are decided as a function of the intrinsic parameters for the closest calibrated viewpoint, while

the extrinsic parameters are computed from the camera attitude and the manipulator position.

After deciding the best feasible sensor configuration (viewpoint) to observe an object and to perform its pose refinement, the calibration of the active camera mechanism and its configuration are adjusted, and the object is localized.

1.4.4 Visual Sensing Planning

The visual sensing planner construct a complete visual strategy for an assembly plan. This method decides the specific sequence of viewpoints to be used for localizing the objects that were specified by the visual sensing analyzer. As shown in Figure 1.1, the visual sensing planner recibes its input information from four sources: the ICdg, the CAD models of the assembly parts, the active camera model, and viewpoint implicit order. It outputs a total and definite preventive sensing strategy describing the sequence of camera motions to use during the execution of the original nominal assembly plan.

The proposed method transforms the problem of deciding a sequence of camera motions into a multi-objective optimization problem that is solved in two phases: a local phase that reduce the set of potential viewpoints to small sets of viewpoints with the best Psp values of the kinematically feasible viewpoints for the active camera; and a global phase that decides a single viewpoint for each object in a task and then stitch them together to form the visual sensing strategy for the assembly plan. Additionally to the Psp values, the multi-objective optimization method try to minimize a measure of the level of occlusion from the considered viewpoints and the distance that the camera has to be moved. A dynamic programming approach was selected to perform the optimization process.

1.5 Main Contributions

This dissertation investigates several issues related to the use of geometric information about the models of component objects of assemblies and the process of contact formation among such objects for tackling the automatic planning of sensing strategies. The studies and experiments conducted during this research have led to the development of novel methods for enabling robots to detect critical errors and deviations from a nominal assembly plan during its execution. The errors are detected before they cause failure of assembly operations, when the objects that will cause a problem are manipulated. Having control over these objects, commanded adjustment actions are expected to correct the errors.

The main contributions of this thesis are the following:

- A method to systematically deduce assembly skill primitives and force compliance skills requirements for assembly tasks.
- Introduction of a graph representation that codes the sensing requirements of an assembly plan.
- A method to propagate direct and indirect translational and rotational dependencies among environmental objects.
- A method to analytically describe a region of tolerated error for assembly tasks.
- An algorithm to quantify and represent the uncertainty of a template matching localization tool that works on intensity-images.
- A method to evaluate potential viewpoints to locate the camera for localization tasks.
- Development of an active camera mechanism for visual verification tasks.
- A method to construct preventive vision strategies for a complete assembly plan.

1.6 Literature Review and Background

1.6.1 Assembly Execution using Vision and Force Feedback Data

Common assembly tasks executed by humans combine the assembler's experience and information of their sensory mechanisms. Industrial robots with the capability to integrate a priori information and sensor data, acquire knowledge, and perform reasoning would be ideal for the factory of the future. In this respect, current robot systems should use internal and external multi-sensor information for controlling and monitoring the robot-manipulation tasks. For this, research has been prompted in combining both force and vision sensor data and knowledge-based processing methods to obtain more intelligent behavior in robot-based work cells [53, 84, 102].

Vision is a useful robotic sensor since it mimics the human sense of seeing and allows to get measures of the environment without touching. Its usefulness has been demonstrated since the early work of Shirai and Inoue [89] for correcting the position of a robot through a visual feedback loop. The accuracy of a visual control solution depends on the accuracy of the visual sensor and the robot. The two major approaches for using vision are the

alternating loop of picture taking and robot motion, and the closed-loop position control referred as *visual servoing*, first introduced by Hill and Park [36]. A review of control based on visual feedback can be found in [43]. Most research in visual control has concentrated on free motion control.

Other useful robotic sensors are force and tactile sensors. These sensors are indispensable for performing assembly tasks in contact space. The two major approaches to force control are the impedance control approach proposed by Hogan [42] and the hybrid control proposed by Raibert and Craig [76]. Successful systems, using both approaches, have been developed for cases where the knowledge of the environment is exact, however, robots are far less effective in applications where the environment and object position are not accurately controlled [43].

The programming difficulty and the lack of sensor integration have kept the creation of sensor-based robot programs as a great challenge. This challenge has been confronted by many researchers by constructing skill libraries and behaviors as source of robust task-achieving programs [104]. Several researchers have proposed robot control primitives. Smithers and Malcolm [91] proposed behavior-based modules for programming robotic assembly in which uncertainty is resolved at run-time. Steward et al. [94] developed a reconfigurable module (Chimera) based in a theory of computation for sensor-based robotics. Other researchers continued developing the skills or elementary operations needed for effective use of robots [34, 52, 8]. Morrow et al. [65] developed a sensori-motor command layer to combine sensing and action to apply to many tasks within a domain. They included four vision-driven primitives based on visual servoing techniques, to enforce positioning constraints; and four force-driven primitives to perform tasks in the presence of contacts.

Tominaga and Ikeuchi [103] proposed a method to make robust observations against noise by decomposing motion trajectories into small segments based in face contact analysis and allocating an operation element referred to as sub-skill. They selected their four motion sub-skills from the basic motions proposed by Suehiro in [96]. Mosemann and Wahl [66] also presented a method to decompose complex sequences for assembly operations into skill primitives for enabling a system to execute the operations in a real robot workcell. The skill primitives are elementary sensor-based robot movements(like "MoveTo"), system commands (like "OpenGripper") and sensor functions (like "LocateObject"). To classify the type of assembly operation, they used features like local depart spaces, symbolic spatial relations, and the necessary tools. Other works about synthesizing and using skill-primitives to decompose and execute assembly tasks are documented in [68, 90, 39, 59].

In assembly tasks, a robot falls into difficulties if it monitors the task only by vision, force, or touch sensor [1]. Some works on using multiple sensor capabilities have focused on integration of vision and force sensor information. Allen [4] developed a system

that uses both of stereo vision and touch sensor information. His system maintains a model database of the objects and the fusion of vision and touch information is used for matching the sensing data with the model database. Ishikawa et al. [51] presented a multi-sensor method of vision and force information to estimate contact position between a grasped object and the other objects in the environment. Nelson and Khosla [71] combined the force and vision feedback using the so called *vision and force resolvabilities*. The term resolvability refers to the ability of a sensor to resolve object positions and orientations. Collani et al. [107] presented a neuro-fuzzy model to integrate vision and force control in the execution of assembly tasks. They used uncalibrated cameras.

1.6.2 Contact-State Analysis and Identification

The contact-state identification problem is a very important problem for the automatic planning of assembly tasks under uncertainty and the automatic planning of sensing strategies for helping during their robotic execution.

The representation of a contact state between two objects is usually done in terms of the involved topological elements, i.e. faces, edges and vertices. In [60], Lozano-Pérez presents the contact states as a set of contact primitives that are defined as point contacts, i.e. vertex-edge in 2-D objects and vertex-face contacts in 3-D objects. Desai and Volz [19] defines the contact primitives, called elemental contacts, as a pair of topological elements, and a contact state as a set of elemental contacts called contact formation. Contact analysis is simplified with these primitives, since less primitives are required to describe a contact state. Further on, Xiao [110] introduces principal contacts as those elemental contacts necessary for characterizing motion freedom, and the contact formation as a set of principal contacts.

Besides configuration information, force information is also used for contact identification. Hirai et al. [38, 37] deal with the estimation of contact states from force information by using state classifiers based on geometric models of the objects, and which are formulated with the theory of polyhedral convex cones. This theory originally by Tucker and Goldman [57] has also been used by Paul [74] to partition the contact space into a set of finite topologically distinct states, and Hirukawa et al. [40] to analyze the freedom of an object in contact.

Brost and Mason [11] present the dual representation, which is a method for analyzing planar contact problems that represent planar motions and forces by acceleration centers. This graphical method allows the reasoning about sets of feasible contact motions, and the sets of forces consistent with those contact motions. Other approaches use force information to estimate the contact position when the geometry of the manipulated object is assumed to be unknown [56, 35].

Contact state identification in the presence of uncertainties is complex because several contact states may be compatible with the sensed information. Desai and Volz [19] presents an algorithm to verify termination conditions of compliant guarded motions which has an static and an active phase based on an hypothesis and test scheme. Similarly, Spreng [93] uses test motions for verifying contact hypothesis in terms of motion freedoms. The determination of all the possible contact states due to the uncertainties is not a trivial problem. Rosell et al. [78] presents a procedure that computes the set of contact configurations for planar assembly tasks using the knowledge of the robot configuration and taking into account modeling and sensing uncertainty. This procedure complements their procedure based in force information [7]. To avoid the complexity of finding the contact hypothesis for assembly tasks in the space (i.e. with 6 DOF), Xiao and Zhang [111, 112, 113] introduce a method for growing polyhedral objects by its location uncertainties in physical space, and implement an algorithm for finding all principal contacts possibly established between their features.

Other approaches that model the assembly tasks as discrete event dynamic systems, focus on the recognition of the contact events. McCarragher et al. [63] use a process monitor based on Hidden Markov Models for this purpose, and similarly Eberman [25] presents a statistical, model-based contact-state observer.

1.6.3 Visual Sensor Placement and Sensing Strategies

The quick growth of automation in manufacturing industry requires that computer vision play an important role in assembly automation. Then, determination of viewpoints and viewing strategies become critical for achieving full automation and high efficiency in assembly execution. The relevant work on *sensor placement*, which do that, can be categorized into model-based and nonmodel-based. Nonmodel-based sensor placement is used for 3-D object reconstruction and modeling, while model-based sensor placement is used in assembly execution, inspection, object recognition, dimension measurement, etc. [14].

In model-based vision tasks, researchers try to find an admissible domain of viewpoints to place the sensor to look at one or several object features [15]. The works can be divided in those that use a *generate-and-test strategy*, like the HEAVEN system of Sakane and Sato [82], and those that use a *synthesis approach*, like the *Machine Vision Planner* (MVP) system of Tarabanis et al. [101].

The generate-and-test strategy discretize the space for sensor positions, like HEAVEN that uses a tessellated sphere of pre-given radius for inspecting an object. Other systems that follow this approach are the *Vision Illumination Object* (VIO) system [81], and the Illumination Control Expert (ICE) system [85]. Trucco et al. [105] reported GASP (*General Automatic Sensor Planning*), a generate-and-test system for sensor planning

which is used to compute the optimal positions for inspection tasks. GASP uses a feature inspection representation to compute a viewpoint based on feature visibility, using an approximated model, and measurement reliability, based on an experimental quantitative assessment. The feature inspection representation is computed off-line and is used by GASP to compute on-line plans that they call *inspection scripts*.

In the other hand, the synthesis approach model the constraints as analytical functions, describing spaces that satisfy several constraints, like MVP that determine sensor locations and sensor parameters for viewing a set of surfaces and avoiding occlusion, conforming to feature detectability constraints such as: *visibility* – for the features to be detectable from the sensor; *field of view* – for the features to fall onto the active area of the sensor; *focus* – for the features to be into the focus range of the sensor; and *spatial resolution* – for the features to appear of an appropriated size. Additional constraints include *illuminability* – for the features to be detectable by the sensor; *dynamic range* for the image irradiance from the features to be inside of the dynamic range of the sensor; and *contrast* – for edges features to present an adequate disparity in image intensity values. Tarabanis et al. [100] presents an excellent survey of research in the area of vision sensor planning.

Hutchinson and Kak [44] proposed a method to planning sensing strategies dynamically, based on the system's current best information about the world. They apply the concept of entropy from Shannon's information theory [87] to minimize the remaining ambiguity in proposed sensing operations. To do this the system formulates object hypotheses and assesses its relative belief in those hypotheses using the Dempster-Shafer Theory [86], an approach to reasoning with partial evidence. Abrams et al. [3] also extended MVP to perform dynamic sensor planning in an environment in which objects are moving. They basically break a task into intervals an determine a single viewpoint to monitor each interval. To solve the occlusion problem, the system computes the volumes swept by all moving objects during the intervals, and computes viewpoints which avoid the occlusion by these swept volumes [2].

Zhang [114] proposed a method for determining an optimal two-dimensional spatial placement of multiple sensors participating in robot perception tasks. The method assumes that the sensor uncertainties are specified in terms of their covariance matrices. Then using an interpretation that the uncertainty in each sensor represents an ellipse, the method determines the sensor placement to minimize the uncertainty of the consensus estimate by minimizing the area of the uncertainty ellipse.

Briggs and Donald [10] proposed algorithms for computing the regions from which a sensor has unobstructed or partially obstructed views of a target in a goal. They applied these algorithms to the *Error Detection and Recovery* problem [23] for recognizing whether a goal or failure region has been achieved. Khawaja et al. [55] proposes an algorithm for the automatic placement of camera and light. The algorithm, which applies a

generate-and-test approach, is trained with synthetic images generated from information about the properties of the material of the object to observe, contact information from a CAD model of an assembly, and a physical lighting model. Reed and Allen [77] presented a constraint-based sensor planning method for scene modeling. They integrated sensor imaging constraints, occlusion constraints, and sensor placement constraints to determine admissible viewpoint to improve the quality of the model.

Gu et al. [32] proposed a sensor placement approach for dimensional inspection. Their method is based on robustness measure computed from models for displacement errors and the quantization error. The robustness measure is computed as the ratio between a maximum permissible entity inspection variance, and the actual entity inspection variance. Sheng et al. [88] developed a CAD-based camera-planning system for dimensional inspection in automotive manufacturing. They find feasible viewpoints combining recursively a two vision-sensor-planning methods without considering kinematic constraints of the inspection robot. The kinematics constraints are integrated in a second step by assigning a kinematics performance measure to candidate viewpoints.

Miura and Ikeuchi [64] also use a robustness measure to evaluate feasible visual sensing strategies. Its predicted success probability is computed from the clearance for insertion tasks and a quantified uncertainty model of the sensor used during the execution of assembly tasks.

There are other researchers that use genetic algorithms for sensor placement solutions. Chen and Li [13, 14] proposed a method to plan model-based sensor tasks, mainly for industrial inspection. They construct an optimal *sensor placement graph* with a genetic algorithm that uses a min-max criterion, and then determine the shortest path for achieving the sensing operations over the graph. The nodes of the sensor placement graph represent viewpoints and the edges represent shortest collision-free paths between the viewpoints. The edges are labeled with weights that represent the corresponding distances. Olague and Mohr [72] use a multicellular genetic algorithm to solve the problem of determining where to place the cameras to minimize the error in the detection 3-D points in reconstruction tasks. They describe the camera errors as ellipsoids.

Active Sensing in robotics try to give answer to the next aspects: (1) where to locate the sensors, and (2) how to decide for actions, to minimize costs and maximize information gain. Typical tasks where active sensing is useful are tasks executed in less structured environments where the uncertainties are that important that they influence the task execution. Examples are: industrial robot tasks [30], mobile robot navigation in a known map [79], vision applications for active selection of camera parameters [20], reinforcement learning [98], etc. Denzler and Brown [18] use Shannon's information theory to select information-gathering actions to maximize information that reduce uncertainty and ambiguity about state estimation. They apply the approach to object recognition using an active camera for sequential gaze control and viewpoint selection.

Other related works for viewpoint selection in object recognition include [83, 9, 73].

1.7 Thesis Organization

This chapter provided an overview of the entire thesis, the research context and background information. The rest of this dissertation is organized as follows:

The next two chapters provide a description of a novel approach to the construction of preventive visual sensing strategies from descriptions of nominal assembly plans. The proposed approach is based on an analysis of the contact-state formation process produced by the sequence of assembly steps specified in the assembly plan. Chapter 2 describe the foundations of the analysis, and deals with the deduction of vision and force sensing requirements for the manipulated object of a current assembly task. Chapter 3 completes the description of the sensing analysis module by introducing a graph representation that codes the full sensing requirements, and presents a propagation scheme and a method for determining critical alignment constraints among objects in the environment of a task. The method propagates critical translational and rotational constraints by defining new dependency relations between the assembly parts.

Chapter 4 describes a proposed sensor planning strategy based in the computation of a success measure calculated from the representations of the task tolerance to errors in the configurations of the assembly parts, and the uncertainty in the measurement results obtained from an object localization tool. This success measurement describes the goodness of a set of potential viewpoints considered for position the sensor (a camera) for a visual feedback operation. Chapter 5 describes an active camera mechanism devised to execute the visual sensing tasks and propose a method to obtain full visual sensing strategies for complete nominal assembly plans. Finally, Chapter 6 summarizes the conclusions and contributions of this research, and provides suggestions for future work.

List of Tables

Chapter 5

Active Placement and Sensing Planning

To implement the visual sensing strategies proposed by the sensing analyzer and realize the sensor configurations prescribed by the sensor planner an active-camera mechanism had to be developed. The goals of this chapter are to describe a camera-on-hand mechanism developed and used for the present work and a present a proposed sensing planner to construct a total and definite sensing strategy for a complete assembly plan.

5.1 The Active Camera Mechanism

An active-camera mechanism composed by a pan-tilt computer-controllable camera and a Cartesian robot-arm was constructed to perform the visual sensing strategies generated by the sensing analyzer. As shown in Figure 5.1 in this mechanism, a camera is fixed to the robot hand. The robot moves the camera without rotation to achieve its target position. The pan-tilt mechanism of the camera is used achieve its target orientation.

Based in the above model, a program was implemented to move and re-orient the camera to commanded poses in world coordinates. It also actualize the calibration settings accordingly. If the commanded position is outside the working space of the robot, it tries to realize it by a projection that conserves the commanded orientation. The poses are specified by two 3-D points, one to define the new viewpoint (position for the optical center of the camera) and another to define the new viewing direction (see Figure 5.2).

Since the Cartesian manipulator just translates the camera, its working space is described by a rectangular parallelepiped. Such volume restrains the allowed viewpoint space for the active camera. This kinematic constraint together with the limits on the pan and tilt rotation angles of the camera determine the reachable configurations for the positioning mechanism.

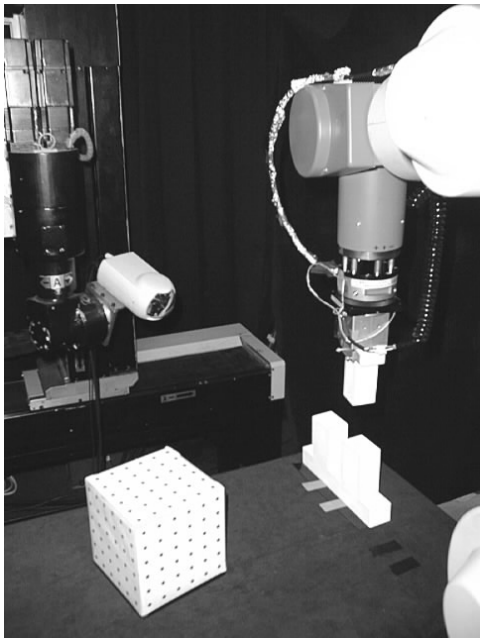


Figure 5.1: The Camera's Positioning Mechanism.

Since the working space of the robot is a convex polyhedral, an easy way of checking if the desired viewpoint is inside of it is through the signed distance of the point to every one of its forming planes. The distance will be positive in a case where the point is in the exterior half-space defined by the plane and pointed by its normal, negative if the point is in the interior half-space, and zero if it is on the plane.

The signed distance can be computed by direct substitution in the plane equation

$$a\,x + b\,y + c\,z + d = 0 \tag{5.1}$$

obtained from its normal vector $\mathbf{n} = (a, b, c)^T$ where $\|\mathbf{n}\| = 1$.

5.2 Active Camera Calibration

In the general case, calibration is the process of adjusting the parameters of a quantitative measurement or controlled instrument. The type of parameters and process are determined by the type of instrument and the goal of its calibration. Its implementation can be hard-wired, physically implemented, and/or performed by software.

In the case of the present robotic assembly system, all its components have to be calibrated. Every device has its own local coordinate system. The device measurement interfaces are defined with respect to its local reference frame. In the general case, the

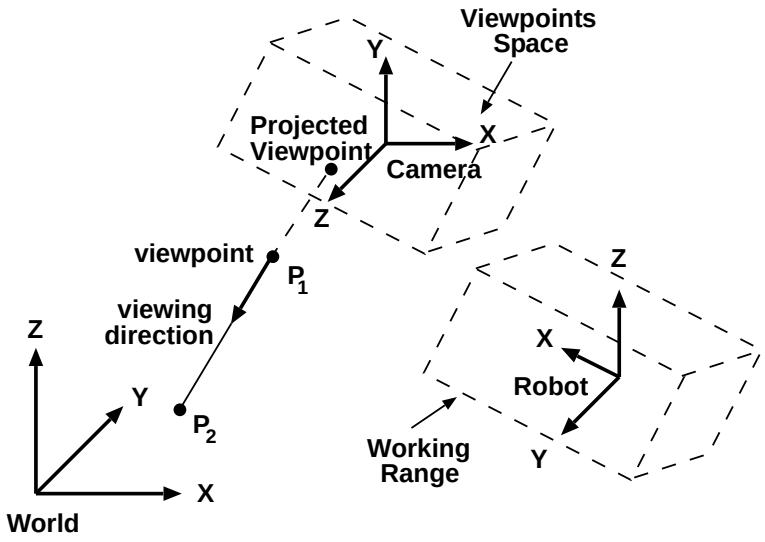


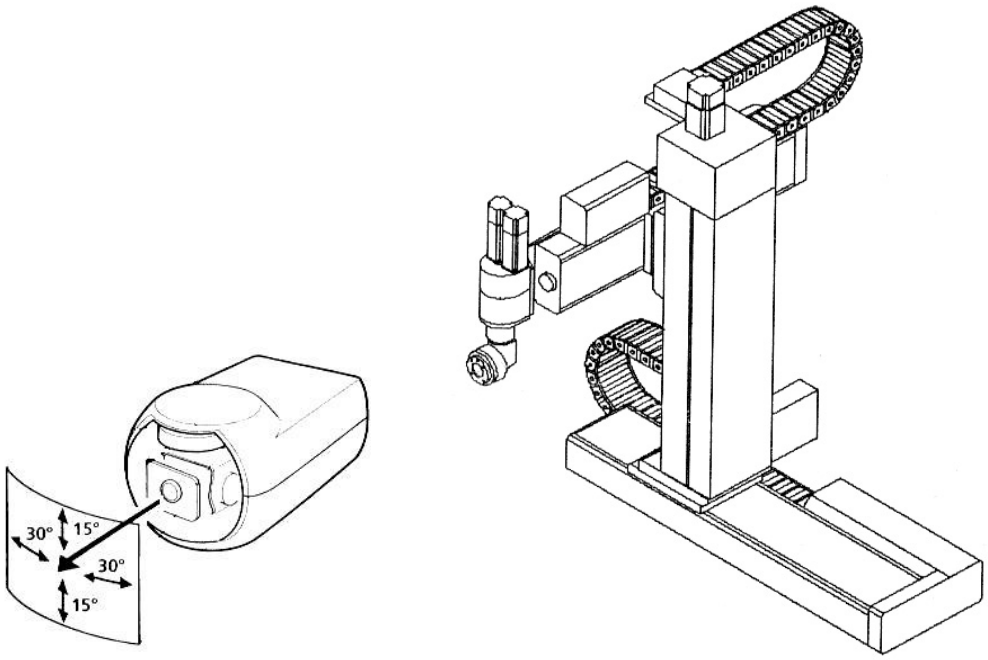
Figure 5.2: Positioning Process.

calibration process define an special (possibly completely new) coordinate system (the "world coordinate system"), to isolate the assembly plan from the specific local frames of the devices, and then every device is calibrated so that its readings can be interpreted and/or commands can be issued with respect to the same global reference frame.

The active camera for this work is composed by a pan-tilt computer-controlled camera (SONY model EVI-G20) and a Cartesian robot arm (DAIKIN ROBOTEC). The camera is fixed to the robot hand and is translated without rotation inside of the working range of the robot, the orientation of the active camera is completed by the pan-tilt mechanism of the camera.

The pan-tilt mechanism allows limited rotations around orthogonal axes almost perfectly aligned with respect to the vertical (pan: +/- 30 degrees) and horizontal (tilt: +/- 15 degrees) axes of the camera (see fig. 5.3a). The robot allows five degrees of motional freedom, three for translation and two for rotation (see fig. 5.3b). Since only translation will be allowed during camera positioning, the rotational freedom will be used only during the initial configuration to conveniently orient the camera for taking the maximum advantage of the camera's pan-tilt ranges.

In the context of computer vision, camera calibration is the process of determining a set of intrinsic and extrinsic parameters that will account for the geometrical and optical characteristics of the camera. To calibrate a camera the parametric model that describe its behavior has to be selected or developed. Currently, several camera models to capture the imaging properties of fixed-parameter and variable-parameter lenses have been proposed. In the fixed type, the parameters are constants. In the variable type, some of the parameters are dynamically adjusted by functions of the lens control settings.



(a) Sony EVI-G20 camera and its effective pan-tilt ranges.

(b) DAIKIN ROBOTEC Cartesian robot.

Figure 5.3: Active camera components.

For this dissertation, the camera is modeled by using an augmented Tsai's fixed camera model [106]. In addition to the conventional camera parameters proposed by Tsai, a set of additional parameters was included to adjust the global pose of the camera after the action of the robot and the camera's pan-tilt mechanism.

5.2.1 Calibration of the Fixed Camera Model

The starting point to get a camera calibration model for the present camera positioning mechanism is the Tsai model for fixed cameras. The Tsai's fixed camera model is a perspective-projection model based on the ideal pinhole camera originally proposed for using off-the-shelf TV cameras and lenses (see fig. 5.4). It consist of five intrinsic parameters that describe the image-formation process of the camera: the effective focal length of the pin hole camera (f), a coefficient of radial distortion (k_1), the coordinates of image center (C_x, C_y), and a scaling factor to compensate for any uncertainty in the ratio between the number of sensor elements on the CCD and the number of pixels in the camera's frame buffer in the x direction (s_x), and six extrinsic parameters that define the position (T_x, T_y, T_z) and orientation (α, β, γ) of the local 3-D camera coordinate frame.

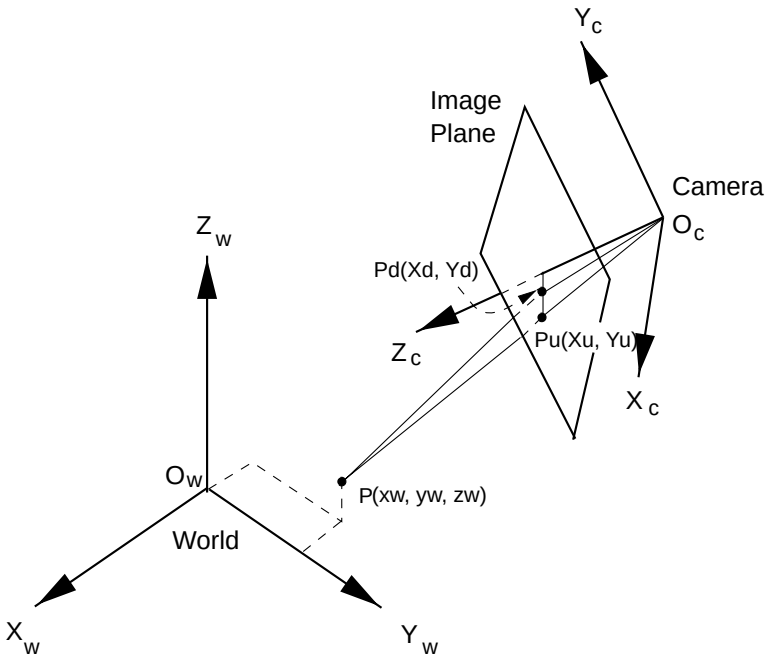


Figure 5.4: Tsai's fixed perspective-projection camera model with radial lens distortion.

The relationship of a point in world coordinates (x_w, y_w, z_w) and a image's point (X_f, Y_f) is governed by the next sequence of four transformations.

The first transformation establish the relation between the point in world coordinates to the points in camera coordinates (x_c, y_c, z_c) by the rigid body transformation as

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{5.2}$$

where

$$\mathbf{R} = \mathbf{R}_z(\gamma)\, \mathbf{R}_y(\beta)\, \mathbf{R}_x(\alpha) \tag{5.3}$$

is the 3 x 3 rotation matrix that is obtained as the product of basic rotation matrices around the X, Y, and Z axes of the world coordinate system.

The second transformation define the relation between the camera coordinates and the undistorted sensor coordinates (X_u and Y_u) by using the ideal perspective projection with the pinhole geometry

$$X_u = f \frac{x_c}{z_c} \tag{5.4a}$$

$$Y_u = f\,\frac{y_c}{z_c} \tag{5.4b}$$

The third transformation defines the relation between distorted (X_d, Y_d) and undistorted sensor coordinates under radial distortion as

$$X_u = X_d\,(1 + k_1\,\rho^2) \tag{5.5a}$$
$$Y_u = Y_d\,(1 + k_1\,\rho^2) \tag{5.5b}$$

where

$$\rho = \sqrt{X_d^2 + Y_d^2} \tag{5.6}$$

The fourth and last transformation, is to determine the point coordinates on the image plane from its distorted coordinates in the sensor as

$$X_f = d_x^{-1}\,X_d\,s_x + C_x \tag{5.7a}$$
$$Y_f = d_y^{-1}\,Y_d + C_y \tag{5.7b}$$

where d_x and d_y are the effective center-to-center distances between the camera's sensor elements in the x_c and y_c directions.

The data for the model calibration consist of 3-D world coordinates of a set of feature points on a calibration object (in mm) and the respective 2-D coordinates of the corresponding intensity points in the image took with the camera. For this work, a box of known dimensions was used as calibration tool (see Figure 5.1) and the non-coplanar calibration strategy proposed by Tsai from points on three of the six faces of the box. The world coordinate frame was fixed to one of the corners in the base of the calibration tool.

The calibration is posed as an optimization problem where the purpose is to minimize the sum of the squares of a system of nonlinear equations defined by the above described world-to-camera transformations. A modified Levenberg-Marquardt method with a Jacobian calculated by a forward-difference approximation is used to solve for the five intrinsic and six extrinsic unknown camera calibration parameters.

5.2.2 Calibration of the Camera Manipulator

The relation among the world, camera, and robot that carries the camera is illustrated in the Figure 5.5. In this section the interest is posed in the deduction of the 3-D rigid

transformation T_{wr} that defines the relation between the world and the robot coordinate systems.

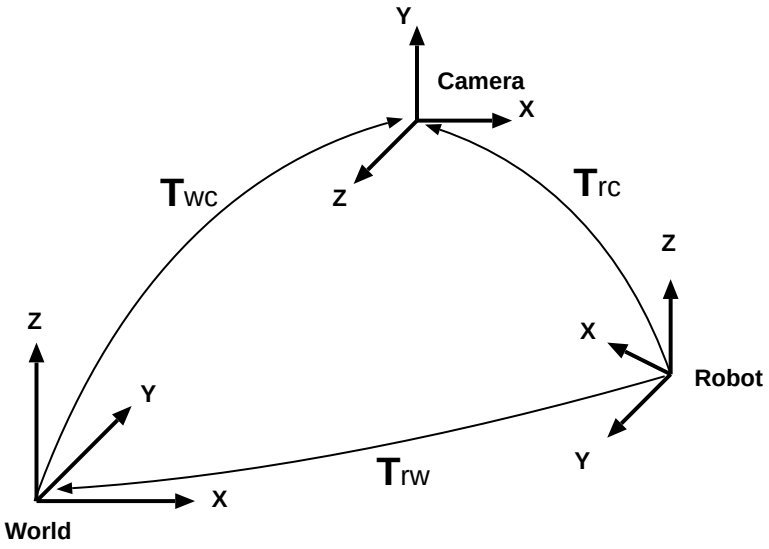


Figure 5.5: Coordinate Frames Relations.

The relation between the world and the camera coordinate systems is described by the 3-D rigid transformation matrix $\mathbf{T}_{wc}$ where a world point $\mathbf{p}_w$ is transformed into a camera point $\mathbf{p}_c$ as

$$\mathbf{T}_{wc}\,\mathbf{p}_w = \mathbf{p}_c. \tag{5.8}$$

$\mathbf{T}_{wc}$ is computed from the extrinsic parameters of the camera calibration model.

The robot-to-world relation is calibrated by following the next steps:

1. Translate the robot's hand to several positions $(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_n)$ taking an image of the calibration tool at each position. The calibration tool does not move. The images have to include the three calibration planes.
2. Calibrate the fixed camera calibration model for each of the images.
3. Generate a system of $n-1$ equations as explained next.
4. Solve the system for $\mathbf{T}_{rw}$.

The base transformation for the deduction of the relation is

$$\mathbf{p}_w = \mathbf{T}_{rw}\,\mathbf{p}_r \tag{5.9}$$

where $\mathbf{T}_{wr}$ is the representation of the robot-to-world rigid transformation as a 4x4 homogeneous matrix, $\mathbf{p}_w$ is a tri-dimensional vector representing a point in world coordinates, and $\mathbf{p}_r$ is the corresponding point in robot coordinates.

If there are two sets of corresponding points it can be solved for the elements of the matrix by solving the over-constrained system. However the selected strategy was to use the data from the calibration of the camera on different positions to generate the world points, and although their corresponding robot points can not be gotten their displacements from the robot commanded motion can be calculated. Then, instead of using the base transformation 5.9, the following was used:

$$\mathbf{p}_{wi} - \mathbf{p}_{wj} = \mathbf{T}_{rw} \left(\mathbf{p}_{ri} - \mathbf{p}_{rj} \right) = \mathbf{T}_{rw} \, \Delta_{rij} \tag{5.10}$$

where $\mathbf{p}_{wi}$ and $\mathbf{p}_{wj}$ are world points obtained from two calibrated camera positions, $\mathbf{p}_{ri}$ and $\mathbf{p}_{rj}$ are their corresponding robot points, and Δ_{rij} the displacement from the commanded robot motions.

The world points are computed from the extrinsic parameters of the camera calibration model as:

$$\mathbf{p}_{wi} = \mathbf{T}_{wci}^{-1} \cdot \mathbf{p}_{ci} = \mathbf{R}_{wci}^{-1} \cdot \mathbf{p}_{ci} - \mathbf{R}_{wci}^{-1} \cdot \mathbf{t}_{wci} \tag{5.11}$$

where $\mathbf{R}_{wci}$ and $\mathbf{t}_{wci}$ are respectively the 3x3 rotation matrix and 3-D translation vector that define the relation between the world and the ith camera frames. Using as camera point of reference its optical center $\mathbf{O}_{ci}$ (origin of its coordinate system) the expression is simplified to

$$\mathbf{p}_{wi} = -\mathbf{R}_{wci}^{-1} \, \mathbf{t}_{wci}. \tag{5.12}$$

Since the commanded robot motion are pure translations, Δ_{rij} is calculated as the difference of the robot positions used to get the equation 5.10.

The rotation and translation parameters of the robot-to-world relation are estimated by minimizing the sum of the square norms:

$$\| \mathbf{R}_{rw} \, \Delta_{rij} + \mathbf{t}_{rw} - (\mathbf{O}_{ci} - \mathbf{O}_{cj}) \|^2 \tag{5.13}$$

where $\mathbf{R}_{rw}$ and $\mathbf{t}_{rw}$ represent the rotation matrix and the translation vector that describe the target relation.

5.2.3 Calibration of Pan and Tilt Rotation Axes

After the camera is moved and re-oriented, some of its parameters have to be modified. The intrinsic and extrinsic parameters of the model that are affected depend on the nature of the movement.

This section deals with the problem of pan and tilt axes calibration. The problem is approached by establishing a series of simplifying assumptions and then relaxing some of them to reduce the generated error by such approximations. The goal is to define a set of functions parameterized by the pan and tilt commanded values and some possible additional variables that will minimize the error obtained from the results of the world-to-camera conversion process based on one calibrated camera pose, used as our model reference, and the results obtained by calibrating the camera in each possible camera pose.

The calibration data of reference is obtained with the camera in the home position for the pan and tilt mechanism. In this position, the pan and tilt values are closest to zero and simplifies the process of calibration by allowing to calibrate each rotation axis independently, as it will be shown later.

Two assumptions are maintained during all the experiments. First, that the intrinsic parameters are not affected by the camera's panning and tilting actions. This simplifies the analysis to the study of the effect on the extrinsic parameters, reducing the number of variables to include in the calibration process. The new goal is to minimize the difference among estimations of points in camera coordinates from their world coordinates obtained using by the new model and those obtained by the camera calibrated in each pose.

Second, that pan and tilt actions are performed as pure rotations. This is the assumption that reduces the problem to determining the descriptors of rotation axes.

The other two big assumptions are: first, that pan and tilt rotation axes pass through the optical center of the camera (origin of the camera coordinate frame). In this case, the optical center does not moves as a consequence of the pan and tilt actions and new commanded positions for the active camera can be reached solely by the manipulator irrespectively of the commanded change of attitude.

Second, that the pan and tilt rotation axes are aligned with those of the camera, the pan axis with the Y camera axis, and the tilt axis with the X camera axis. Then, allowing to describe the change in the camera orientation by a simple combination of two basic rotation matrices.

Te last two simplifying assumptions are eliminated later.

Next, comes a description of the formulations and results under the different sets of assumptions: first, using all the assumptions or what is further referred as *full alignment*;

second, by maintaining the alignment between axes, but eliminating the restriction of passing by the optical center of the camera; and finally, by considering that the axes would not be aligned with the camera axes, what is further referred as *full unalignment*.

A. Full Alignment

This is the case formulated above and the one originally used to develop the application to move the camera to a new commanded pose.

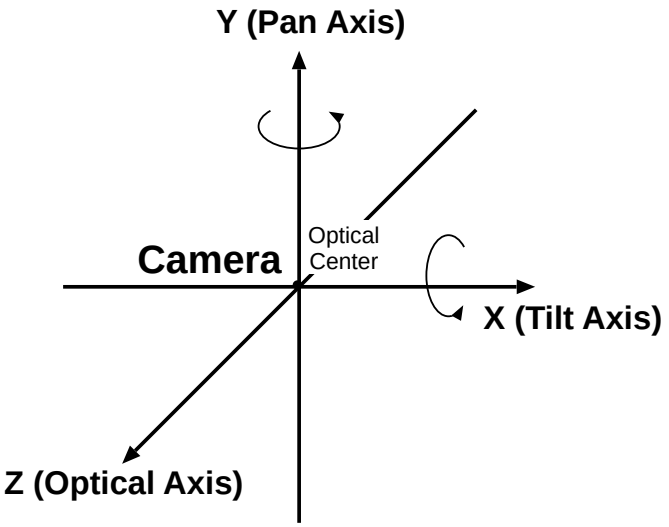


Figure 5.6: Configuration of pan and tilt rotational axes under the assumption of full alignment.

Since the pan axis is aligned to the Y camera axis, the tilt axis is aligned to the X camera axis, and both axes pass through the origin of the camera frame (as illustrated in Figure 5.6), its effect can be described as the single combination of basic rotation matrices

$$\mathbf{R}_y(p)\,\mathbf{R}_x(t) \tag{5.14}$$

which produce the following rigid transformation matrix

$$\begin{bmatrix} \cos(p) & \sin(p)\,\sin(t) & \cos(t)\,\sin(p) & 0 \\ 0 & \cos(t) & -\sin(t) & 0 \\ -\sin(p) & \cos(p)\,\sin(t) & \cos(p)\,\cos(t) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.15}$$

where p and t are the pan and tilt rotation angles, respectively.

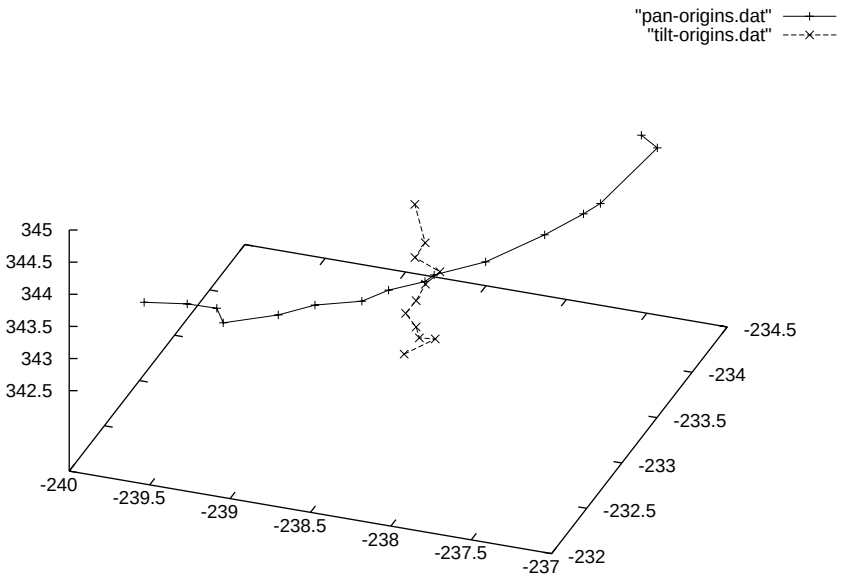


Figure 5.7: Motion of the optical center of the camera in world coordinates under panning and tilting motion.

In this case, there is no requirement to include new parameters to the camera calibration model in addition to the pan and tilt angles of the camera calibration data of reference.

B. Displaced Centers of Rotation

From a set of experiments it was realized that the optical center actually moves when the camera is panned and/or tilted. And as it can be observed in Figure 5.7 the motion is systematic, which makes evident that the pan and tilt rotation axes do not pass through it.

This section eliminates the assumption that the rotation axes pass through the optical center of the camera. Therefore, the origin of the camera coordinate system will move as consequence of the panning and tilting actions. Then, its new position has to be computed before determining the transition required by the robot to reach the commanded camera position.

Since the axes alignment assumption is maintained, the problem is reduced to determine the translation required to match the pan and tilt axes with its respective camera axes.

The translation required by the pan axis can be represented as the center of a circle on the XY camera, with coordinates $(x_p, 0, z_p)^T$. Similarly, the translation required by the tilt axis can be represented by the center of a circle on the ZY plane, with

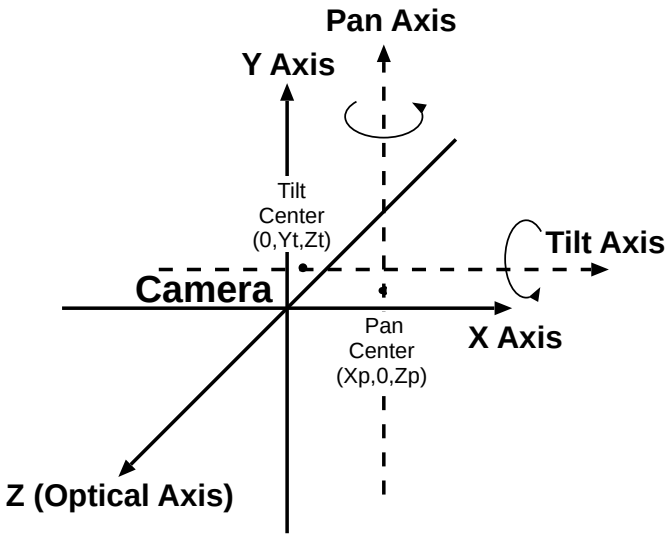


Figure 5.8: Configuration of pan and tilt rotational axes when they do not pass through the optical center of the camera.

coordinates $(0, y_t, z_t)^T$. Figure 5.8 illustrates the new situation.

To estimate these parameters and those of the next section the camera is calibrated while modifying the pan and tilt values. Since the initial idea was to calibrate the pan axis independently from the tilt axis, two sets of data were taken: one maintaining the tilt value as close to zero as possible, and the other doing the same with the pan values.

The methods of this section work with projections of the optical center of the camera when calibrated in the different poses. The origins are first converted to world coordinates by using the extrinsic parameters from the fixed camera calibration process and then are converted to camera coordinates using the reference calibration data.

Next, four methods are presented that are used to determine the searched points. The three first methods, solve for pan and tilt rotational reference points independently, then reducing the problem to two dimensions. The fourth method solves for the four unknowns simultaneously.

B.1 Best Circle Fitting

This method fits the best circle to the projection of the 3-D converted origins to the target planes. It does not take into account the explicit pan and tilt values. In the ideal case, it would not be necessary.

The equation of the circle applied to each case gives

$$(x - x_p)^2 + (z - z_p)^2 = r_p^2 \quad (5.16a)$$
$$(y - y_t)^2 + (z - z_t)^2 = r_t^2 \quad (5.16b)$$

where x_p and z_p are the unknown coordinates of the center of the circle for the pan rotation axis, y_t and z_t are the unknown coordinates of the center of the circle for the tilt rotation axis, and r_p and r_t are the radii for the pan and tilt axes, respectively. However, since the converted origins are supposed to describe a circle that pass through the origin of the reference camera, the coordinates of the searched centers can be used to compute the radii as

$$r_p^2 = x_p^2 + z_p^2 \quad (5.17a)$$
$$r_t^2 = y_t^2 + z_t^2 \quad (5.17b)$$

then, substituting equation 5.17 in equation 5.16 and simplifying

$$x_i^2 + z_i^2 - 2\,(x_i\,x_p + z_i\,z_p) = 0 \quad (5.18a)$$
$$y_j^2 + z_j^2 - 2\,(y_j\,y_t + z_j\,z_t) = 0 \quad (5.18b)$$

is gotten for each converted origin with coordinates $(x_i, 0, z_i)$ from the ith camera pose for pan axis calibration and $(0, x_j, z_j)$ from the jth camera pose for tilt axis calibration. These equations define a system that is solved by using a least-squared technique.

B.2 Center Averaging

In this case, a center of the circle described by the camera's panning and tilting is computed for each camera pose and the global pan and tilt centers are computed as an average by

$$C_{pan} = \left(\frac{\sum_{i=0}^{n} x_{pi}}{n}, 0, \frac{\sum_{i=0}^{n} z_{pi}}{n}\right) \quad (5.19a)$$
$$C_{tilt} = \left(0, \frac{\sum_{j=0}^{m} y_{tj}}{m}, \frac{\sum_{j=0}^{m} z_{tj}}{m}\right) \quad (5.19b)$$

for n camera poses for pan axis calibration and m camera poses for tilt axis calibration. The values for (x_{pi}, z_{pi}) and (y_{tj}, z_{ti}) are computes as follows:

The effect of rotating a 2-D point with respect to an arbitrary point (x_c, y_c) is computed as

$$\begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{5.20}$$

for a rotation of an angle θ. Applying equation 5.20 to the ith origin produces the following system of two implicit equations with two unknowns

$$x_c\,(1 - \cos(\theta_i)) + y_c\,\sin(\theta_i) = x'_i \tag{5.21a}$$
$$-x_c\,\sin(\theta_i) + y_c\,(1 - \cos(\theta_i)) = y'_i \tag{5.21b}$$

after solving the system and simplifying the solution is

$$x_c = \frac{y'_i + x'_i\,\cot(\frac{\theta_i}{2})}{2} \tag{5.22a}$$
$$y_c = \frac{x'_i - y'_i\,\cot(\frac{\theta_i}{2})}{2} \tag{5.22b}$$

The individual pan and tilt centers of rotation are then computed associating origin's Z coordinate with equation's X and origin's X coordinate with equation's Y for pan, and origin's Y with equation's X and origin's Z with equation's Y for tilt.

B.3 SVD Center Fitting

This method is based on equations 5.21 that here is shown in matrix form for the ith camera pose

$$\begin{bmatrix} (1 - \cos(\theta_i)) & \sin(\theta_i) \\ -\sin(\theta_i) & (1 - \cos(\theta_i)) \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix}. \tag{5.23}$$

When the converted origins are substituted on equation 5.23, linear systems of $2n$ equations with two unknowns for pan axis calibration and $2m$ equations for tilt are obtained.

The systems are solved by computing the pseudo-inverse of the matrix by using the singular value decomposition technique and then multiplying by the vector of converted origins' coordinates.

B.4 Simultaneous Center Fitting

The last method used to find the reference points for pan and tilt rotation solves for both reference points simultaneously. The other three methods solve for pan and tilt separately using calibrated camera poses on specially selected camera planes to eliminate the effect of the other unknowns.

The full transformation caused when any of the axes pass through the optical center of the camera is expressed as

$$\mathbf{T}(x_p, 0, z_p)\,\mathbf{R}_y(p)\,\mathbf{T}(-x_p, 0, -z_p)\,\mathbf{T}(0, y_t, z_t)\,\mathbf{R}_x(t)\,\mathbf{T}(0, -y_t, -z_t) \tag{5.24}$$

which here is shown in full matrix form

$$\begin{bmatrix} \cos(p) & \sin(p)\sin(t) & \cos(t)\sin(p) & x_p - x_p\cos(p) - \sin(p)\,(z_p - z_t + z_t\cos(t) + y_t\sin(t)) \\ 0 & \cos(t) & -\sin(t) & y_t - y_t\cos(t) + z_t\sin(t) \\ -\sin(p) & \cos(p)\sin(t) & \cos(p)\cos(t) & z_p + x_p\sin(p) - \cos(p)\,(z_p - z_t + z_t\cos(t) + y_t\sin(t)) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.25}$$

where p and t are the pan and tilt angles respectively.

Since the origins are used to perform the fitting, the expected position of the converted origins of the camera at different poses is computed as the translation vector of matrix 5.25.

The error, computed as the square distance between the expected and actual coordinates of the converted origins, is minimized by using the same modified Levenberg-Marquardt method used before to calibrate the fixed camera calibration model.

C. Full Unalignment

Finally, this section eliminates the assumption of alignment between the pan and tilt axes and the camera's Y and X axes. The new situation is illustrated in Figure 5.9.

There are several ways in which an arbitrary axis can be represented. We selected to represent its orientation by two angles and its position by a point.

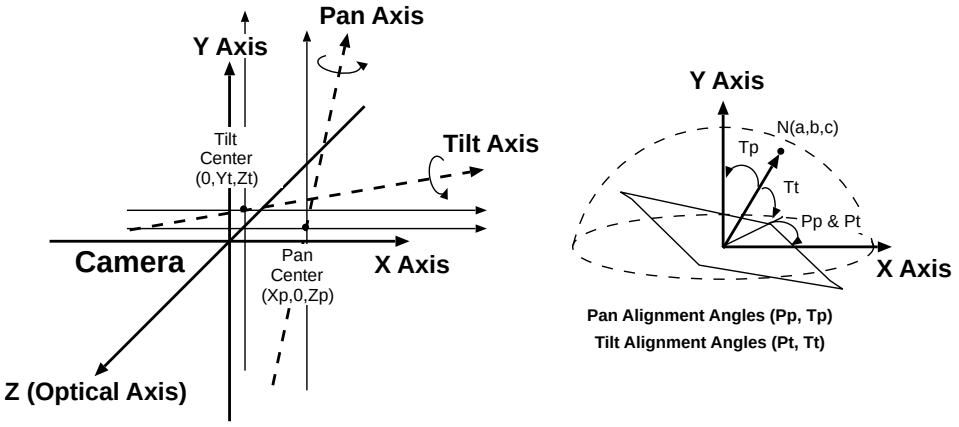


Figure 5.9: Configuration of pan and tilt rotational axes when they do not pass through the optical center of the camera and are not aligned with the axes of the camera's coordinate system.

The points are defined as those where the axes intersect specific planes in the camera reference: the ZX plane for pan and the YZ plane for tilt, then, presenting similar elements to those of the last section.

The angles are defined so that they can align the rotation axis with specific camera axes by a combination of two rotations. In the case of the pan axis, it is aligned with the Y camera axis by first rotating it an angle p_p with respect to the Y camera axis and then rotating it an angle t_p with respect to the Z camera axis. And in the case of the tilt axis, it is aligned with the X camera axis by first rotating it an angle p_t with respect to the Y camera axis and then rotating it an angle $-t_t$ with respect to the Z camera axis (see Figure 5.9).

Since the simultaneous fitting process of pan and tilt unknowns do not showed better results, as will be shown later, the eight unknowns were fitted separately.

The new transformation matrix for that represents the effect of panning the camera is computed as

$$\mathbf{R}_y(p_p)\,\mathbf{R}_z(-t_p)\,\mathbf{T}(x_p,0,z_p)\,\mathbf{R}_y(p)\,\mathbf{T}(-x_p,0,-z_p)\,\mathbf{R}_z(t_p)\,\mathbf{R}_y(-p_p) \tag{5.26}$$

which evaluated and simplified gives

$$
\begin{aligned}
P[0][0]: &\ \cos(p)\left(\cos(p_p)^2\cos(t_p)^2+\sin(p_p)^2\right)+\cos(p_p)^2\sin(t_p)^2\\
P[1][0]: &\ -(((-1+\cos(p))\cos(p_p)\cos(t_p)+\sin(p)\sin(p_p))\sin(t_p))\\
P[2][0]: &\ -(\cos(p_p)^2\cos(t_p)\sin(p))-\cos(t_p)\sin(p)\sin(p_p)^2-\sin(\tfrac{p}{2})^2\sin(2\,p_p)\sin(t_p)^2\\
P[0][1]: &\ (-((-1+\cos(p))\cos(p_p)\cos(t_p))+\sin(p)\sin(p_p))\sin(t_p)\\
P[1][1]: &\ \cos(t_p)^2+\cos(p)\sin(t_p)^2\\
P[2][1]: &\ (\cos(p_p)\sin(p)+(-1+\cos(p))\cos(t_p)\sin(p_p))\sin(t_p)\\
P[0][2]: &\ \cos(p_p)^2\cos(t_p)\sin(p)+\cos(t_p)\sin(p)\sin(p_p)^2-\sin(\tfrac{p}{2})^2\sin(2\,p_p)\sin(t_p)^2\\
P[1][2]: &\ -((\cos(p_p)\sin(p)-(-1+\cos(p))\cos(t_p)\sin(p_p))\sin(t_p))\\
P[2][2]: &\ \cos(p)\left(\cos(p_p)^2+\cos(t_p)^2\sin(p_p)^2\right)+\sin(p_p)^2\sin(t_p)^2\\
P[0][3]: &\ -(\cos(p_p)\cos(t_p)\left(-x_p+x_p\cos(p)+z_p\sin(p)\right))+\left(z_p-z_p\cos(p)+x_p\sin(p)\right)\sin(p_p)\\
P[1][3]: &\ (-x_p+x_p\cos(p)+z_p\sin(p))\sin(t_p)\\
P[2][3]: &\ \cos(p_p)\left(z_p-z_p\cos(p)+x_p\sin(p)\right)+\cos(t_p)\left(-x_p+x_p\cos(p)+z_p\sin(p)\right)\sin(p_p).
\end{aligned}
\tag{5.27}
$$

And, the new transformation matrix for that represents the effect of tilting the camera is computed as

$$
\mathbf{R}_y(p_t)\,\mathbf{R}_z(t_t)\,\mathbf{T}(0,y_t,z_t)\,\mathbf{R}_x(t)\,\mathbf{T}(0,-y_t,-z_t)\,\mathbf{R}_z(-t_t)\,\mathbf{R}_y(-p_t) \tag{5.28}
$$

which evaluated and simplified gives

$$
\begin{aligned}
T[0][0]: &\ \cos(t)\sin(pt)^2+\cos(pt)^2\left(\cos(tt)^2+\cos(t)\sin(tt)^2\right)\\
T[1][0]: &\ -(\cos(tt)\left(\sin(pt)\sin(t)+\cos(pt)\left(-1+\cos(t)\right)\sin(tt)\right))\\
T[2][0]: &\ -(\cos(tt)^2\sin(2\,pt)\sin(\tfrac{t}{2})^2)-\sin(t)\sin(tt)\\
T[0][1]: &\ \cos(tt)\left(\sin(pt)\sin(t)-\cos(pt)\left(-1+\cos(t)\right)\sin(tt)\right)\\
T[1][1]: &\ \cos(t)\cos(tt)^2+\sin(tt)^2\\
T[2][1]: &\ \cos(tt)\left(\cos(pt)\sin(t)+\left(-1+\cos(t)\right)\sin(pt)\sin(tt)\right)\\
T[0][2]: &\ -(\cos(tt)^2\sin(2\,pt)\sin(\tfrac{t}{2})^2)+\sin(t)\sin(tt)\\
T[1][2]: &\ -(\cos(tt)\left(\cos(pt)\sin(t)-\left(-1+\cos(t)\right)\sin(pt)\sin(tt)\right))\\
T[2][2]: &\ \cos(pt)^2\cos(t)+\sin(pt)^2\left(\cos(tt)^2+\cos(t)\sin(tt)^2\right)\\
T[0][3]: &\ \sin(pt)\left(zt-zt\cos(t)-yt\sin(t)\right)+\cos(pt)\left(-yt+yt\cos(t)-zt\sin(t)\right)\sin(tt)\\
T[1][3]: &\ \cos(tt)\left(yt-yt\cos(t)+zt\sin(t)\right)\\
T[2][3]: &\ \cos(pt)\left(zt-zt\cos(t)-yt\sin(t)\right)+\sin(pt)\left(yt-yt\cos(t)+zt\sin(t)\right)\sin(tt).
\end{aligned}
\tag{5.29}
$$

As it can be observed, the expressions has became longer and more complex.

The first intention was to use the same fitting method used in the fourth method of the last section, directly, by only changing the equations. However, the new system demonstrated to suffer of many local minima and since the solution method performs a process of successive approximations, it did not give good values.

Then it was decided to solve the problem in two steps: first, to fit planes to the set of converted origins, get the axes' orientations, and then use the same strategy of the fourth method of the last section to solve for the axes' reference points by using the transformation matrices 5.27 and 5.29.

The plane is fitted with equation

$$a\,x + b\,y + c\,z + d = 0 \tag{5.30}$$

with normal $\mathbf{n} = (a, b, c)^T$ where $\|\mathbf{n} = 1$ by using a least square technique.

The orientation angles of the axes are calculated from the components of the normal of the fitted plane. For the pan axis as

$$p_p = \arctan(\sin(p_p)/\cos(p_p)) = \arctan(-c/a) \tag{5.31a}$$
$$t_p = \arccos(n \cdot \bar{y}) = \arccos(b) \tag{5.31b}$$

where $\bar{y}$ is a unit vector in the direction of the Y camera axis $(0, 1, 0)^T$.

And for the tilt axis as

$$p_t = \arctan(\sin(p_t)/\cos(p_t)) = \arctan(-c/a), \tag{5.32a}$$
$$t_t = \frac{\pi}{2} - \arccos(n \cdot \bar{y}) = \frac{\pi}{2} - \arccos(b). \tag{5.32b}$$

5.2.4 Experimental Results

This section presents the effects of calibration of the pan and tilt rotation axes on the accuracy of the active camera calibration model.

Since the initial idea was to calibrate the pan axis independently from the tilt axis, two sets of data were taken: one panning the camera while maintaining the tilt value as close to zero as possible, and the other tilting the camera while maintaining the pan values as close to zero as possible.

The methods of the section that determine centers of rotation work with projections of the optical center of the camera when calibrated in the different poses. The origins are first converted to world coordinates by using the extrinsic parameters from the fixed camera calibration process and then are converted to camera coordinates using the reference calibration data.

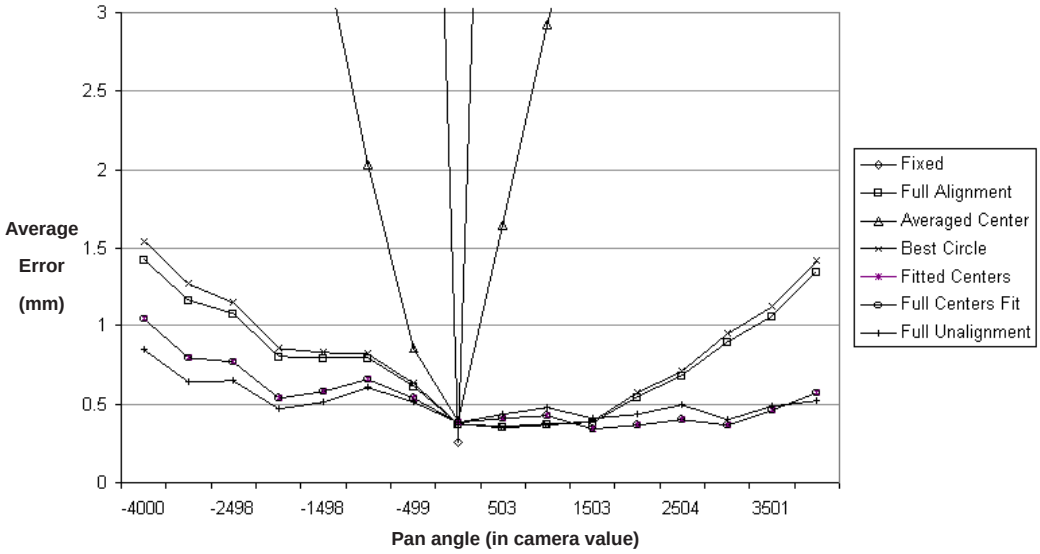


Figure 5.10: Precision after Calibration of Pan and Tilt Rotation Axes when Camera is Panned.

In the case of the method for full unalignment, the number of points had to be augmented to get better estimates in the plane fitting process. This because small deviations with respect to the true alignment is scaled by the distance from the camera, and as a consequence produce bigger errors. To increase the number of points, instead of using only the origins, a set of uniformly separated points on the optical axis (Z axis) of the camera were used. This measure worked out very well.

The accuracy obtained after the calibration process is shown in Figure 5.10 for the camera poses used for pan calibration, and in Figure 5.11 for camera poses used for tilt calibration.

The plotted values were computed as follows:

1. A big set of randomly distributed 3-D points in world coordinates P_w is created.
2. Each point in P_w is converted to the camera frame of reference for the active camera calibration model, giving P_c.
3. For each camera pose $\mathbf{c}_i$:
   (a) Convert a point $\mathbf{p}_{wj}$ from P_w to the coordinates in $\mathbf{c}_i$, by directly using its pre-computed fixed camera calibration data, getting $\mathbf{p}^i_{wj}$.
   (b) Convert its correspondent point $\mathbf{p}_{cj}$ from P_c to the coordinates in $\mathbf{c}_i$ by using the active camera calibration model, getting $\mathbf{p}^i_{cj}$. This conversion applies the transformation matrices described earlier for each case, but with negated pan

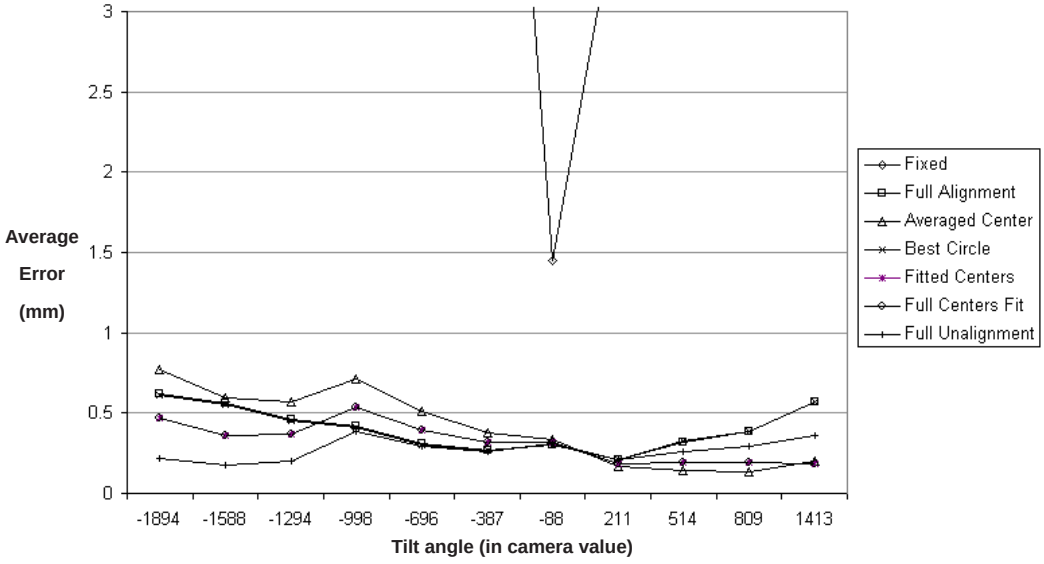


Figure 5.11: Precision after Calibration of Pan and Tilt Rotation Axes when Camera is Tilted.

and tilt angles. The negated angles are explained as the effect of rotating the coordinate frame and not the points.

(c) Compute the error as the absolute distance between $\mathbf{p}^i_{wj}$ and $\mathbf{p}^i_{cj}$ and add it to the total error for $\mathbf{c}_i$.

4. Compute the average error for each camera pose by dividing the total error by the number of points.

In order to judge which method gives better results, two factors has to be observed. One is the magnitude of the error and the other is the tendency of its growing while getting away from the reference camera pose, that was selected as close to $(0, 0)$ for the pan and tilt values, respectively.

It can be observed, that the error tends to increase with the distance from the reference, as was expected. This is because the models do not consider all the factors and in the best case are only approximations. The better the approximation the smaller the expected error. As an extreme illustration of this, it was included the plot of the average error when using only the reference data (with legend *fixed*). The error grows so fast that only a small portion appears in the figures. This obviously wrong approximation shows the worst behavior under the pan and tilt actions.

Then, the accuracy of the model grows when the constraining assumptions are eliminated. An exception is the case of where the centers of pan and tilt rotation are computed as an average. This case presents bigger errors and faster growing tendency

than the case of full alignment. The reason for this is that this kind of computation is an easy prey of noise, that in this case pulls out the position of the centers. When the commanded rotation is small, a small error in the origin position computation can finish fitting circles of huge radius.

Another thing that it can be observed in the figure is that the results obtained by calibrating the pan and tilt independently or doing it at the same time are practically the same, even when different fitting methods where used.

The method that gave the best (minimum) error and tendency behavior was the least constrained; that which allows the pan and tilt axes be unaligned and do not pass through the optical center of the camera, especially in the case of the tilt rotation axis.

However, most of the methods present a reasonably good results. This is because the camera that was used has its pan and tilt axes almost perfectly aligned with the camera axes, as could be noted from the plane fitting results, and they actually pass near the optical center. The difference in the results would be more evident in the case of some other devices.

5.3 Active Camera Adjustment

This section determines the correction in the controllable variables of the active camera mechanism required to reach a new commanded position and orientation. The new camera pose is given by two points in world coordinates, one being the viewpoint (position for the camera's optical center) and the other to be used for computing the new camera attitude.

First the method computes the changes that will produce the new orientation and then those that will correct the position of the viewpoint.

5.3.1 Correcting the Camera's Attitude

In the proposed positioning mechanism, the final attitude is determined solely by the camera. Therefore, new values for the pan and tilt angles have to be calculated after a new camera pose is required. To do this, the current viewing direction and the new one are converted to camera coordinates, and the rigid transformation $\mathbf{T}_{pt}$ that will align both vectors is deduced from

$$\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = \mathbf{T}_{pt} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{5.33}$$

where $(x_d, y_d, z_d)^T$ and $(x_c, y_c, z_c)^T$ are the camera coordinates of the new desired and current viewing directions, respectively.

The computation of the values for the pan and tilt rotation angles come from the form of $\mathbf{T}_{pt}$ which, as was described before, is based on a valid set of assumptions.

Next the formulations and solutions for the first two cases are presented: **(1) full alignment:** when the pan and tilt axes are aligned with the camera axes and pass through its optical center, and **(2) Displaced centers of rotation:** when the axes are aligned but do not pass through the optical center. The third case, where the axes are not aligned is not included because its formulation and solution includes very large equations and as it was shown in the experiments, the axes are almost perfectly aligned for the camera used, so that its benefit is marginal.

A. Full Alignment

In this case, $\mathbf{T}_{pt}$ is computed as the rotation matrix

$$\mathbf{R}_{pt} = \mathbf{R}_y(\theta)\,\mathbf{R}_x(\phi) = \begin{bmatrix} \cos(\theta) & \sin(\theta)\,\sin(\phi) & \sin(\theta)\,\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ -\sin(\theta) & \cos(\theta)\,\sin(\phi) & \cos(\theta)\,\cos(\phi) \end{bmatrix} \tag{5.34}$$

obtained from the pan angle θ and the tilt angle ϕ.

When the current and new viewing direction are represented by arbitrary vectors, the following system of three equations is derived

$$\begin{aligned} x_d &= x_c\,\cos(\Delta\theta) + y_c\,\sin(\Delta\theta)\,\sin(\Delta\phi) + z_c\,\sin(\Delta\theta)\,\cos(\Delta\phi) && \text{(5.35a)} \\ y_d &= y_c\,\cos(\Delta\phi) - z_c\,\cos(\Delta\phi) && \text{(5.35b)} \\ z_d &= -x_c\,\sin(\Delta\theta) + y_c\,\cos(\Delta\theta)\,\sin(\Delta\phi) + z_c\,\cos(\Delta\theta)\,\cos(\Delta\phi) && \text{(5.35c)} \end{aligned}$$

where $\Delta\theta$ and $\Delta\phi$ are the correction pan and tilt angles, respectively.

Solving the above system we get

$$\begin{aligned} \sin(\Delta\theta) &= \frac{-z_d x_c \pm x_d\sqrt{y_c^2 + z_c^2 - y_d^2}}{x_d^2 + z_d^2} && \text{(5.36a)} \\ \cos(\Delta\theta) &= \frac{x_d x_c \pm z_d\sqrt{y_c^2 + z_c^2 - y_d^2}}{x_d^2 + z_d^2} && \text{(5.36b)} \\ \Delta\theta &= atan2(\sin(\Delta\theta), \cos(\Delta\theta)) && \text{(5.36c)} \end{aligned}$$

and

$$\sin(\Delta\phi) = \frac{-y_d z_c \pm y_c\sqrt{y_c^2 + z_c^2 - y_d^2}}{y_c^2 + z_c^2} \tag{5.37a}$$

$$\cos(\Delta\phi) = \frac{y_d y_c \pm z_c\sqrt{y_c^2 + z_c^2 - y_d^2}}{y_c^2 + z_c^2} \tag{5.37b}$$

$$\Delta\phi = atan2(\sin(\Delta\phi), \cos(\Delta\phi)) \tag{5.37c}$$

Then, the new pan and tilt angles to be commanded to the camera are

$$\theta_{new} = \theta_{old} + \Delta\theta \tag{5.38a}$$

$$\phi_{new} = \phi_{old} + \Delta\phi. \tag{5.38b}$$

An alternative to the above general solution is to compute directly the absolute values for the pan and tilt angles to be commanded aligning an specific vector to the desired new viewing direction. Since the camera model assumes that the optical axis of the camera is aligned with the camera's Z axis, the vector $(0, 0, 1)^T$ can be used in place of the current viewpoint orientation. In this case, the system to solve is

$$x_d = \sin(\theta)\ \cos(\phi) \tag{5.39a}$$

$$y_d = -\sin(\phi) \tag{5.39b}$$

$$z_d = \cos(\theta)\ \cos(\phi) \tag{5.39c}$$

which gives the following solution

$$\sin(\theta) = \pm\frac{x_d}{\sqrt{x_d^2 + z_d^2}} \tag{5.40a}$$

$$\cos(\theta) = \pm\frac{z_d}{\sqrt{x_d^2 + z_d^2}} \tag{5.40b}$$

$$\theta = atan2(\sin(\theta), \cos(\theta)) \tag{5.40c}$$

and

$$\begin{aligned} \sin(\phi) &= -y_d && (5.41a) \\ \cos(\phi) &= \pm\sqrt{x_d^2 + z_d^2} && (5.41b) \\ \phi &= atan2(\sin(\phi), \cos(\phi)) && (5.41c) \end{aligned}$$

these results can be easily verified by simple substitution on the above general equations.

Before sending the command to the camera, the angles have to be checked against the kinematic constraints defined by the physical pan and tilt rotational ranges for the specific camera.

After the correction and under the present assumptions, the extrinsic parameters of the fixed camera calibration model can be recomputed as

$$\begin{aligned} \mathbf{R}'_{wc} &= \mathbf{R}_y(-\theta)\,\mathbf{R}_x(-\phi)\,\mathbf{R}_{wc} && (5.42a) \\ \mathbf{t}'_{wc} &= \mathbf{R}_y(-\theta)\,\mathbf{R}_x(-\phi)\,\mathbf{t}_{wc} && (5.42b) \end{aligned}$$

where $\mathbf{R}'_{wc}$ and $\mathbf{t}'_{wc}$ are the new rotation matrix and translation vector, respectively.

Since in this case the optical center does not move, its position in world coordinates to be used as the current viewpoint $\mathbf{v}_c$ is obtained by

$$\mathbf{v}_c = -\mathbf{R}_{wc}^{-1}\,\mathbf{t}_{wc} \tag{5.43}$$

B. Displaced Centers of Rotation

In this case, the method solves for the alignment of the Z camera axis with the new commanded viewing direction in the camera coordinates of the camera calibration reference. The problem is formulated as

$$\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = \mathbf{T}(x_p, 0, z_p)\,\mathbf{R}_y(\theta)\,\mathbf{T}(-x_p, 0, -z_p)\,\mathbf{T}(0, y_t, z_t)\,\mathbf{R}_x(\phi)\,\mathbf{T}(0, -y_t, -z_t) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.44}$$

that generates the following system of three equations

$$\begin{aligned} x_d &= x_p - x_p\,\cos(\theta) + \sin(\theta)\,(-z_p + z_t + \cos(\phi) - z_t\,\cos(\phi) - y_t\,\sin(\phi)) && (5.45a)\\ y_d &= y_t - y_t\,\cos(\phi) + (-1 + z_t)\,\sin(\phi) && (5.45b)\\ z_d &= z_p + x_p\,\sin(\theta) + \cos(\theta)\,(-z_p + z_t + \cos(\phi) - z_t\,\cos(\phi) - y_t\,\sin(\phi)) && (5.45c) \end{aligned}$$

After solving for ϕ, by using equation 5.45b and a set of trigonometric identities, and simplifying the expression, ϕ is computed as

$$\phi = \arccos\left(\frac{-y_d\,y_t + y_t^2 \pm (-1 + z_t)\,\sqrt{-y_d^2 + 2\,y_d\,y_t + (-1 + z_t)^2}}{y_t^2 + (-1 + z_t)^2}\right). \tag{5.46}$$

Since from equation 5.45b

$$\sin(\phi) = \frac{y_d - y_t + y_t\,\cos(\phi)}{-1 + z_t} \tag{5.47}$$

then, substituting the result of equation 5.46 in equation 5.47 and simplifying the expressions

$$\begin{aligned} \sin(\phi) &= \frac{(-1 + z_t)\,(y_d - y_t) \pm y_t\,\sqrt{-y_d^2 + 2\,y_d\,y_t + (-1 + z_t)^2}}{y_t^2 + (-1 + z_t)^2} && (5.48a)\\ \cos(\phi) &= \frac{-y_t\,(y_d - y_t) \pm (-1 + z_t)\,\sqrt{-y_d^2 + 2\,y_d\,y_t + (-1 + z_t)^2}}{y_t^2 + (-1 + z_t)^2} && (5.48b)\\ \phi &= atan2(\sin(\phi), \cos(\phi)) && (5.48c) \end{aligned}$$

Now, solving and simplifying from equations 5.45a and 5.45c, the result is

$$\begin{aligned} \sin(\theta) &= \frac{x_p\,(z_d - z_p) + \alpha\,(-x_d + x_p)}{x_p^2 + \alpha^2} && (5.49a)\\ \cos(\theta) &= \frac{x_p\,(-x_d + x_p) - \alpha\,(z_d - z_p)}{x_p^2 + \alpha^2} && (5.49b)\\ \alpha &= z_p - z_t + (-1 + z_t)\,\cos(\phi) + z_t\,\sin(\phi) && (5.49c)\\ \theta &= atan2(\sin(\theta), \cos(\theta)) && (5.49d) \end{aligned}$$

where after substituting the values for $\cos(\phi)$ and $\sin(\phi)$ of equation 5.48 in α equation and simplifying gives

$$\alpha = z_p - z_t \pm \sqrt{-y_d^2 + 2\, y_d\, y_t + (-1 + z_t)^2}. \tag{5.50}$$

Again, before sending the command to the camera, the obtained angles have to be checked against the kinematic constraints defined by the physical pan and tilt rotational ranges for the specific camera.

After the correction and under the current assumptions, the extrinsic parameters of the fixed camera calibration model can be recomputed as

$$\begin{aligned} \mathbf{R}'_{wc} &= \mathbf{R}_y(-\theta)\, \mathbf{R}_x(-\phi)\, \mathbf{R}_{wc} && \text{(5.51a)} \\ \mathbf{t}'_{wc} &= \mathbf{R}_y(-\theta)\, \mathbf{R}_x(-\phi)\, \mathbf{t}_{wc} + \mathbf{R}_y(-\theta)\, (\mathbf{I} - \mathbf{R}_x(-\phi))\, \mathbf{T}(0, y_t, z_t) + && \text{(5.51b)} \\ & \quad (\mathbf{I} - \mathbf{R}_y(-\theta))\, \mathbf{T}(x_p, 0, z_p) \end{aligned}$$

where $\mathbf{I}$ is the 3x3 identity matrix.

This time, as result of the panning and/or tilting the camera, the optical center of the camera moves, and this motion has to be considered when determining the robot motion.

The old optical center and origin of the camera coordinate frame moves to a position with respect to the new coordinate system computed as

$$\mathbf{t}'_{wc} - \mathbf{R}'_{wc}\, \mathbf{R}_{wc}^{-1}\, \mathbf{t}_{wc} = \mathbf{R}_y(-\theta)\, (\mathbf{I} - \mathbf{R}_x(-\phi))\, \mathbf{T}(0, y_t, z_t) + (\mathbf{I} - \mathbf{R}_y(-\theta))\, \mathbf{T}(x_p, 0, z_p). \tag{5.52}$$

The position in world coordinates of the new origin of the camera system after the attitude correction, and the new current viewpoint $\mathbf{v}_c$ before the robot correction is computed as

$$\begin{aligned} \mathbf{v}_c &= -\mathbf{R}_{wc}'^{-1}\, \mathbf{t}'_{wc} \\ &= \mathbf{R}_{wc}^{-1}\, (\mathbf{T}(0, y_t, z_t) - \mathbf{t}_{wc}) + \mathbf{R}_{wc}^{-1}\, \mathbf{R}_x^{-1}(-\phi)\, (\mathbf{T}(x_p, 0, z_p) - \mathbf{T}(0, y_t, z_t)) - \\ &\quad \mathbf{R}_{wc}^{-1}\, \mathbf{R}_x^{-1}(-\phi)\, \mathbf{R}_y^{-1}(-\theta)\, \mathbf{T}(0, y_t, z_t). \end{aligned} \tag{5.53}$$

5.3.2 Correcting the Viewpoint Position

In the described positioning system, the robot completes the camera's new pose by translation about its local coordinate system. To determine the correction required and

the values for the new position to command the robot, the problem is worked out in the world context.

The desired viewpoint $\mathbf{v}_d$ is already given as input to the system in such reference and the current viewpoint $\mathbf{v}_c$ is calculated as shown before.

The required translation in world coordinates $\Delta\mathbf{t}_w$ is

$$\Delta\mathbf{t}_w = \mathbf{v}_d - \mathbf{v}_c = \mathbf{T}_{rw}\,\Delta\mathbf{t}_r \tag{5.54}$$

where $\mathbf{T}_{rw}$ is the transformation matrix from robot to world coordinates and

$$\Delta\mathbf{t}_r = \mathbf{r}_s - \mathbf{r}_c \tag{5.55}$$

is the required translation in robot coordinates, $\mathbf{r}_s$ is the searched robot position, and $\mathbf{r}_c$ is the current robot position. Thus, to reach the new viewpoint position, the new commanded robot position is obtained as

$$\mathbf{r}_s = \mathbf{R}_{wc}^{-1}(\mathbf{v}_d + \mathbf{R}_{wc}^{-1}\,\mathbf{t}_{wc}) + \mathbf{r}_c \tag{5.56}$$

After the robot correction, the camera's attitude is not modified leaving the extrinsic parameters of the camera calibration model for rotation unchanged, but the extrinsic parameters for translation changes to

$$\mathbf{t}'_{wc} = \mathbf{t}_{wc} - \mathbf{R}_{wc}\,\mathbf{R}_{rw}\,\Delta\mathbf{t}_r. \tag{5.57}$$

5.4 Preventive Visual Strategies for Robotic Assemblies

After determining the assembly steps that require of visual feedback operations and knowing how to quantify a predicted success probability (Psp) for each viewpoint in the viewing sphere of objects to be observed during the execution of the assembly plan, to complete the preventive visual sensing strategy lacks to decide the sequence of movements for the active vision sensor.

Each task in the new assembly plan includes a set of objects that have to be observed to determine a potential adjustment in the configuration of the manipulated object of the task. As explained in Chapter 4, each of these objects will be placed in the center

of a discretized viewing sphere and for each viewpoint localized in the center of each tessel of this sphere a Psp value will be computed.

The obtainment the sequence of movements for the sensor is a classic example of a multi-objective optimization problem, since its solution pretends to maximize the probability of success of the assembly operations and at the same time minimize the displacements of the optical sensor. However, during the execution of the plan, the best viewpoint for observing an object during a task could be far from the best viewpoint to observe an object for the following task. Inclusive, sometimes during the same task, several objects have to be localized in order to determine preventive adjustments in the manipulated object, and best viewpoints could also be far from each other.

Other factors to consider in the construction of a criteria for determining the sequence of viewpoints for the sensor are those related with the reachability of the viewpoints, and also the possible variability in the real Psp values when a viewpoint is expected to suffer a high level of occlusion. The first factor is related to the kinematic constraints of the active sensor that limit the possible camera configurations and do not allow to reach the position and orientation established by the viewpoint. The second factor is related to the accuracy of the computed Psp value associated with a viewpoint; this value is obtained from an uncertainty model of the sensor experimentally quantified. The quantification of the parametric model of uncertainty is realized through localization of isolated objects and using a technique that is robust to some degree of occlusion, but if the level of occlusion is high the Psp values will degrade a lot.

To support the problem description and its implemented solution Figure 5.12 is used, where viewing spheres of objects A and B had been simplified to viewing circles, and where the work space of the active sensor, which has a parallelepiped shape in this work, had been simplified to a rectangular region. In this figure, the viewpoints $\mathbf{v}_{i1}, \mathbf{v}_{i2}, ..., \mathbf{v}_{in}$, in the viewing circle of the object are illustrated as small triangles oriented through the circle center. The other kinematic constraint associated with the sensor orientation is determined by the limitation of the pan and tilt angle ranges. The small rectangles that appear in the center of the viewing circles represent bounding boxes – a simplification of convex hulls – for the object to localize. This bounding boxes will be used to conform collision and occlusion criteria.

The method to determine the viewpoints sequence to implement during the execution of the assembly plan is based on an divide-and-conquer strategy of two stages. In the first stage, and without considering the order of the assembly steps, obtains a reduced set of the best viewpoints sequences for each of the assembly steps requiring of preventive vision. In the second stage and considering now the sequence of steps of the assembly plan, from each the reduced sets of viewpoints sequences, obtained in the first stage, selects the best one to be implemented as part of the global strategy of preventive vision. Dividing the problem in two stages its complexity is reduced, since this grows

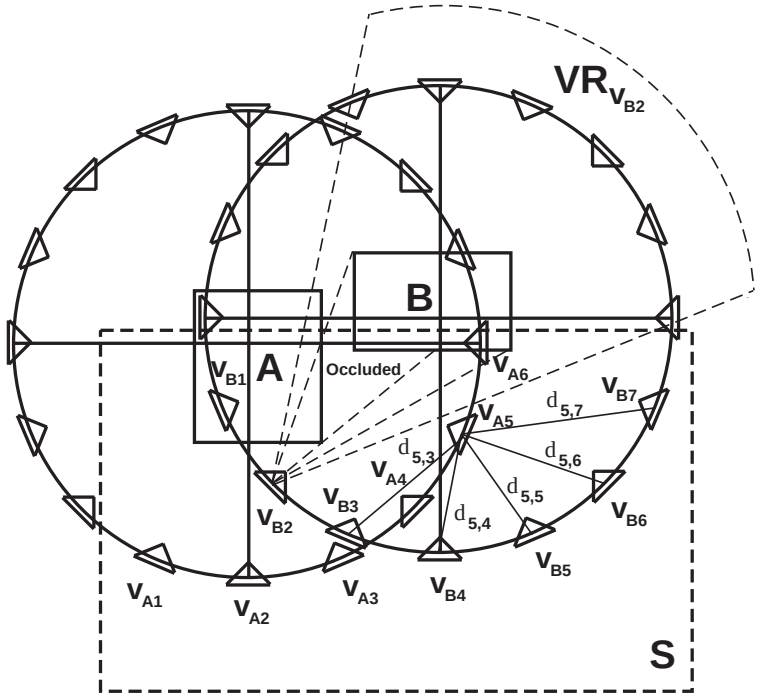


Figure 5.12: Elements for determining a viewpoints sequence to observe two objects.

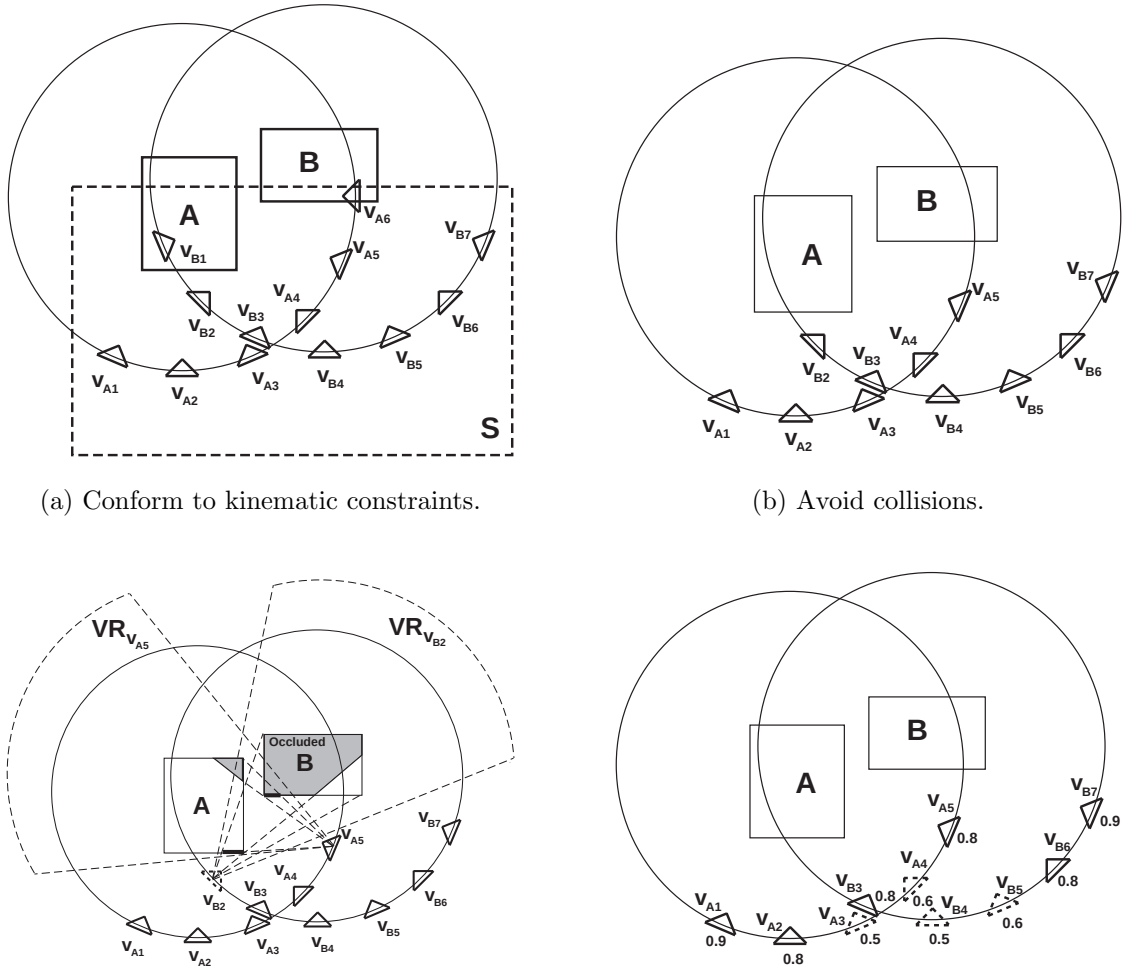
exponentially with the number of assembly steps, the number of objects to observe, and the number of reachable viewpoints by the active sensor.

5.4.1 First Stage: Selecting Viewpoints for Assembly Steps

First stage is subdivided into two sub-stages. In the first sub-stage reduce sets of viewpoints are selected, where viewpoints are realizables and have good Psp values. In the second sub-stage sequences of viewpoints are created from the sets obtained in the first sub-stage.

The first sub-stage is implemented by following four-step procedure:

1. *Conform to kinematic constraints.* For each object i to observe, eliminate all the viewpoints of its viewing sphere that can not be reached due to the kinematic constraints of the sensor mechanism, i.e. eliminate every viewpoint $\mathbf{v}_{ij}$ which falls out of the work space of the active sensor and/or which requires of rotation angles that are bigger than those allowed by the pan-tilt mechanism of the camera. Figure 5.12 depicts a situation where two objects must be observed, Object A and Object B. Each of their viewing circles contain 16 viewpoints. The viewpoints to be eliminated are those which fall outside the big rectangle; those that appear



(a) Conform to kinematic constraints.

(b) Avoid collisions.

(c) Reduce viewing occlusion.

(d) Select successful viewpoints.

Figure 5.13: Selection of sets of viewpoints for each assembly step.

without a label. For explanation purposes, it is assumed that camera has a rotation range of 180 degrees. Under this assumption, as shown in Figure 5.13(a), the labeled viewpoints are maintained.

2. *Avoid collisions.* Eliminate viewpoints that fall inside bounding boxes of objects to localize. In this way collisions between the objects and the sensor mechanism are avoided. This step eliminate viewpoints $\mathbf{v}_{A6}$ and $\mathbf{v}_{B1}$ as illustrated in Figure 5.13(b).

3. *Reduce viewing occlusion.* Eliminate viewpoints for which a computed occlusion index exceeds a predefined threshold t_o. Though the localization tool is robust to partial occlusion, its performance degrade quickly when more than 25% of the object to localize is occluded. Since the models of the objects used by the tool are

composed by edgels, the index of occlusion is computed as a percentage of edgels visible from a viewpoint that are occluded by other environmental objects in the assembly scene.

The exact index of occlusion i_o can be computed as follows:

(a) Obtain the set F_o of all the polynomial faces of the environmental objects, different from the object to localize, that could be occluding an otherwise visible edgel. These faces are described by a normal vector $\mathbf{n}_f$ with a component in an opposite direction to the viewing direction $\mathbf{d}_v$ when the camera is in the analyzed viewpoint v_{ij}, s.t. $\mathbf{n}_f \cdot \mathbf{d}_v < 0$.

(b) Review each edgel of the set of visible edgels E_v obtained from the 2DTM model of the object to localize, to determine if the straight line segments described between these and v_{ij} intersects any of the faces in set F_o. In case of intersection, increase the counter c_o of occluded edgels.

(c) Compute the occlusion as $i_o = c_o/|E_v|$.

The exact computation of the occlusion index is expensive, so a more cost effective form of computing an approximation to the index is modifying the first step of the algorithm, so that instead of using the polynomial faces of the environment objects, use the rectangular faces of the bounding faces of these objects. Doing this, there is a risk to overestimate the index, however this would have an effect similar to that of using a smaller occluding threshold value than the predefined one t_o. Figure 5.13(c) illustrate the occluded regions for viewpoints $\mathbf{v}_{A5}$ and $\mathbf{v}_{B2}$ as shadowed regions of bounding boxes of objects to observe. As can be seen, the level of occlusion of Object B observed from $\mathbf{v}_{B2}$ viewpoint is big, easily exceeding any reasonable threshold, then this viewpoint is eliminated. However, the level of occlusion in object A observed from $\mathbf{v}_{B2}$ viewpoint is very small and therefore this viewpoint is preserved.

4. *Select successful viewpoints.* From set V_{ir} of preserved viewpoints for object i, a subset V_{is} is select to contain at most n viewpoints with the highest Psp values. The n value is defined in agreement with quantity and Psp value criteria, the size of the subset must be small to prevent a combinatorial explosion and the Psp values must be high but not as to eliminate all the candidate viewpoints. To implement these criteria new threshold have to be determined. Figure 5.13(d) show Psp values for each of the preserved viewpoints, where the last viewpoint to be eliminated in the case of using a reasonable threshold of 0.7 are depicted with dashed lines, i.e. $\mathbf{v}_{A3}$, $\mathbf{v}_{A4}$, $\mathbf{v}_{B4}$, and $\mathbf{v}_{B5}$.

The second sub-stage of the first stage is implements by the following three step procedure:

1. *Create viewpoint tuples.* Create tuples of n viewpoints (n-tuples) each; the n value is determined by the number of objects to observe in an assembly step. Tuples are obtained as the Cartesian product of the reduced sets of viewpoints of the preceding sub-stage; i.e. if in an assembly step only one object will be observed, the set of tuples will contain a 1-tuple for each preserved viewpoint of viewing sphere V_A; if two objects will be observed during a step, the set of tuples will contain a 2-tuple for each pair resulting from the Cartesian product $V_A \times V_B$; if three objects will be observed during a step the set of tuples will contain the triplets resulting from the Cartesian product $V_A \times V_B \times V_C$; and so on.

2. *Configure sequences.* Each of the n-tuples obtained in the preceding step should be converted in sequences of n viewpoints, where the order of these will be that which requires of moving the sensor the smallest distance. The metric used is computed as the sum of the Euclidean distances between pairs of viewpoints, i.e. the sequence s_i obtained for the n-tuple i is defined as permutation $\pi(j)$ of $n!$ possible permutations that for $1 \leq i, j, k \leq n$,

$$\|\pi(j)\| \leq \|\pi(k)\|, \tag{5.58}$$

   where $j \neq k$ and

$$\|\pi(i)\| = \sum_{p=1}^{n-1} d_{Euclid}(v_{i,p}, v_{i,p+1}). \tag{5.59}$$

3. *Select best sequences.* Select a reduced number n_s of the best sequences obtained in the preceding step using as decision criteria a combination of the contribution of the viewpoints in the sequence to the success of the task, their indexes of occlusion, and the required motion of the sensor. All this factors where already considered in isolation to reduce the complexity of the problem, however they also have to be used together to determine the best sequences.

   The selection function C_s applied to sequence i, which contains $n_s(i)$ viewpoints, is computed as

$$C_s(i) = \alpha_1 \lambda_s(i) + \alpha_2 \delta_s(i) + \alpha_3 \gamma_s(i) \tag{5.60}$$

   values α_1, α_2, and α_3, where $0 \leq \alpha_1, \alpha_2, \alpha_3 \leq 1$ y $\alpha_1 + \alpha_2 + \alpha_3 = 1$, describe the relative importance assigned to each of the criteria included in the selection function. In this function the combination of Psp values quantified as $\lambda(i)$ has priority over the accumulated displacements of the sensor quantified as $\delta(i)$, and this has priority over the combination of levels of occlusion quantified as $\gamma(i)$, i.e. $\alpha_1 \leq \alpha_2 \leq \alpha_3$. Expressions for the computation of the criteria where normalized

to values between 0 and 1, where 0 is the worse value and 1 the best one. The best viewpoints sequences are those that obtain the highest $C_s(i)$ values.

The $\lambda_s(i)$ criterion quantifies the composed probability of having success during the execution of the assembly step if sequence i is utilized, as

$$\lambda_s(i) = \Pi_{j=1}^{n_s(i)} Psp(vp(s_i, j)) \tag{5.61}$$

where function $vp(s_i, j)$ returns the viewpoint j of sequence i.

The $\delta_s(i)$ criterion compute the required displacement for the sensor to implement sequence i in order to quantify a norm for the average non-traveled space inside the working space of the active sensor, as

$$\delta_s(i) = 1 - \frac{\|\pi(i)\|}{n_s(i) d_{sensor}} \tag{5.62}$$

where d_{sensor} is computed as the Euclidean distance between two 3d-points describing the parallelepiped box of the working volume of the active sensor.

Finally, the $\gamma_s(i)$ criterion quantifies the average visibility of the objects to localize from the viewpoints in sequence i, as

$$\gamma_s(i) = \frac{\sum_{j=1}^{n_s(i)} 1 - i_o(vp(s_i, j))}{n_s(i)}. \tag{5.63}$$

5.4.2 Second Stage: Constructing A Global Strategy

Once the best potential strategies (sequences of viewpoints) for each of the assembly steps are known, a global strategy to be implemented for the assembly plan must be constructed. For this stage the order or the assembly steps is crucial. The different alternatives of visual strategies are constructed as chains of sequences of viewpoint starting from an initial position of the active sensor.

This second stage is realized by an iterative algorithm similar to a beam local search method in a space of states. In each step of the algorithm, the sequences of viewpoints, obtained in the first stage, for the following assembly step requiring of preventive vision are processed; i.e. at the end, the depth of the search tree will be at most equal as the number of steps in the assembly plan. States in this search process describe incomplete plans. The level of the search tree denotes the number of steps that have required of preventive vision.

In a similar way to a beam local search, the algorithm keeps track of k states rather than just one, as other local search strategies like hill climbing. In each search step,

all the successors of the k states are generated. If all the steps in the assembly plan requiring preventive vision were processed, the selection of the best strategy on this tree level gives the visual strategy to implement. On the contrary, the best k successors of the full list of successors are selected, and the process is repeated for the next assembly step. Same evaluation function is used for both cases.

The function that obtains the successor states in the search creates a successor state for each sequence selected in the first stage.

The decision of the k best successors is realized based on an evaluation function that considers the same criteria used in the selection function for the best viewpoints of the preceding stage. However, in this occasion it combines the criteria values of the past stage and includes the distance required for the sensor reach the initial viewpoint of the following sequence.

The evaluation function $g_e(i)$ for state e_i is computed as

$$g_e(i) = \alpha_1 \lambda_e(i) + \alpha_2 \delta_e(i) + \alpha_3 \gamma_e(i) \tag{5.64}$$

where coefficients α_1, α_2, and α_3 are the same as those of equation 5.60.

This time, the $\lambda_e(i)$ criterion quantifies the average of expected execution success when implementing the chain of sequences associated with the successor i, which was constituted from state $i-1$ after adding sequence j. The value $\lambda_e(i)$ is computed by recursive equation

$$\lambda_e(i) = \frac{n_e(i-1)\lambda_e(i-1) + \lambda_s(j)}{n_e(i)} \tag{5.65}$$

where recursive equation $n_e(i) = n_e(i-1) + n_s(i)$ returns the number of viewpoints contained by the chain of sequences of state i, $\lambda_e(0) = 0$, and $n_e(0) = 0$.

The $\delta_e(i)$ criterion quantifies the required displacement of the sensor to implement the chain of sequences associated with state i by using the recursive equation

$$\delta_e(i) = \frac{n_e(i-1)(1-\delta_e(i-1)) + n_s(i)(1-\delta_s(i)) + d_{reach}(s_i)}{n_e(i-1) + n_s(i) + 1} \tag{5.66}$$

where $\delta_e(0) = 0$, $d_{reach}(s_i)$ is the Euclidean distance between the last viewpoint of sequence $i-1$ and the first viewpoint of sequence i, and $d_{reach}(s_1)$ is the Euclidean distance between the initial position of the active sensor and the first viewpoint of the first sequence.

Finally, the $\gamma_e(i)$ criterion quantifies an average of the visibility level for the chain of sequences of viewpoint of state i by using the recursive equation

$$\gamma_e(i) = \frac{n_e(i-1)\gamma_e(i-1) + n_s(j)\gamma_s(j)}{n_e(i)} \tag{5.67}$$

where $\gamma_e(0) = 0$.

Chapter 2

Sensing Analysis for Robotic Assembly

This chapter presents the analysis realized to construct the foundations of a method for the automatic generation of sensing strategies in robotic assembly. From the specification of a sequence of steps, in a nominal assembly plan, a set of force and vision sensing requirements are determined to monitor, control and correct the execution of that plan by a robot arm. The analyzer determine appropriate sensing strategies for each assembly task from a study of the transitions in the contact relations among the assembly elements. The geometric information, required to recognize the contacts formation process, is obtained from CAD models of the parts and the environment.

The nominal assembly plans for this dissertation are restricted to binary plans, which are also linear and sequential, that describe a totally-ordered sequence of assembly steps. These plans are entered as input to the sensing analyzer. An assembly plan describe the order in which each assembly component has to be manipulated by a robot arm and the sequence of configurations through which these parts have to pass during the mating process. An object configuration in the input plan describes its absolute pose with respect to an arbitrary coordinate system of reference, the *assembly frame*. Every assembly element is modeled as a polyhedral object.

As a result of the analysis, force sensing is prescribed during the execution of assembly steps to monitor and control the robot actions when there are assembly elements in contact, and visual sensing is prescribed for detecting and correcting deviations from the original plan, before a task is realized. The use of preventive vision is expected to prevent or at least reduce the probability of failure of current and future assembly operations.

The output of the analyzer answer the following questions:

- Which assembly steps require of force sensing?
- What type of force sensing operation is needed for such steps?
- Which assembly steps require of preventive vision?
- What kind of visual information is required?
- Which objects contain such information?

The visual sensing strategy is described in such a way that it can be used as input to a sensor planning module presented in chapter 4. In it, visual information about critical dimensions of assembly tasks is used to construct a criteria for the selection of the best sensor configurations for performing the sensing tasks.

2.1 Analysis of Contact States Formation

Not all the assembly tasks require of sensing feedback information. The use of sensors and sensing operations is needed only in those tasks where the amount of uncertainty about the actual configuration of objects with respect to some planned tolerances is big enough to put in risk their successful execution. In this dissertation, such dimensions are called the *critical dimensions of an assembly task*.

In order to start giving answers to the research questions, the following additional questions have to be answered:

- What are the dimensions of an assembly operation?
- How can its critical dimensions be identified?
- Which type of sensing is required for a critical dimension?
- What kind of feedback information is needed to verify the satisfaction of the constraints defined by a critical dimension?

2.1.1 Definitions

A *system* is a complex consisting of several elements with similar or dissimilar properties. The systems considered in an assembly context are capable of changing their configuration. Such systems possess *mobility*. To describe a system that changes, the values of certain parameters that describe the geometry or the state of the system at

any value of time must be specified. These parameters are called *coordinates*. A number of coordinates can describe the configuration of the system. Not all the possible coordinates are necessary to describe a system. The minimum number of independent coordinates needed to describe the configuration of a system completely is the number of its *degrees of freedom* (DOF). Any set of coordinates that are independent and are equal in number to the number of degrees of freedom of the system is called a set of *independent coordinates*. Any remaining coordinates, which may be determined as a function of the independent coordinates, are called *dependent coordinates*. The relations between the dependent coordinates, when expressed analytically, are called *equations of constraint*.

A *dimension*, in a geometric context, is any of the least number of independent coordinates required to specify a point in space uniquely. That is why a point on a plane, a *bi-dimensional space* or *2-D space*, can be specified by two dimensions, which describe its position with respect to the origin of a coordinate system of reference. Consequently, a point in a *tri-dimensional space*, or *3-D space*, require of three coordinates to describe its position, or *translation*, with respect to the origin of a coordinate system of reference.

In the case of rigid objects in a geometric context, the number of dimensions required has to specify its orientation in addition to its position. A *rigid object* is defined as a system of particles for which distances between particles remain unchanged. If a particle on such an object is located by a position vector fixed to the object, the vector never changes its position relative to the object, even when the body is in motion. The position and orientation of a rigid object, better known as its *pose*, completely specify the spacial distribution of these particles. The orientation of a rigid object in a 2-D space requires of one additional dimension to specify its rotation with respect to a point of reference (usually the origin), s.t. three dimensions are required. In the case of a 3-D space, the specification of an object's pose require of six dimensions, three for its position and three for its orientation.

For this dissertation, the *dimensions of an assembly operation* are defined as the least number of independent coordinates required to specify the pose of the object manipulated by the robot arm and the poses of the objects in the environment that participate in contact relations with the manipulated object. Before defining the critical dimensions of assembly operations in this sensing analysis context, the success criteria of their execution has to be formulated.

As will be explained later, an assembly operation can be characterized by a *transition* between contact states. A *contact state* is deduced from the configuration of the features in contact of mating parts. Under this context, an assembly plan execution will be considered successful if after the last assembly step, all the expected contacts among the assembly parts are attained; a condition that is also expected if every assembly step succeed. Then, an assembly step is considered as successful if the transition between

the expected contact states is realized.

This characterization of success for an assembly operation facilitates the segmentation of the nominal plan and relaxes its dependency on the absolute positioning of the parts. If the execution goal were to reach an absolute pose for an object, all the dimensions would be equally important and the criteria to select an specific sensing strategy over others should rely in the minimization of configuration error with respect to all dimensions of the task.

With the new success characterization, to produce an specified contact state, what is important is the relative position among the objects and their features in contact. Since a contact is a directional phenomenon, it is expected that some dimensions become more important than others. Then, the criteria to select an specific sensing strategy relies, mainly, in the minimization of the configuration uncertainty with respect to these dimensions.

In the previous terms and since the commanding event is the change of contact states, a nominal assembly plan can be segmented by detecting the addition or reduction of contacts. Uncertainty can potentially generate unexpected contacts during the execution of an operation. This is the fact that produces the requirement of using sensorial information and transforms the conformant problem of executing an original sequential plan into a contingency problem where a partially observable environment where actions are uncertain require of new perceptual information after performing certain actions [80]. The new plan interleaves assembly steps with sensing tasks, and if needed, with preventive or corrective actions.

Since in the case of contact relations between two objects, the pose of one object fully or partially defines the pose of the other, it becomes clear that the number of DOF of the pair is less than the total number of DOF of two free rigid objects. Therefore, a contact relation describes a constraint that reduces the number of DOF in a system. In the present work, a *critical dimension of an assembly task* is defined as a dimension of an assembly operation associated with a DOF, to be constrained by future contacts, where arbitrary small deviations from the planned configurations of participator objects can violate its equations of constraint. Such violation usually produce failure in the task execution.

2.1.2 Analyzing and Representing Contact States

In order to detect and use contact information for sensing analysis, contacts formation have to be represented and analyzed in a convenient way.

The first step in creating a tool to generate sensing strategies without depending on specific objects is the identification of a reduced group of types of contact states. Ikeuchi

et. al [49] used a taxonomy of contact states for objects in a polyhedral world subject to pure translational movement that identifies all possible assembly relations based on the directions of contact surface normals (in the two-dimensional case, the assembly relations were considered among polygons by using normal directions at the interface edge). The contact directions and possible movement directions were represented on the Gaussian sphere. This taxonomy was originally used to develop the *assembly plan from observation* (APO) method to program a robot. An APO system recognizes assembly operations performed by a human operator in front of a vision sensor. As output, it produces an assembly plan to reproduce the observed behavior using a robot manipulator.

Originally developed to represent face contact relations, this taxonomy of contact-state relations can be used to represent the translational constraint in the general case. In this, contacts are represented by a set of one or more vectors indicating the directions of contact [74]. The representation of the rotational constraint depends on the discovery of rotation references (centers of rotation in 2-D space or axes of rotation in 3-D space) under the presence of contacts. Miura and Ikeuchi [64] included one rotational DOF with respect to the axis of symmetry for polyhedral and cylindrical objects in their analysis of visual sensing planning for APO.

Contact States under Translational Constraint

The constraining effect in translational motion of contacts over a manipulated object depends on its direction and can be represented by the constraint equation

$$\mathbf{n} \cdot \triangle\mathbf{T} \geq 0 \tag{2.1}$$

where $\mathbf{n}$ denotes the contact direction, or *constraint vector*, and $\triangle\mathbf{T}$ the possible translational motion vectors. Though the equality depends on the frictional resistance to motion generated among the contacting features, it will be ignored by this analysis under the assumption of applying enough force to defeat the existent resistance.

The constraint vector and all the possible translational vectors can be represented on the Gaussian sphere (a unit sphere where vector information can be mapped onto). A single contact constrains any motion in a direction that has a component opposing to its own, then reducing the freedom of the manipulated object to a half-space defined by a bisecting plane orthogonal to its direction, or *constraint plane*.

To represent a single contact on a Gaussian sphere, all the vectors are translated so that their starting points (their tails) are located on its center and their ending points (their heads) on its surface. If all the vectors are rotated so that the constraint vector is located on the north pole of the sphere, the translational freedom will be constrained

to the northern hemisphere (see Figure 2.1). Small motions on the sphere's equator are expected to maintain the contact state and motions with a component through the north pole to break it. In the case of multiple contacts with different directions, the situation can be modeled by a system of linear equations similar to equation 2.1. The simultaneous solution of these equations can be represented in the Gaussian sphere as a polyhedral convex cone [31].

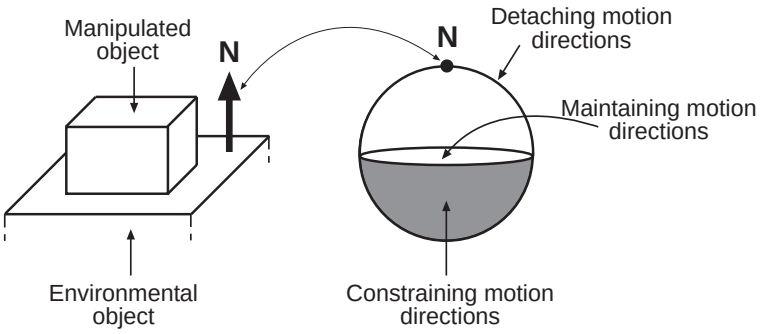


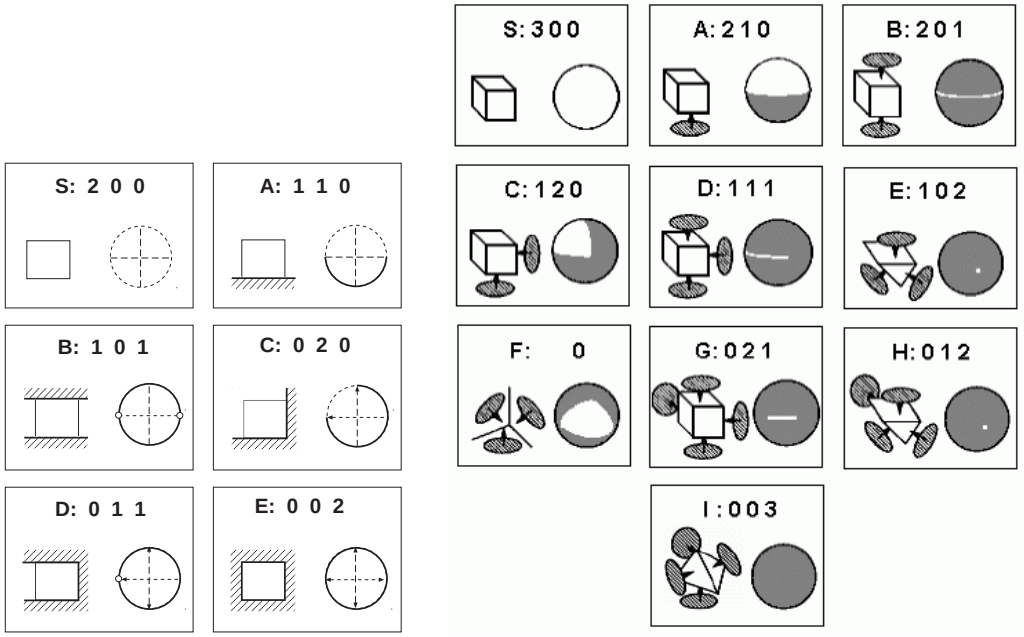
Figure 2.1: Gaussian sphere representation of a single contact constraint.

Following the above representation, the contact states of a manipulated object can be classified depending in the number of completely unconstrained, partially constrained, and completely constrained degrees of freedom (DOF). The completely unconstrained, partially constrained, and completely constrained degrees of freedom are classified as *maintaining DOF*, *detaching DOF*, and *constraining DOF*, respectively, obeying to the expected result of allowing a differential translational motion. A maintaining DOF indicates that there is no constraint component in that direction and that a very small movement is not expected to cause a new contact. A detaching DOF indicates that a constraining component exist in that direction and then a conveniently selected motion can break the contact. In a constraining DOF there is no possibility of movement.

Using the above convention, each contact state can be classified by a triplet indicating the number of maintaining, detaching, and constraining DOF. Considering only the translational DOF, there are six possible combinations of DOF for the 2-D case and ten possible combinations of DOF for the 3-D case. This combinations define the taxonomy of six contact states depicted in Figure 2.2(a) for the 2-D case and the taxonomy of ten contact states depicted in Figure 2.2(b) for the 3-D case.

Contact States under Rotational Constraint

The effect of a translation on a point is a linear displacement in a direction specified by a single vector. In the case of a translated rigid object, which in the general case can be described as a bounded set of points, all the points travel the same distance in the same direction.



(a) 2-D contact states.

(b) 3-D contact states.

Figure 2.2: Contact states taxonomies for translation.

The effect of a rotation on a point is a circular displacement with respect to a reference. In 2-D space, such reference is a point that represents a center of rotation. In 3-D space, such reference is a vector that represents a rotation axis; in this case, the motion is in a plane orthogonal to the rotation axis. Rotation can be in the clockwise or counter-clockwise direction of a right- or left-handed coordinate system.

For a point, the direction of motion and the radius of the arc defined as product of a rotation is proportional to the shortest distance to its reference. The length of the described arc depends on the radius and the angle of rotation. This means that a rotation affect differently the points that constitute an object, making them move different distances and in different directions. This is why the object is not only displaced; it also changes its orientation.

When there are contacts, the existence of a point trying to move in a constrained direction as result of a rotation is enough to declare such rotation as constrained. A rotational degree of freedom also can be completely unconstrained, partially constrained, or completely constrained; but, in this case, the classification of maintaining, detaching, or constraining DOF depends also in the selection of the rotation reference.

2.1.3 Analyzing and Representing Assembly Tasks as Procedure Graphs

Since the contact-state taxonomies defined for the translational case are complete, the contact configuration of assembly elements at any time can be classified as one of the proposed categories, and the effect of motion of a manipulated object identified with a transition between two of those categories. The transition is determined by the type, direction, and magnitude of the movement.

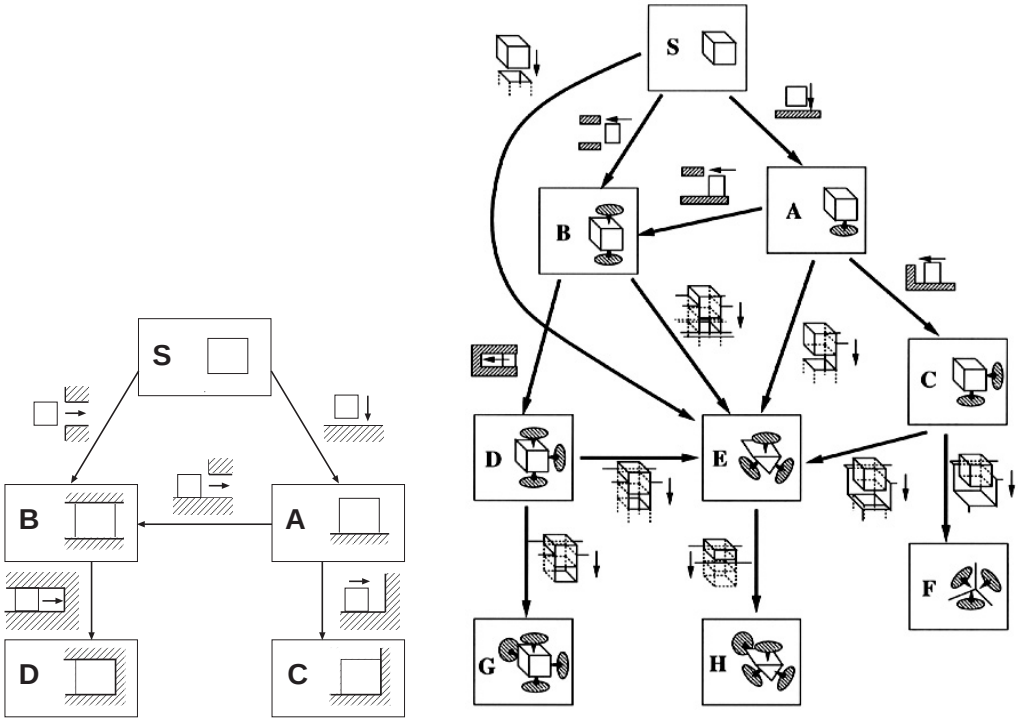
In some versions of APO systems, instead of continuously observing the human actions, an image of the scene is taken at discrete periods of time, usually before and after an assembly step. This causes that many information about the trajectory and manipulation of the objects be lost during the blind periods. To cope with non-perceived events, some assumptions about the type of transitions of contact states are done, e.g. the monotonous increment of constraints (contacts) and the inclusion of assembly steps to avoid the generation of multiple simultaneous contacts. Such assumptions and an assembly by disassembly analysis were used in [49], [48], and [47] to obtain reduced directional transition trees known as the procedure tree (Figure 2.3 show the procedure trees used in APO systems for polygonal and polyhedral objects).

The changes in the contact-state relations that characterize a task are identified by transitions of DOF in the manipulated object. There are six possible types of transitions between DOF: maintaining to detaching (M2D), maintaining to constraining (M2C), detaching to constraining (D2C), detaching to maintaining (D2M), constraining to maintaining (C2M), and constraining to detaching (C2D). Figure 2.4 depicts typical operations related with every DOF transition.

Only the first three transition types– M2D, M2C and D2C –, further referred as *restraining DOF transitions*, were used in the mentioned versions of APO, due to the assumption of the monotonous increment of constraints. In these cases, the manipulated object can pass through a series of previous contact configurations before it reaches its final pose, but once a contact is produced, it is maintained.

The last three transition types – D2M, C2M and C2D –, further referred as *releasing DOF transitions*, are indicative of broken contacts. A reduction of contacts usually means that the assembly plan included some previous assembly steps that moved a manipulated object to a temporal contact configuration conveniently selected to reduce its pose uncertainty by taking advantage of contacts with features of the environment, to modify the grasping configuration, or to coordinate the ordered assembly of multiple objects.

It is also possible that an assembly plan includes some steps that do not modify the category of any DOF of the manipulated object. These kind of steps can not be char-



(a) 2-D transitions. (b) 3-D transitions.

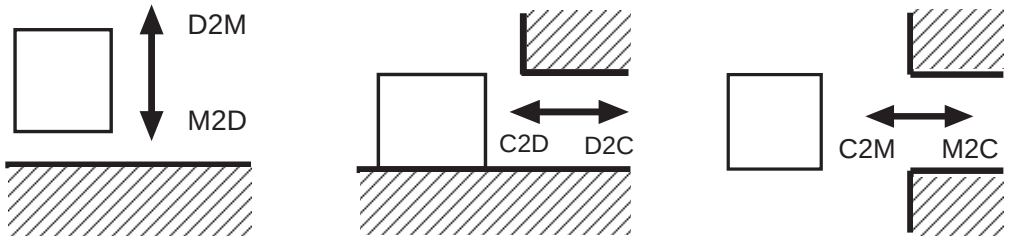
Figure 2.3: Procedure trees used in APO.

acterized by a change on the state of their DOF, but since some of them could require of some kind of sensing information, they have to be considered and included in the representation of transitions.

In general, after *non-redundant* operations that produce temporal contact configurations to modify the grasping configuration on the manipulated object or to coordinate the ordered assembly of multiple objects, the robot arm liberates (ungrasp) the manipulated object. This is when a *transfer operation* becomes a *transit operation*: the manipulated object becomes environmental and the robot hand becomes the manipulated object.

The type of nominal assembly plans expected as input for this study ensure that every part that is assembled can be disassembled. This is true because (1) the assembly parts are rigid objects, (2) only one part is manipulated at a time, and (3) the configuration of an environmental object is not changed by the action of an assembly step.

The inclusion of assembly operations that involve releasing DOF transitions or that do not produce any DOF transitions at all, transforms the procedure trees of Figure 2.3 into the procedure graphs of Figure 2.5. The 2-D procedure graph includes 5 nodes (the contact states) and 13 edges (the transitions), while the 3-D procedure graph includes



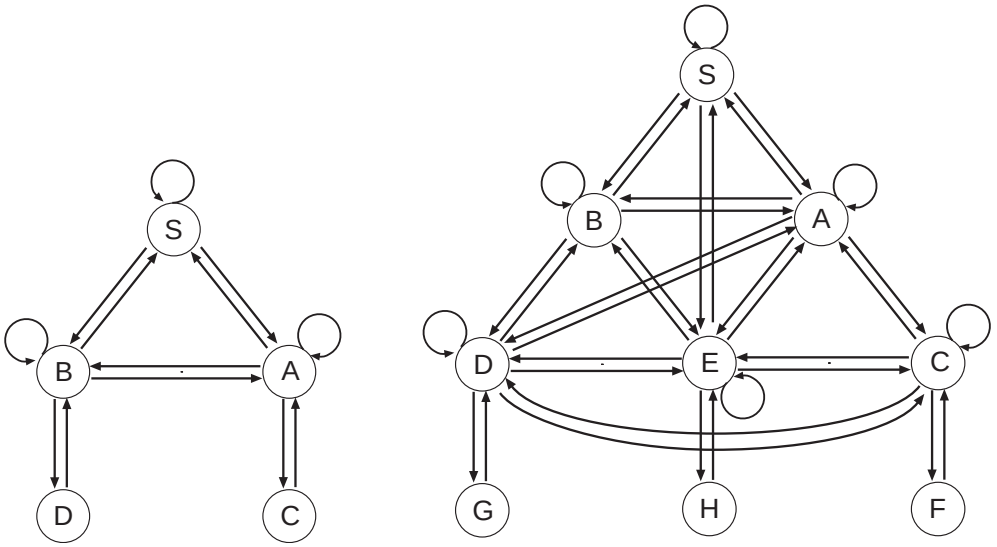
(a) M2D and D2M transitions.

(b) D2C and C2D transitions.

(c) M2C and C2M transitions.

Figure 2.4: Taxonomy of DOF transitions.

9 nodes and 36 edges.



(a) 2-D transitions.

(b) 3-D transitions.

Figure 2.5: Procedure graphs.

2.1.4 Assembly Skill Primitives

Every assembly task comprise some kind of motion. The effect of this motion on a manipulated object's DOF can be represented by the automaton shown in Figure 2.6. This automaton includes three nodes that represent the constraining categories for a DOF, six arcs that represent the six DOF transitions already described, and three additional arcs that represent a preservation of the same state. The last three arcs,

further referred as *pseudo-transitions* of DOF, are: maintaining to maintaining (M2M), detaching to detaching (D2D), and constraining to constraining (C2C).

From an analysis of the typical operations associated with the traversal of the automaton's arcs, the following four assembly skill primitives were extracted:

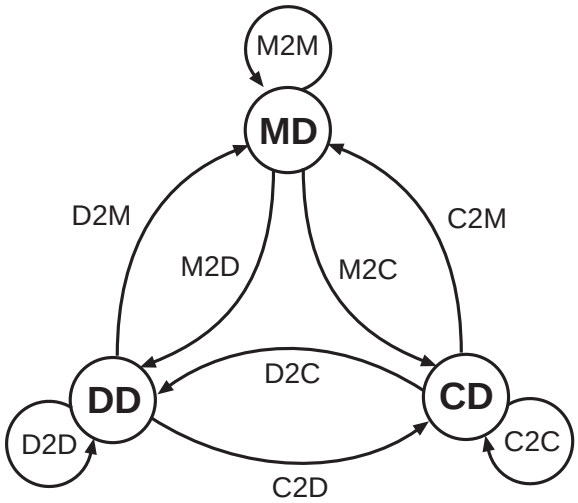


Figure 2.6: Automaton representing the effect of an assembly step on a manipulated object's DOF.

move - an assembly skill primitive to displace an object in a completely unconstrained manner. The object's motion from its current pose to a new pose follows a predefined trajectory. The move skill is required by tasks including M2M and D2M DOF transitions.

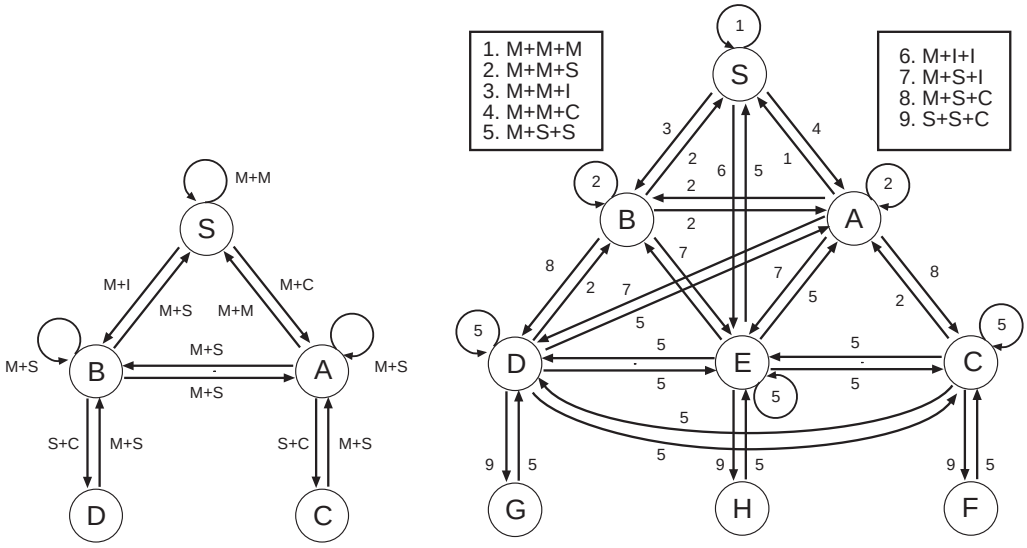
make-contact - a move assembly skill primitive that finishes when a new contact is produced between the manipulated object and the environment. The move-to-contact skill is required by tasks including M2D DOF transitions.

insert - an assembly skill primitive to move the manipulated object into a low-tolerance region where completely unconstrained DOF finish completely constrained. The insert skill is required by tasks including M2C transitions.

slide - an assembly skill primitive to move an object while maintaining the contact with at least one constraining surface (*c-surface*). The slide skill is required by tasks including the rest of the DOF transitions – D2D, D2C, C2M, C2D, and C2C.

Taking the DOF transition analysis to the procedure graphs, it is realized that the execution of some assembly operations require the concurrent use of multiple assembly skill primitives. To identify skills for a particular task it is only needed to review the

triplets used to classify the contact states before and after a task, identify every DOF transition, and add one skill requirement for each DOF transition in accordance with the automaton. Figure 2.7 depicts the required skills for the assembly operations in the procedure graphs.



(a) 2-D skills.

(b) 3-D skills.

Figure 2.7: Assembly skill primitives.

2.1.5 Identifying Contact States

In the present work, the identification of contact states is fundamental. A contact state is derived from the recognition of the features in contact of the assembly components. Such recognition is required during the sensing analysis and the execution of sensing strategies. Since in this work, only surface-to-surface contacts are considered, in both cases, the recognition of features in contact (faces) are extracted by using a transformation from body coordinate systems to face coordinate systems from known (or estimated) object configurations.

During the sensing analysis stage, the contact states are deduced from the planned configurations of CAD models of the assembly elements in the scene. For every assembly step, the new contacts and contact states produced by this operation are used to determine force and vision feedback requirements for each involved object. Furthermore, two representations are constructed: one that describe the evolution of relative dependencies among configurations of objects, and another, that describes objects' freedom of motion.

The assembly relation between the manipulated object and several stationary environmental objects is determined from contacting face pairs by analyzing the contact directions of pairs. Contacting face pairs are extracted from the face configurations. A contacting face pair is composed by a face of the object manipulated in the current task and a face from an environmental object, which have surface normals opposite to each other.

The contact directions are defined as the normal directions from the environmental faces of contact face pairs. Face contact pairs are grouped into a set of contact directional groups so that each group has face pairs with the same contact direction. By examining the occurrence of directions, the contact state relation can be inferred based on the following facts:

- Contact directions in 2-D contact states **C** and **D**, and in 3-D contact states **F**, **G**, and **H** describe a full basis, s.t. contain two independent vectors in the 2-D case and three independent vectors in the 3-D case. The difference between them can be deduced by identifying the number of constraining DOF implied by the contact directions. In the 2-D case, contact state **C** does not include any constraining DOF, while contact state **D** include one constraining DOF. In the 3-D case, contact state **F** does not include any constraining DOF, contact state **G** includes one constraining DOF, and contact state **H** includes two constraining DOF.

  The number of constraining DOF can be computed from the basis by counting the axes where there are projections of the contact directions in both directions, s.t. if $\mathbf{x}$ is a vector from the deduced basis associated with a constraining DOF, and $\mathbf{d}_i$ and $\mathbf{d}_j$ are two different contact directions that confirm this, then $(\mathbf{d}_i \cdot \mathbf{x})(\mathbf{d}_j \cdot \mathbf{x}) < 0$.

- Contact directions in 2-D contact states **A** and **B**, and in 3-D contact states **C**, **D**, and **E** describe an incomplete basis lacking one independent vector, s.t. contain one independent vector in the 2-D case and two independent vectors in the 3-D case. Again, the difference between them can be deduced by identifying the number of constraining DOF implied by the contact directions. In the 2-D case, contact state **A** does not include any constraining DOF, while contact state **B** include one constraining DOF. In the 3-D case, contact state **C** does not include any constraining DOF, contact state **D** includes one constraining DOF, and contact state **E** includes two constraining DOF.

- Finally, contact state **S** of both cases, 2-D and 3-D, describes an assembly task where the manipulated object does not participate in contacts with the environment.

- The maintaining DOF and detaching DOF can be determined from the same analysis realized to determine the number of constraining DOF. A detaching DOF

is associated with a basis vector where all projections of the contact directions are in the same direction, and a maintaining DOF is associated with the vectors that complement an incomplete basis.

During the sensing execution stage, the contact states are deduced from the observed configurations of the assembly elements in the scene. In this stage, the dependency representation constructed during the sensing analysis stage is used to adjust the observed poses of the environmental objects to conform with the expected contact patterns.

Since vision is inherently uncertain, the independently estimated pose of the objects has to be corrected to conform physically feasible assembly relations. In this dissertation, the method considered to perform the correction is the method proposed by Suehiro et al [97]. The method basically corrects the motion parameters based on face contact relations. A face contact equation is defined so that a vertex of one face is on the plane including other contacting face. Contacting faces are recognized by considering that two faces are in contact if the distance between them is smaller than a selected threshold. Motion parameters are corrected by simultaneously solving the resulting system of non-linear face contact equations. A Newton-Raphson method is used to solve the system of non-linear equations. To defeat local minimums a first aproximation is taken from observation.

2.2 Determining the Required Sensing for Robotic Assembly

In order to succeed in the execution of an assembly plan a system that utilize sensing information has to deal with problems resulting from the uncertainties in the robot control, in the sensory information, and in the positioning of the parts to be assembled. Furthermore, this situation is complicated by the need to implement a contact motion control to deal with assembly tasks where the manipulated object is in contact with objects in the environment.

The objective of using sensing feedback data can be preventive or corrective. The goal of preventive sensing is recognizing conditions that would affect the successful performance of present and future tasks. Such conditions define a set of constraints in the expected outcome of the tasks. These constraints can then be used to construct the criteria for the generation of the sensing strategy and the selection among alternatives. On the contrary, the goal of corrective sensing is identifying the new state of an assembly after an operation to decide if something went wrong and decide if additional steps are necessary to recover from detected errors or deviations from the plan.

Preventive sensing is performed before and during the tasks and tries to predict future

states from current conditions. If the predicted states do not comply with the plan, preventive actions are effected. Corrective sensing is performed after the tasks to verify if a planned state was reached. If the expected state was not reached, corrective actions are effected.

Control sensing, a kind of corrective/preventive sensing, is utilized during an assembly operation. In this case, the assembly actions are continuously monitored to control the robotic operations which systematically react to changes in the feedback data, for example, when implementing force or visual servoing.

As described in the section about transitions between contact states, there are six possible types of transitions between DOF, but, how can the knowledge of the transition that characterize an assembly operation be used to decide if it requires of any kind of sensing?. A starting point for answering this question was the recognition of the scope of the different sources of feedback data.

Today, the use of touch and force sensors is still the preferred and best way to detect contacts, since changes on the patterns and values of force and torque data is a conclusive proof of the effects of contacts among objects. This information can be used to control the execution of an assembly step. However, the local nature of force data makes it useless in the absence of contacts. In the other hand, the global nature of visual data makes it a more convenient source of information with preventive intentions. Through vision, a representation of the state of the assembly can be created and maintained. This representation can be used as a tool for predicting problematic future conditions that could be solved in a current situation. The problem with vision is that it is not good for detecting contacts, and commonly it is more expensive and less efficient than force. In conclusion, the construction of a sensing strategy for robotic assembly requires of a trade-off between the use of force and vision feedback data.

2.3 Force and Torque Sensing

Fine-motion planning deals with the determination of the trajectories to be followed by the robot when the manipulated object is in contact with the environment, or when it may be in contact due to the uncertainties being not small enough relative to the clearances between the manipulated object and the environment [61]. Fine-motion tasks are the most error-prone operations during the execution of an assembly plan.

2.3.1 Using Force Compliance

The conventional way to detect contacts is by using touch, and force and torque sensors. The use of other kinds of sensors (e.g. vision sensors) to detect contacts has a predictive nature due to their inherent uncertainty. Though uncertainty is not strange to force sensors, the detection of the forces generated by a contact is the definite indication of its presence.

But the use of force feedback data is not always the best alternative to monitor and control the execution of an assembly operation. Since humans can blindly perform many complex assemblages, it can be initially said that force data contains enough information to perform most tasks by recognizing the geometric clues in the structure of the assembly environment through the analysis of the generated force patterns. However, it should not be ignored the number and nature of human sensors together with the adaptive behavior of human actions. The number of force sensors in a robot usually is reduced and its use is also reduced to a few strategies.

The force and torque readings of a sensor depend on the specific relation between its pose, the configuration of features in contact, the effect of gravity and friction, and the intended direction of motion [109]. Its local nature and consequently its ambiguity is characterized by such relation. When the local conditions of the task are ambiguous from a sensor's reference, the best and common strategy is a systematic search of disambiguating events (identification of specific force and torque patterns). The success and efficiency of this search depends on the specific conditions of the task and the distance from the target configuration. The preferred way of dealing with the burden of a controlled wandering (systematic search) strategy using force sensors is by getting some kind of global information about the position and orientation of the contacting objects.

An strategy to execute an assembly plan under uncertainty usually assumes the use of force sensing, either passive or active, during robot operations to perform constrained motions, aborting jamming and maintaining contacts while moving. Compliance maps reaction forces to corrective motions [62].

Force compliance can be used to perform a big variety of different manipulation tasks. However, in the context of assembly plan execution considered in this dissertation, such manipulation tasks can be classified into two categories in accordance to its nature and pretended goal: *guarded motions* and *compliant motions*.

Guarded motions are those that follow a pre-programmed trajectory until new contacts are detected. These motions avoid smashing events that could move an stationary object, modifying its configuration, and inclusively damaging some assembly elements. Compliant motions are those that occur when the manipulator's pose is constrained by the task geometry. Two methods are mainly used to produce compliant motions:

passive mechanical compliance built into the manipulator, and *force control* – an active compliance implemented in the software control loop. This motions are commonly used to implement three robotic skills: (1) moving while maintaining contact of some features of a manipulated object with one or more environmental surfaces (C-surfaces); (2) coordinating the collaboration of multiple manipulators; and (3) obtaining an stable prehension of objects.

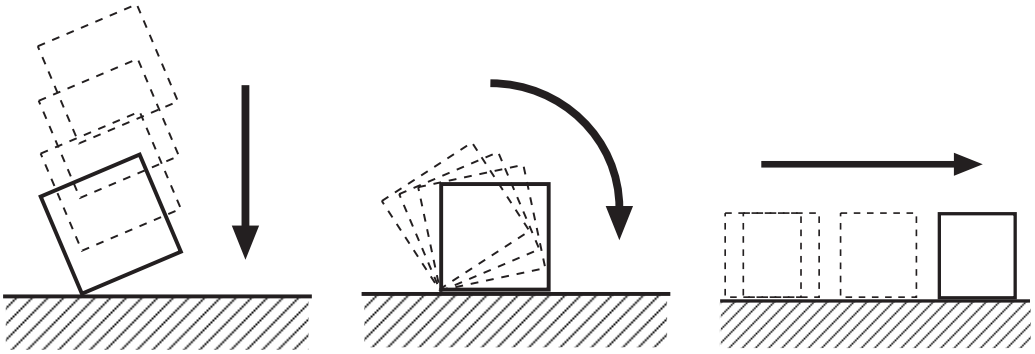
Passive force mechanisms that work based on constraints derived under the assumption of static equilibrium in the system have demonstrated their effectiveness to verify contact states in the absence of uncertainty. However, in general, this is impossible when affected by uncertainty. When the possible uncertainty can be predicted, a set of possible contact configurations can be identified. Then an active force mechanism can be used to eliminate some possibilities [19]. One way of implementing the passive force mechanism is identifying the potential contact configurations from the computation of uncertainty ranges of force and torque based on a model of bounded position and orientation uncertainty. Theoretical bounds can be computed from the CAD model of the objects for the expected topological contact configurations. Ambiguity, however, results from intersecting force and torque ranges due to uncertainty.

As mentioned before, an strategy to implement an active force mechanism to verify a contact state is identifying a set of disambiguating directions of motion. Changes in force and torque data can then be used to eliminate or retain a potential contact state. The determination of disambiguating directions is based on the recognition of constrained directions of motion in one state that are unconstrained in the rest of the possible states. Motions in such directions then are expected to reduce the sensed force for some cases and augment it for others. The unconstrained regions have to be reduced to include the amount of expected uncertainty in the pose of the objects.

2.3.2 Force Compliance Skills

To implement the assembly skill primitives required for the DOF transitions yield by an assembly step, force/torque sensors are used to detect new contacts and to react to tactile stimuli occurring during the motion. All the skills with exception of the move assembly skill primitive require of force compliance capabilities. These force compliance capabilities of the manipulator are assumed by this study. Actually, the move assembly skill primitive require of positional control, but not necessarily of any kind of additional sensing.

From an study of the literature about the synthesis and use of skill-primitives for the execution of assembly tasks and an analysis of required force skills for implementing the make-contact, slide, and insert assembly skill primitives described in this dissertation, three force compliance skills are recognized (see Figure 2.8):



(a) detect-contact skill.

(b) configuration-control skill.

(c) keep-contact skill.

Figure 2.8: Force compliance skills.

detect-contact - a force compliance skill that moves an object until a new contact is produced against the assembly environment. Information about the change on magnitude and directions of the force readings are used for this purpose. Several patterns of change can be produced and predicted from an analysis of the geometry of the contact relations, the coefficients of elasticity and friction of the contacting surfaces, and a model of the inherent passive compliance in the manipulator.

configuration-control - a force compliance skill that corrects the configuration of the features in contact through rotations of the manipulated object. Force and torque information are used for this purpose. The objective here is establishing a relative pose configuration between the contacting objects by changing the orientation of the manipulated object.

keep-contact - a force compliance skill that moves an object while maintains contact with constraining surfaces. Force and torque information is used for this purpose. The ideal case is to move in a correct direction defined by the c-surface for the task, however uncertainty can produce two problems: (1) moving in a detaching direction which would break contacts, and (2) moving in a constraining direction which would could move an environmental object or damage some assembly element. Passive or active force control has to compensate for the changes in force and torque patterns.

The make-contact assembly skill primitive requires applying a detect-contact force compliance skill followed by a configuration-control skill. The detect-contact skill avoids smashing events while monitoring an object motion, but due to uncertainty it is very common that a perfect face-to-face contact is not produced. Usually, some face features

(vertices or edges) of the manipulated object arrives first. The configuration-control skill is used to correct this situation (see Figure 2.9).

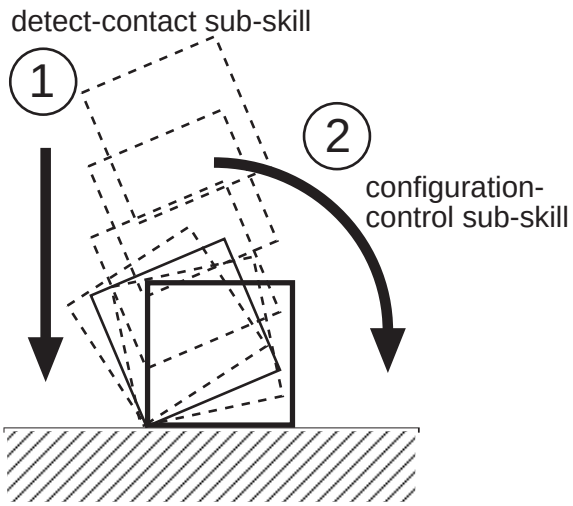


Figure 2.9: Implementing a make-contact assembly skill primitive through force compliance skills

The slide assembly skill primitive only requires of the keep-contact force compliance skill.

The insert assembly skill primitive is required by the most common automated mechanical assembly task and is the most difficult primitive to implement. Though, there have been different proposals to perform it using only optical sensors [75] [64], it usually requires of using the three force compliance skills.

An insertion task is frequently performed into low-tolerance regions where small errors caused by uncertainty produce multiple contact configurations. One common way to manage these situations is through planning systematic strategies to monotonically approach a target configuration [22]. These strategies require of programming a set of reactions to pre-conceived patterns of force and torque data readings that include the detection of expected contacts (detect-contact skill), the adjustment of the manipulated object's configuration (configuration-control skill), and the motion in contact-space (keep-contact skill) (see Figure 2.10).

Table 2.1 summarizes the results of the above force compliance analysis. It presents the force compliance skills required to implement every assembly skill primitive. Using this table is easy to systematically associate force compliance skills to assembly operations; it is only necessary to determine a contact-state transition to know the assembly skill primitives involved through Graph 2.7.

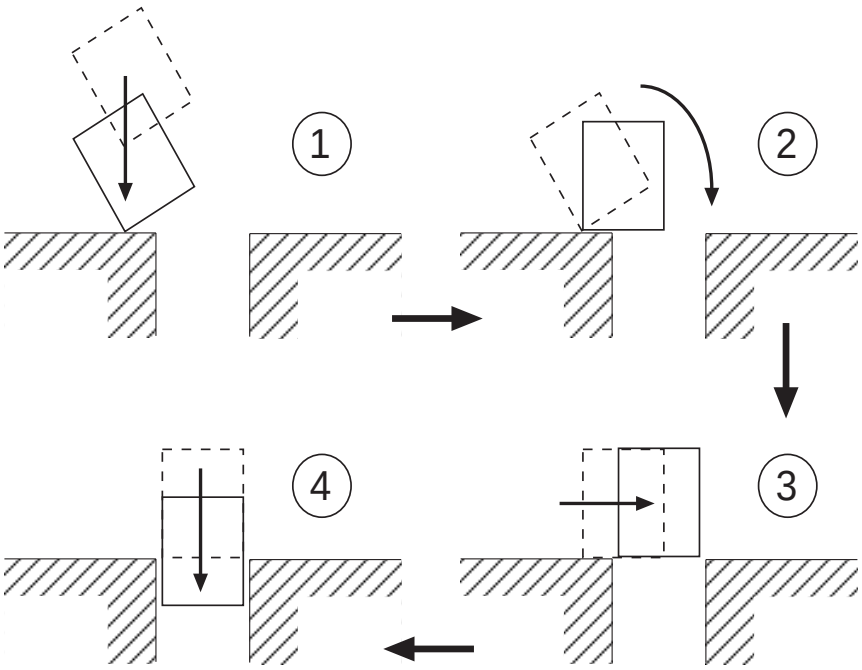


Figure 2.10: Implementing insert assembly skill primitives through force compliance skills

2.4 Visual Sensing

Vision sensors are a source of global feedback information. Through them the structure and properties of a possibly dynamic three-dimensional world are described through the process of one or multiple two-dimensional images. The images (colored or monochromatic) can be captured from one or multiple stationary or mobile cameras [27]. The sensing process can be controlled (active) or non-controlled (passive).

2.4.1 Determining Assembly Relations by Vision

The problem of determining the geometrical relations among objects in contact is inherently related with recognizing and locating objects in the scene. Since object recognition is a fundamental problem in computer vision, a lot of work has been done in its behalf. It has been recognized that the quality of the resulting vision programs is governed by the organization of the sensed features, sensors, object representations, and image acquisition strategies [46]. Since in the present work, it has been assumed that the architecture of the vision system has to be prepared in accordance with its goal and working environment, this research belongs to the task-oriented school [45]. And since knowledge of the objects is actively used for guiding the vision tasks during the recognition process, it is classified as model-based vision.

Table 2.1: Force compliance skills required by each assembly skill primitive.

Assembly skill primitive	*Force Compliance skill*
move	
make-contact	detect-contact configuration-control
slide	keep-contact
insert	detect-contact keep-contact configuration-control

The typical way to detect the contact relations among objects by vision is through a process of geometric reasoning and the use of threshold values. Since the recognition results obtained by the vision system include certain amount of error, the topological contact relations has to be extracted through examining the equations of candidate features in contact (e.g. faces). The visual uncertainty will usually generate gaps or intersections among the contacting elements [99]. For systems that work with rigid objects and assuming a reduced uncertainty, these errors are expected to be small. A convenient selection of threshold values can be utilized to recover correct contact relations.

When the expected vision errors are big so that more than one contact state is consistent with the observed scene, ambiguity is present. Visual ambiguity can be solved by using an additional source of feedback information (e.g. force data). Then a reciprocal relation has to be found among possible heterogeneous sources of sensing feedback data to identify the contact relations during plan execution. For this reason, the conclusion of fine-motion operations usually require using force sensors.

Under the assumption of working with a robotic system with the force compliance skills described in the preceding section, vision is not strictly necessary to develop the assembly skill primitives required to execute an assembly plan. However, it could be convenient and then recommendable to use vision before carrying out some of them.

Vision could also be used to assist force compliance skills. In this respect, vision could be used to reduce the positional uncertainty of the objects serving as a disambiguating criteria in force data. In the case of the detect-contact skill, vision could be helpful to discriminate among different contacting surfaces of similar geometry, e.g. to determine which side of a hole has been reached in a peg-in-hole kind of assembly operation.

In the case of the configuration-control skill, vision could be helpful to determine the grasping configuration and the contacting feature of the manipulated object.

In the case of the keep-contact skill, vision could be helpful to discriminate about causes for the changes on the force and torque patterns, e.g. to determine if a reduction of force means that the manipulated object detached from the constraining surface, which is an error, or if the manipulated object finish traversing the constraining surface, which could be the objective of extracting a peg from a hole.

Vision could be also useful to assist force compliance when several force skills are used concurrently, e.g. when make-contact and slide assembly skill primitives are performed simultaneously, vision would be useful to determine if an increment on a detected force is caused by reaching the target surface of the make-contact skill or by trying to move against a constraining direction imposed by the slide skill.

The use of vision as an assistant to the force compliance skills is an example of visual sensing with corrective intentions. This dissertation is more interested in using vision with preventive intentions. As it was established in Chapter 1, the ultimate goal of this research is using vision to increase the probabilities of success for the execution of an assembly plan without modifying the main sequence of assembly operations commanded in the plan. This means that the sensing analyzer has to take advantage of the periods when the manipulator has control over the assembly parts, to perform some convenient adjustments in their poses.

The analysis to identify the assembly skill primitives that require of preventive vision is divided in two parts: one that analyzes an assembly operation isolated from the rest of the plan, and another, that analyzes the roll of an assembly operation as part of the full plan. In the first case, visual sensing requirements are determined considering only the new contact relations produced by the task when an object is manipulated. In the second case, visual sensing requirements are determined considering the contact relations produced by tasks when an object participates as part of the environment. Next section describes the visual strategy followed for the first case, while the next chapter deals with the second case.

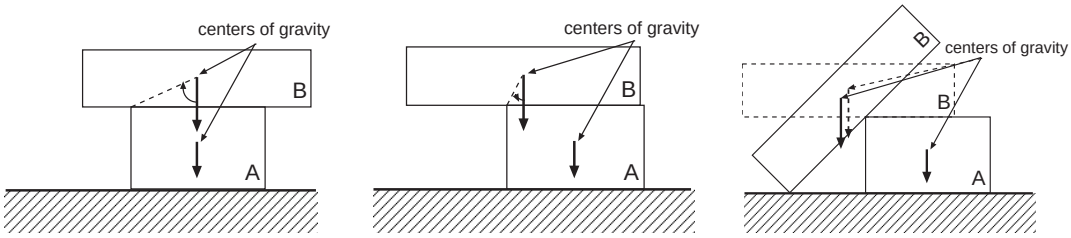
2.4.2 Preventive Vision for a Manipulated Object

Every assembly skill primitive needs of some kind of positional control and machine vision is a global source for this. However, how accurate should this control be?. The answer to this question is that it depends on the task at hand. Every assembly operation supports certain amount of error. To always succeed, the accuracy of the positional control strategy should be bigger enough to allow smaller errors than those tolerated by the task.

A move assembly skill primitive transfers a manipulated object between two prescribed poses. The transferred object is expected to finish in a completely unconstrained state

with respect to the DOF that originated the skill requirement. In this case, relatively small errors yield by positional uncertainty are not expected to produce failure, and then, the cost of using preventive vision is not considered worth its marginal benefit.

A make-contact assembly skill primitive transfers a manipulated object until it enters in a new contact relation and then corrects its configuration to ensure correct face contact relations. As mentioned in the previous section, in this case, vision could be used as an additional tool to discriminate among potential contacting surfaces of similar geometry. Vision could also be useful in two additional situations: for operations including small contacting regions, and for operations that include stability concerns on a partial assembly.



(a) Strong stability configuration.

(b) Weak stability configuration.

(c) Instability effect.

Figure 2.11: Task stability due to the partial assembly configuration.

The two first situations are related because the problem is that a small positional error would make the operation to fail in reaching the planned contact region and would finish in a different one. The problem with the third situation is that the center of gravity of a partial assembly translates when new contacts appear and for partial assemblies with weak stability conditions an error in the contacting position could make that some elements fall apart as exemplified in Figure 2.11.

However, these three situations are not further considered in this study because, commonly, they are indicative of bad assembly planning decisions, or of unavoidable singular assembly tasks requiring of visual and force servoing not exactly preventive vision as proposed in this work. Then, preventive vision is not required to monitor the execution of this assembly skill primitive, either.

A slide assembly skill primitive transfers a manipulated object while maintaining the contact with at least one constraining surface. Vision is not good to detect contacts, and then, it is not a good idea to use it for keeping contacts while moving. Stability can be a concern when using this skill, too, because a sliding action continuously change the position of the center of gravity of the partial assembly, but the use of preventive vision is discarded for the same reason mentioned above: it is expected that the assembly

planner produce a sequence of stable operations.

An insert assembly skill primitive is where the assembly plan executioner can really take advantage of vision as a source of preventive feedback information. Insertion is a low-tolerance operation where small errors typically cause failure. Insertion fails when the inserting features enter in contact with adjacent faces to the inserting slot, instead of penetrating the target region. This failure can be realized by monitoring the insertion process until a first contact is detected. Force and torque data can be used to determine success or failure.

Critical Dimensions for Translation

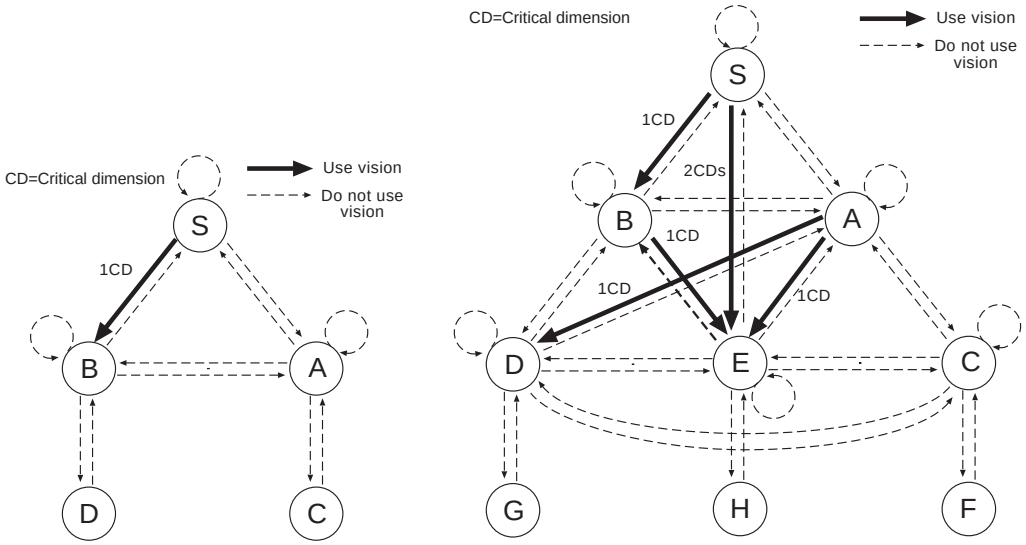
To succeed in an insertion operation, the inserting features has to conform with some alignment constraints. Since an insert assembly skill primitive is required by operations including maintaining to constraining DOF transitions, the absence of contacts before the assembly operations, with respect to these DOF, makes force feedback data useless to verify the fulfillment of such constraints. The job of vision will be then, to observe the participant objects in the insertion and determine re-alignment adjustments on the manipulated object to reduce the probabilities of error.

Figure 2.12 shows the contact-state transitions that require of visual sensing as highlighted arcs on the procedure graphs for the 2-D and 3-D cases. These are the transitions that include the insert assembly skill primitive. As it can be observed, visual sensing is recommended for a reduced group of assembly operations (only one for the 2-D case).

To successfully execute an insertion, the configuration of the features to be inserted and the configuration of the insertion slots have to correspond and include certain amount of clearance. Bigger clearances allow bigger pose errors during an insertion. The type and magnitude of error tolerance of an insertion depends on the geometry of the features involved, which means that positional errors with respect to some DOF should be more important than others. Figures 2.13(a) and 2.13(b) depict such situations.

In the peg-in-hole example of Figure 2.13(a), an error in the position of the peg with respect to the X axis of the coordinate frame of reference, is more important than errors with respect to the Y and Z axes. Very small errors with respect to the X axis could easily cause that the assembly operation fails by hitting neighboring faces to the hole, but reasonable errors with respect to the Y and Z axes can be supported and still succeed in the insertion. The X DOF of the peg represents what is further referred as a *critical dimension for visual verification* of an assembly task. Figure 2.13(b) presents a peg-in-hole operation where the critical dimensions are represented by the X and Y DOF of the peg.

In conclusion: *every M2C DOF transition in an assembly operation requires of using*



(a) 2-D procedure graph. (b) 3-D procedure graph.

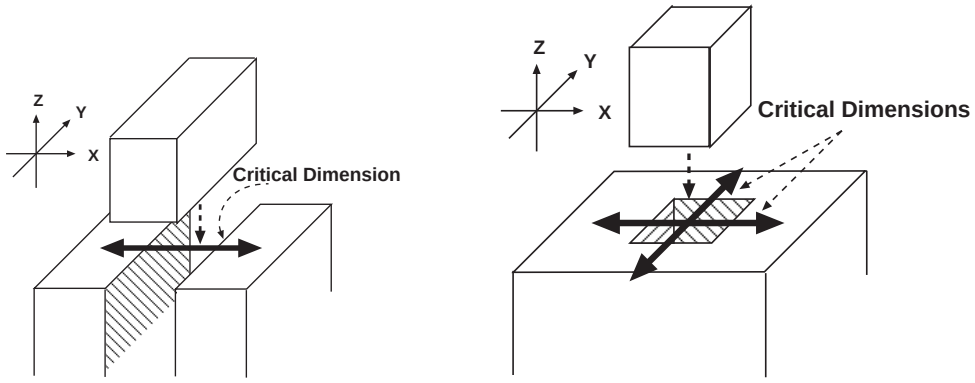
Figure 2.12: Tasks requiring visual sensing.

visual verification and includes at least a new critical dimension. The critical dimensions will be used to compose a decision criteria for the sensor planning module of Chapter 4.

Critical Dimensions for Rotation

There is at least one additional critical dimension for both cases, the rotational DOF with respect to the Z axis of the peg. An error in the attitude of the peg with respect to this axis is equivalent to translating some of the peg vertices in the critical directions. Then, the first example would include two critical dimensions – the X translation DOF and the Z rotation DOF – and the second example would include three critical dimensions – the X and Y translation DOF and the Z rotation DOF. Actually, being more rigorous in the rotational analysis, the example of Figure 2.13(a) should include another critical dimension for rotation with respect to the Y axis; and similarly, the example of Figure 2.13(b) should include two additional critical dimensions for rotation with respect to axes X and Y.

In this work, the followed approach takes advantage of the constraining effect of contacts for the definition of the rotation references, which are fixed as follows: for assembly in 2-D space, the considered center of rotation for the polygon representing the manipulated object is situated in the origin of its coordinate frame; and for assembly in 3-D space, the possible rotation axes associated with critical dimensions, further referred as *critical rotation axes*, for the polyhedral representing the manipulated object, are considered to



(a) One critical dimension. (b) Two critical dimensions.

Figure 2.13: Critical dimensions for the insert assembly skill primitive.

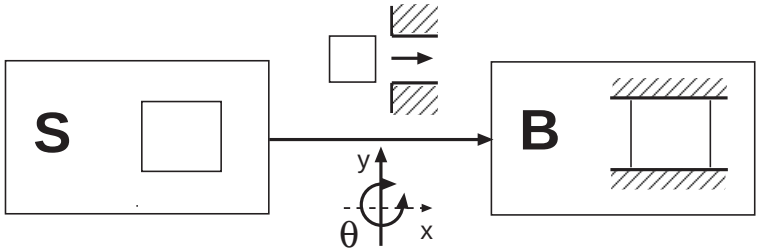


Figure 2.14: Transition of assembly relations that include critical dimensions for rotation in 2-D space

pass also through the origin of its coordinate frame.

The critical dimensions for rotation are also found in the tasks associated with the same reduced set of transitions of assembly relations that describe the critical dimensions for translation. Figure 2.14 depicts the only transition for assembly in 2-D space that require to verify visually the orientation of the manipulated object, and Figure 2.15 illustrate the five transitions for assembly in 3-D space that require to verify visually the orientation of the manipulated object with respect to some critical rotation axes.

Figure 2.15 depicts the critical dimensions as elements of coordinate frames drawn as solid lines. The axes of the coordinate frames represent critical dimensions for translation, and the arrowed ellipses around the axes represent critical dimensions for rotation. the insertion direction in each transition is aligned with the X axis. As can be noted in this figure, the most constraining tasks in a single assembly step are those that describe a transition from a contact state of type A to a contact state of type E, where the only unconstrained dimension is the translation in the direction of insertion.

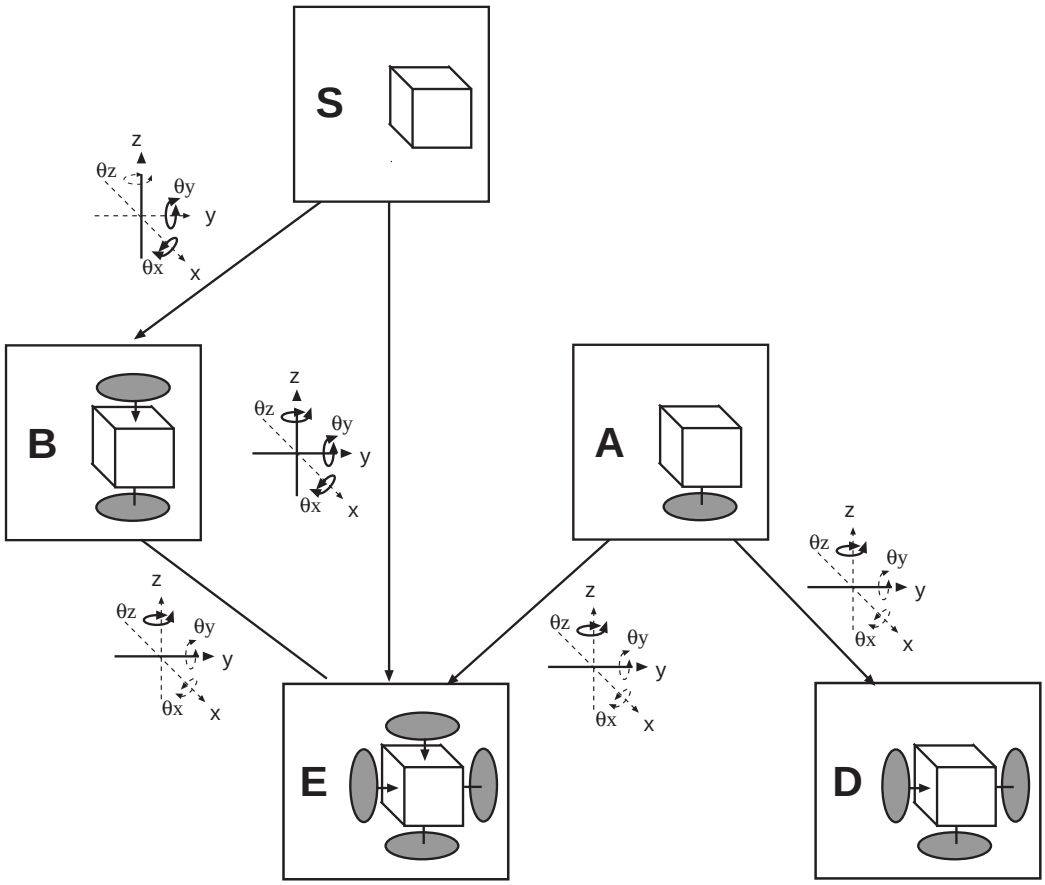


Figure 2.15: Transitions of assembly relations that include critical dimensions for rotation in 3-D space

Chapter 3

Preventive Vision for Robotic Assembly

This chapter completes the method introduced in the previous chapter for identifying the assembly tasks that require of preventive vision. A geometric reasoning mechanism is developed to discover additional preventive vision requirements. These new requirements apply to some tasks in which their manipulated objects participate as part of the environment of other tasks. These environmental objects conform, directly or indirectly, some critical configurations required for the successful execution of future failure-prone tasks.

It also introduces a graph representation to express direct and indirect dependencies, among assembly elements, caused by insertions and contacts. This graph is used to support the method that constructs a preventive vision strategy for the full assembly plan. Next, this chapter details the strategy used to construct the graph and determine the visual feedback information required by each assembly step.

The approach and method has been implemented as a computational system. This chapter finishes by presenting several examples solved by the system. Every example describe the nominal assembly plan analyzed, the graphical illustration of the assembly steps (as obtained from the system), an illustration of its final ICdg, and the modified assembly plan including force and vision sensing requirements.

3.1 Preventive Vision for Environmental Objects

An assembly task is executed in an environment composed by stationary objects known as *environmental objects*. Some of them are fixed since the very first beginning of the assembly process. These objects are commonly limited to a planar table, but it could include some other objects, like hard automation devices which could be used

to reduce some sources of uncertainty. The assembly environment also could include parts assembled previously that form part of the partial assembly. Since this research is not interested in planning additional assembly operations, the possible adjustments, resulting from configuration errors and inconsistencies observed by the vision system with respect to the assembly plan, are restricted only to modify the configuration of manipulated objects. This includes the pre-assembled environmental objects during their manipulation by the robotic assembler.

The sensing analyzer considers the definition of new visual requirements for environmental objects in order to achieve the following two goals:

1. Succeed in insertion operations, because the success depends not only in the position control of the manipulated object, it is also important that the environmental conditions for the insertion exist.

2. Succeed in producing all the expected contacts among the assembly elements during the execution of every operation.

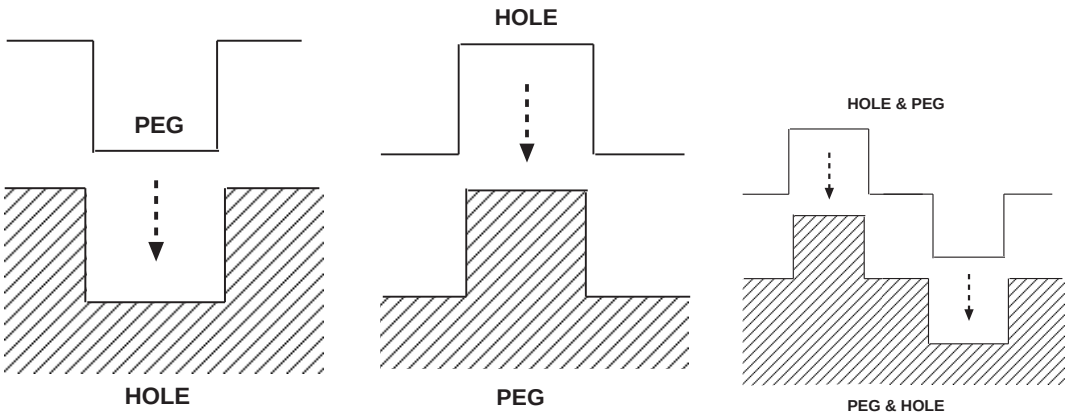
The analysis of preventive vision for insertion tasks assumes that every expected contact actually exist, then, the accomplishment of the first goal depends on the achievement of the second one.

3.1.1 Visual Sensing for Insertion Condition

In an insertion operation, the assembly environment contains inserting features as in Figure 3.1(a), or insertion slots as in Figure 3.1(b), or inclusively in some cases it contains both of them, inserting features and insertion slots (further referred as *insert features*) as in Figure 3.1(c). The manipulated object complements the insertion geometry with its own inserting features and insertion slots.

To succeed in an insertion task, the insert features of the environment have to correspond and be aligned with the insert features of the manipulated object. The configuration of the inserting and insertion features is further referred as the *insert configuration*.

When the insert configuration includes features of only one environmental object, there is not need to add new visual sensing requirements for it; only the manipulated object configuration would have to be adjusted to compensate for any observed difference with respect to the environmental object's pose. However, when the insert configuration is formed by two or more environmental objects, an *indirect insert dependency* is defined among them. If any of these environmental objects is a pre-assembled part, then, it is possible that new visual sensing requirements have to be included into its preventive



(a) Peg-in-hole insert configuration.

(b) Hole-in-peg insert configuration.

(c) Mixed insert configuration

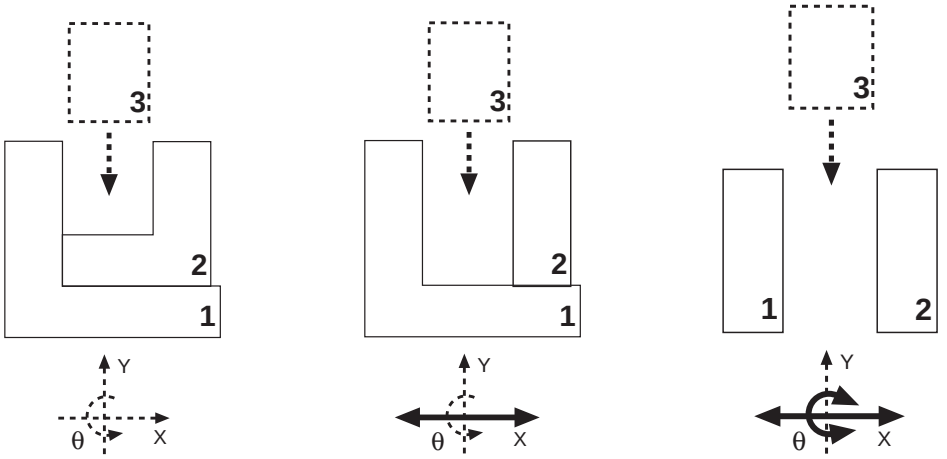
Figure 3.1: Configurations for insertion.

visual sensing strategy. These sensing requirements would have to be fulfilled when this object is manipulated.

An indirect insert dependency describes some alignment constraints among the objects that participate in an insert configuration. In some cases, the contact relations among these objects enforce the fulfillment of all the alignment constraints, and then, there is not need to add new visual sensing requirements for them. In some other cases, the environmental objects are not touching each other, and then, the only way to check if the alignment constraints have been fulfilled is by visually verifying every constraint. Still, there are cases where some of the environmental objects are in contact but the contact relations are not enough to enforce the fulfillment of all the alignment constraints described by the indirect insert dependency. In this last case, some of the alignment constraints are enforced by the contacts and others have to be verified visually.

Figure 3.2 presents three cases of a peg-in-hole operation where two environmental objects form the insertion slot. Although, every operation includes two critical dimensions that describe the alignment constraints for the peg (one for translation and one for rotation), the number of critical dimensions for the environmental objects vary. The numbers of the objects describe the assembly order.

Figure 3.2(a) illustrates a situation where the contacts among the environmental objects ensure a correct insert configuration, s.t. vision is not needed to verify the correct alignment constraints on the environment. Figure 3.2(b) illustrates a situation where the contact among the environmental objects eliminates one of the alignment constraints (the rotational one) for the insertion; to verify the fulfillment of the second one, vision is required. Figure 3.2(c) illustrates a situation where the complete absence of contacts



(a) Vision not required.

(b) Vision required to observe one critical dimension.

(c) Vision required to observe two critical dimensions.

Figure 3.2: Vision to verify indirect insert dependencies.

among the environmental objects requires of using vision to verify both of the alignment constraints imposed by the insertion, in this case, the X position and orientation of object 2 with respect to object 1.

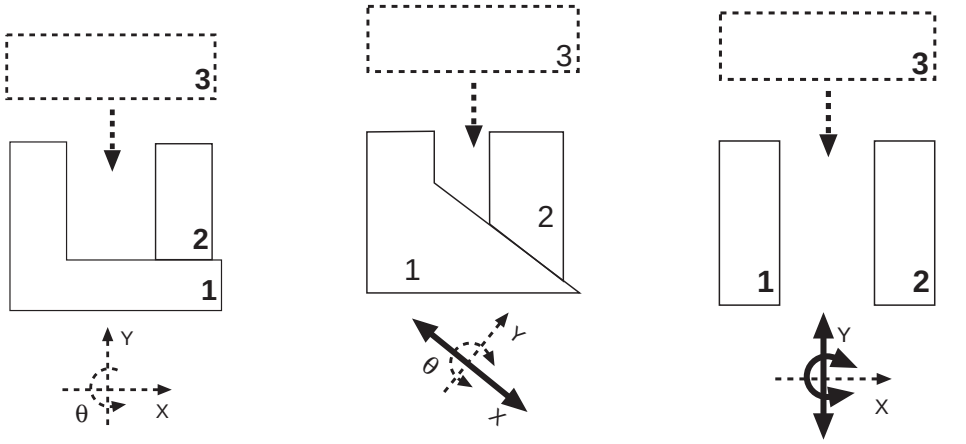
3.1.2 Visual Sensing for Contact Prescription

In this dissertation, the execution of an assembly operation is considered successful if all the expected contacts are achieved. This means that even in the case of assembly operations that do not include insert assembly skill primitives, some of these could fail because not necessarily all the expected contacts would be achievable. To achieve all the expected contacts, the configuration of the contacting features on the environment has to correspond with the configuration of the contacting features on the manipulated object. The configuration of the contacting features is further referred as the *contact configuration*.

When a contact configuration includes features of only one environmental object, there is not need to add new visual sensing requirements for it; only the manipulated object configuration would have to be adjusted, using force sensing, to compensate for any sensed difference in the force and torque patterns. Instead, when the contact configuration is formed by two or more environmental objects, an *indirect contact dependency* is defined among them. If any of these environmental objects is a pre-assembled part, then, it is possible that new visual sensing requirements have to be included into its

preventive visual sensing strategy. These sensing requirements have to be fulfilled when this object is manipulated.

An indirect contact dependency describes some alignment constraints among the objects that participate in a contact configuration. In some cases, the contact relations among these objects enforce the fulfillment of all the alignment constraints, and then, there is not need to add new visual sensing requirements for them. In some other cases, the environmental objects are not touching each other, and then, the only way to check if the alignment constraints has been fulfilled is by verifying visually every constraint. Still, there are cases where some of the environmental objects are touching each other but the contact relations are not enough to enforce the fulfillment of all the alignment constraints described by the indirect contact dependency. In this last case, some of the alignment constraints are enforced by the contacts but others have to be verified using vision.



(a) Vision not required.

(b) Vision required to observe two critical dimensions.

(c) Vision required to observe two critical dimensions.

Figure 3.3: Vision to verify indirect contact dependencies.

Figure 3.3 presents three cases where the make-contact assembly skill primitive is used to form a two objects contact configuration. Although, every operation includes two critical dimensions that describe the alignment constraints for the manipulated object (one for translation and one for rotation), the number of critical dimension for the environmental objects vary.

Figure 3.3(a) illustrates a situation where the contacts among the environmental objects ensure a correct contact configuration, s.t. vision is not needed to verify the alignment constraints on the environment. Figure 3.3(b) illustrates a situation where the contact among the environmental objects eliminates one of the alignment constraints (the rota-

tional one) for the make-contact skill; to verify the fulfillment of the other one, vision is required. Figure 3.3(c) illustrates a situation where the complete absence of contacts among the environmental objects requires of using vision to verify both of the alignment constraints imposed by the make-contact skill, in this case, the Y position and orientation of object 2 with respect to object 1.

3.2 Insert and Contact Dependency Graph

In the above section was learn that to support the successful execution of certain insertion operations and move-to-contact operations, vision should be used to verify that conditions in the environment are adequate. Such conditions were described as configuration dependencies among environmental objects forming certain insert configurations and contact configurations. To ensure the right conformation of such configurations, the dependency analysis has to be propagated backwards to the objects that conform the environment of those environmental objects, and so on. To store and structure all the discovered dependencies an *insert and contact dependency graph* (*ICdg*) is introduced.

An ICdg is a labeled, direct graph where nodes represent objects and arcs represents alignment constraints. The alignment constraints are extracted from an analysis of the insert relations and contact relations resulting from the operations described into a nominal assembly plan. The direction of the arcs answer to the assembly order, and then, describes a dependency of the configuration of one object (the arc's source) on the configuration of another (the arc's target).

The ICdg is constructed and used, during the sensing analysis stage, as a tool for determining the critical dimensions for the manipulation of objects. A manipulation operation without critical dimensions do not need of its visual verification.

The ICdg is also used, during the sensing execution stage, for adjusting the observed configurations of the objects to conform with all the alignment constraints defined by past, current, and future relations originated by insertions and contacts among the assembly elements.

3.2.1 Elements of an Insert And Contact Dependency Graph

An ICdg could include two types of nodes and three types of arcs. One type of nodes is added for each environmental object that is not moved during the full assembly process. A second type of node is added for each manipulated object. The knowledge of the type of a node will be useful during the sensing analysis and execution of the sensing tasks. The object related to nodes of the first type do not need to be observed because their

poses are usually known since the starting of the assembly, e.g. a work table. They can act like fixed constraining references for the other objects.

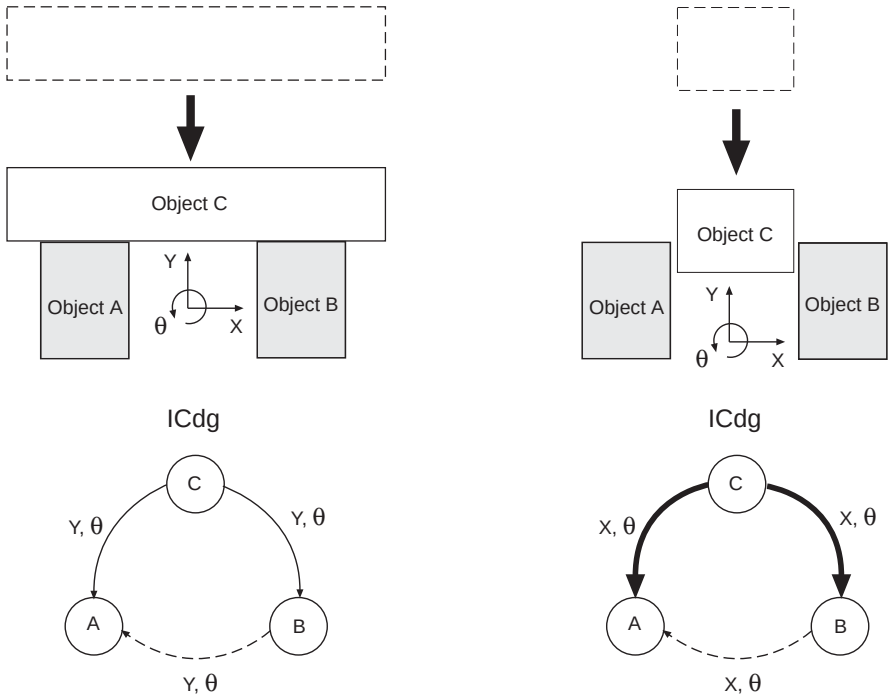
When an assembly operation requires of using vision, the pose of some environmental objects related to the nodes of the second type would need to be observed. Since the observed poses will include certain amount of error, a process to adjust the pose parameters has to be performed. The adjustments must be realized before executing the manipulation task, in order to conform to the expected direct and indirect assembly relations.

An ICdg include three types of arcs devised to record the nature of the dependency between the objects they link. Two types of arcs record alignment constraints that describe direct relations originated from contacts and insertions produced during current and past manipulations of the objects involved. These arcs are generated between the manipulated object and one or more environmental objects. The third arc type records alignment constraints that describe both indirect insert dependencies and indirect contact dependencies among environmental objects that directly or indirectly participate and then affect the success of current and future manipulations. Figure 3.4 depicts typical situations, in a 2-D space, that generate each of the arc types.

The labels of the arcs indicate the constrained DOF of the source node that can be described as a function of the DOF of the target node. Between two nodes not linked by any arc, do not holds any alignment dependency, which means that errors in the definition of the pose of one of the objects is not expected to affect the mating operation of the other.

Figure 3.4(a) depicts a subassembly produced by a make-contact assembly skill primitive. The subassembly includes a relation among three objects. The relation is between a manipulated object (Object C) and two environmental objects (objects A and B). Assuming a common coordinate reference frame for the three objects, conveniently aligned with the contact direction, the solid arcs, between Object C and objects A and B, represent alignment constraints due to their contacting faces. The attributes of the arc, Y and θ, describe the DOF constrained by the contact relation. This means that if the pose of Object A or Object B is known, the values for the Y positional parameter and the θ rotational parameter of Object C can be deduced. In this case, the face contact relation is not enough to determine a value for the X positional parameter, which stays as the only unconstrained DOF.

Furthermore, objects A and B were not related before the current operation. For succeeding in getting both face contacts is not enough to control the pose of Object C. Additional alignment requirements, for this operation, are that Object B must have the same orientation than Object A (same value for θ parameter) and that their edges that would contact Object C must be collinear, restricting the Y position of Object B



(a) Arcs produced by a make-contact assembly skill primitive.

(b) Arcs produced by an insertion assembly skill primitive.

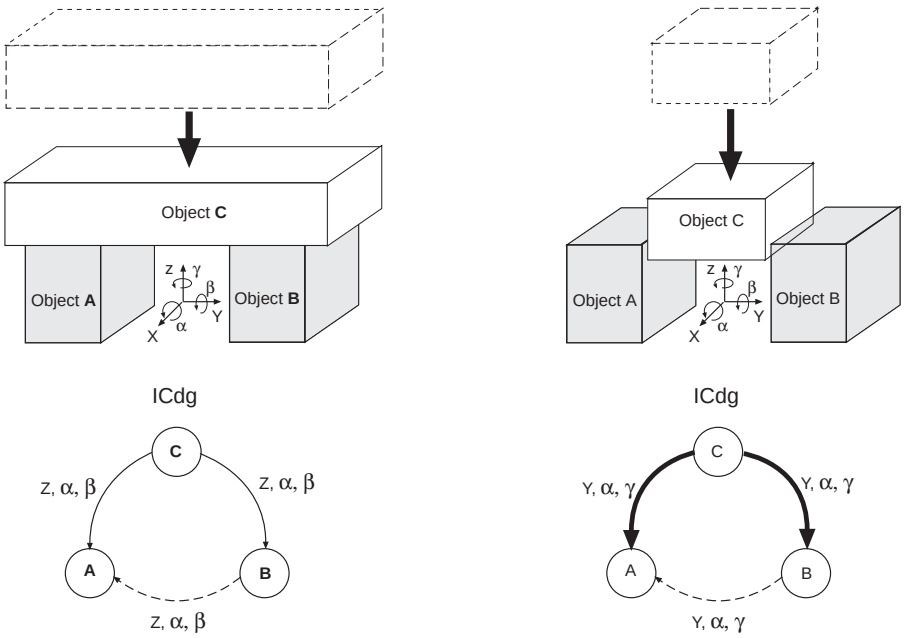
Figure 3.4: Types of arcs used to record alignment constraints that describe relations among objects in 2-D space.

with respect to the Y position of Object A. Y and θ are the constrained DOF in this contact relation, and since there are not contacts between the environmental objects that could help to verify the alignment constraints imposed by the task, these DOF describe the critical dimensions requiring preventive vision. A dashed arc, from the latter pre-assembled environmental object to the other environmental object, is used to represent this type of dependency relation. The most important result here is that the contact with multiple objects produce these kind of arcs by propagating the alignment constraints up to the constraining objects.

Figure 3.4(b) depicts a configuration produced by the insertion of a manipulated object (Object C) into an insertion slot formed by two environmental objects (Objects A and B). Again, assuming a common coordinate reference frame for the three objects, conveniently aligned with the direction of near contacts produced by the insertion, the solid arcs between Object C and objects A and B represent alignment constraints due to their near contacting faces. Usually, insertion operations include some clearance (a separation) between the inserting features and the features of the insertion slot, but since insertion is a low tolerance operation, deviation from the planned configurations

could be fatal. Then assuming a successful insertion, similar alignment constraints to those described by actual contacts are assumed. In the illustrated insertion case, the near contacting faces are aligned in such a way that the constrained DOF are X and θ.

Also, in this case, objects A and B were not related before the current operation. For succeeding in the insertion, controlling the pose of Object C is not enough. Additional alignment requirements, for this operation, are that object B have the same orientation than object A (same value for θ parameter) and that the orthogonal distance (related to the X parameter) between the opposing faces describing the insertion slot is not reduced. X and θ are the constrained DOF in this insert relation, and since there are not contacts between the environmental objects that could help to verify the alignment constraints imposed by the task, these DOF describe the critical dimensions that need of using visual sensing. A dashed arc from the latter pre-assembled environmental object to the other environmental object, is used to represent this type of dependency relation. Again, this kind of arcs also result from the propagation of the alignment constraints up to the constraining objects.



(a) Arcs produced by a make-contact assembly skill primitive.

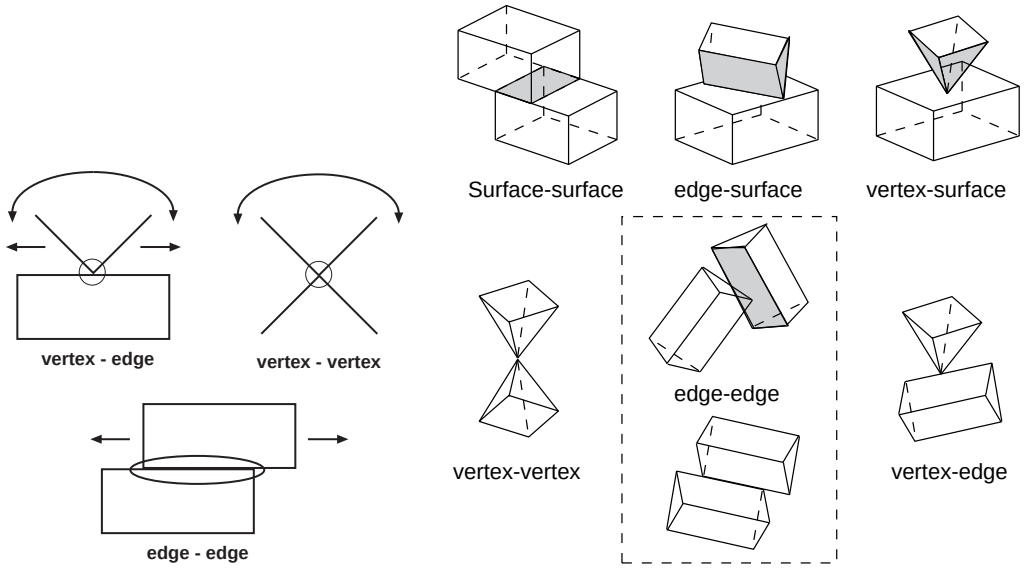
(b) Arcs produced by an insertion assembly skill primitive.

Figure 3.5: Types of arcs used to record alignment constraints that describe relations among objects in 3-D space.

In the case of assembly operations in a 3-D space, the motion parameters involved are six, three translational DOF, for describing position, and three rotational DOF, for describing orientation. These constrained DOF are presented in Figure 3.7. The same

three types of arcs are used to express the dependency between inserting and contacting objects. Figure 3.5 illustrates typical situations that produce each kind of arc.

The sensing analyzer takes advantage of the configuration of the features in contact to determine the alignment constraints that require of using vision. When an object enters in contact with other objects, its freedom of motion is reduced. As seen, this reduction depends on the type of contact among the objects. Figure 3.6 depicts the different types of contact relations that could be formed between two objects in the 2-D and 3-D cases.



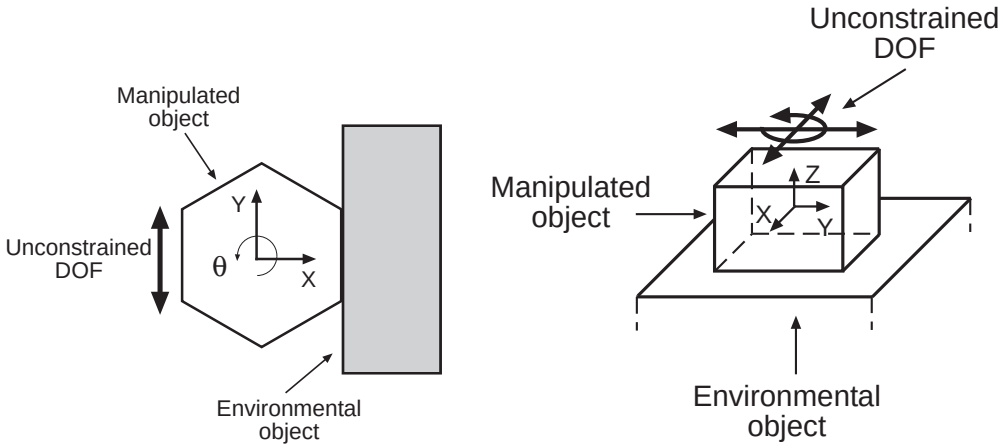
(a) 2-D contact types.

(b) 3-D contact types.

Figure 3.6: Types of contact relations between polyhedral objects.

As mentioned before, in this study, the contact relations are limited to those described by edge-edge contacts in the 2-D case or those described by surface-surface contacts in the 3-D case. These are also referred as face contact relations. A 2-D edge-edge contact eliminates two possible critical dimensions, the rotational DOF and one translational DOF with respect to a coordinate frame of reference where one coordinate axis is conveniently aligned with the normal of the contacting edges (see Figure 3.7(a)). A 3-D surface-surface contact eliminates three possible critical dimensions, two rotational DOF and one translational DOF with respect to a coordinate frame of reference where one coordinate axis is conveniently aligned with a normal of the contacting surfaces (see Figure 3.7(b)).

An ICdg is constructed incrementally by following the assembly steps order. It starts containing only the nodes associated with fixed environmental objects and evolves through the addition of new nodes and arcs. A new node is introduced to the ICdg



(a) 2-D edge-edge contact.

(b) 3-D surface-surface contact.

Figure 3.7: Constraining effects of face contact relations.

for each assembly part manipulated by the robot for the first time. New arcs describing direct dependency relations are introduced as consequence of tasks including make-contact assembly skill primitives and insertion assembly skill primitives.

Since the same object can be manipulated several times by the robot, a decision had to be taken with respect to the evolution of the structure of the ICdg. The alternatives considered were:

1. Add a new node each time an object is manipulated, and consequently, have several nodes associated with the same object in the same ICdg, each with its own dependency arcs;

2. Maintain only one node for each object in the ICdg, and add new arcs for new dependency relations and erase arcs associated with broken dependency relations;

3. Maintain only one node for each object in the ICdg, and add new arcs for new dependency relations and for each maintained dependency relation, indexing each arc with the number of the assembly step in which it was added; and

4. Generate a new ICdg for each assembly step.

The second option was discarded because it would forget dependencies that could be critical and require of preventive vision. The fourth option was discarded because it is similar to the first one, but includes more redundancy that is needed. Finally, the third option was discarded because even being a little less redundant than the first option, it makes more difficult and inefficient the bookkeeping and access to the information

in the data structure. In conclusion, a new node with its own arcs is added for each assembly step, inclusive when its manipulated object had already been manipulated in previous steps.

3.2.2 Propagation of Alignment Constraints

As shown in the preceding section, the propagation of alignment constraints in an ICdg starts from a currently manipulated object up to the environmental objects. A constraint propagation pattern involves at least three objects and is fired up by at least one of the following three conditions:

First condition: an object is inserted or enters in contact with multiple environmental objects making the object configuration to depend upon theirs and the defined dependencies include common constrained DOF.

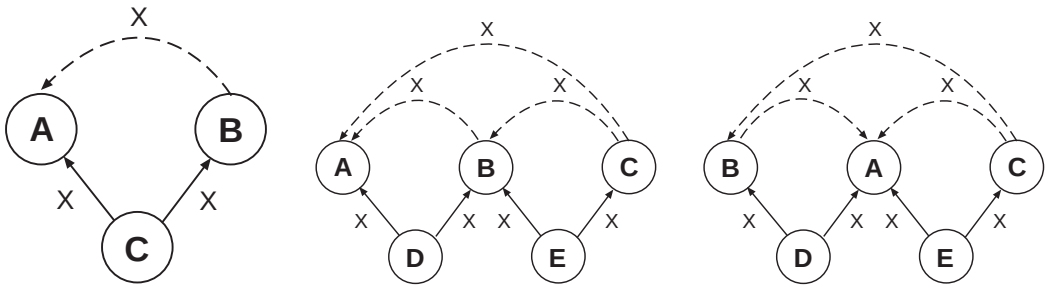
For example, Figure 3.8(a) illustrates the result of an operation that puts Object C in contact with environmental objects A and B. After the operation, X DOF of Object C can be described from the correspondent DOF of objects A or B. If Object B is a pre-assembled object that was manipulated more recently than Object A, an indirect dependency of Object B with respect to Object A was made explicit (dashed arc). A constraint produced by this reason will be further referred as *joining constraint*.

Second condition: an object is inserted or enters in contact with multiple environmental objects, and when analyzing the dependency between a pair of environmental objects is realized that one of the objects depends on the other and this other already depends on a third environmental object; in both cases the dependencies include common constrained DOF.

For example, Figure 3.8(b) illustrates the result of an operation that puts Object E in contact with the environmental objects B and C, while Object B was already a dependent of environmental Object A. Since both dependencies include the same X constrained DOF and assuming that Object A was manipulated before Object C, Object C will inherit the dependency of Object B on Object A. An indirect dependency of Object C with respect to Object A was made explicit(dashed arc). A constraint produced by this reason will be further referred as *inherited constraint*.

Third condition: an object is inserted or enters in contact with multiple environmental objects, and when analyzing the dependency between a pair of environmental objects is realized one of two situations: one of the objects depends on the other and depends on a third object, or the object on which the other depends has already a third object that depends on it. In both situations the pairs of dependencies include common constrained DOF.

For example, Figure 3.8(b) illustrates the result of an operation that puts Object E in contact with the environmental objects A and C, while environmental Object B was already a dependent of Object A. Since both dependencies include the same X constrained DOF and assuming that Object B was manipulated before Object C, Object C will share the dependency of Object B on Object A. An indirect dependency of Object C with respect to Object B was made explicit (dashed arc). A constraint produced by this reason will be further referred as *shared constraint*.



(a) Propagation of joining constraints.

(b) Propagation of inherited constraints.

(c) Propagation of shared constraints.

Figure 3.8: Type of alignment constraints resulting from propagation.

The assembly operation applied to the manipulated object, which produces new direct dependencies, only fires up the constraint propagation process. After it starts, propagation of constraints makes explicit some indirect dependencies among the environmental objects, which were implicit before the operation. These new dependencies could then propagate additional joining, inherited, and shared constraints among other environmental objects. Therefore, a constraint propagation scheme had to be used for ensuring that all the indirect dependencies are found.

The constraint propagation scheme, used in this dissertation, was developed based on the following knowledge:

Direct dependency Object B directly depends on Object A if Object B participates in insert relations or contact relations with Object A and Object A was an environmental object during the manipulation of Object B.

Indirect dependency Object B indirectly depends on Object A if Object B participates in insert relations or contact relations with other objects that relates directly or indirectly with Object A through the same constrained DOF, and Object A was an environmental object during the manipulation of Object B.

Propagation Scope The constraint propagation has to be limited to nodes that represent objects, that directly or indirectly affects the successful execution of insertion operations and make-contact operations. These objects have to participate in relations that include constrained DOF common to those described in the new direct dependencies of the manipulated object.

Prevalence of dependencies among environmental objects The past dependencies defined among current environmental objects are maintained by the environmental objects of a new assembly operation.

Renewal of dependencies for the manipulated object Since the level of constraint among objects can grow or decrease as a result of an assembly operation, when an object is manipulated more than one time, its past dependencies with environmental objects have to be actualized every time that it is manipulated.

Completeness of a direct-dependency graph (Ddg) All the objects in a graph constructed only from the complete set of direct dependencies, depends on each other. This graph is complete because there is no object outside it that have a dependency (direct or indirect) with an object included in it.

The generation of indirect dependencies for the environmental objects is not monotone. The discovery of new dependencies fires up additional propagation of constraints. This situation naturally requires of a search mechanism that includes backtracking. The constraint propagation method takes advantage of the knowledge described as the completeness of a direct-dependency graph. After the direct dependencies of the manipulated object with respect to the environmental objects are determined, the objects included in a direct dependency graph can be found, and indirect dependencies can be defined among objects that are not related directly.

Not all the indirect dependencies are relevant for visual sensing. The indirect dependencies that are relevant are only those that relate with critical dimensions for a future assembly task, when these critical dimensions are not enforced by the direct dependencies existing before the manipulation of the environmental objects is performed.

Since an environmental object could be manipulated several times before the task that made an indirect dependency relevant, the constraint propagation method has to determine the manipulation step where vision will be useful for preventing any possible error with respect to the critical dimensions.

The constraint propagation method is applied for each assembly step. It starts determining all the face contacts in which the manipulated object participates, and finishes when all the direct dependencies for the manipulated object and the new indirect dependencies for the environmental objects are deduced.

3.3 Dependency Relations and Their Propagation

In the preceding sections, the critical dimensions of an assembly task are described as DOF associated with the coordinate axes of a reference frame defined in a *convenient* way, but what does it make a definition of such a reference frame convenient?.

If a reference frame is arbitrarily selected, for example as the reference frame of the world or as the reference frame of one of the objects involved in the task, there is a risk of finishing with more critical dimensions than strictly necessary. Such situation is clearly illustrated in Figure 3.9(c) where the critical dimensions for Object 2 were determined. If instead of aligning the reference frame as depicted it were aligned as in the other two illustrated cases, the result would be two critical dimensions, one associated with X axis and another with Y axis. However, after orienting the reference frame in such a way that Y axis is aligned with the contact surface between the two environmental objects, the critical dimensions were reduced to only one associated with its X axis.

Based in a similar reasoning, a method was developed to define *convenient* orientations for coordinate frames of reference to describe the critical dimensions of an assembly task.

3.3.1 A Reference Frame to Describe Dependency Relations

Since the critical dimensions describe the dependencies of the configuration of the object that is manipulated in the task upon the configuration of its assembly environment, there will be one reference frame for each environmental object on which it depends. This means that there will be a reference frame associated with each arc in the ICdg.

Because contacts considered in this work are of type surface-surface and since a task can produce several contacts (or several constraint relations in the case of insertion), the method obtains the reference frame by applying the Gram-Schmidt procedure [95] to a set of normal vectors that describe the orientation of surfaces that directly or indirectly participate in a new contact state.

The Gram-Schmidt procedure finds orthonormal vectors $\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_k$, from a set of independent vectors $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k \in \Re^n$, s.t $span(\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_k) = span(\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k)$, where $span(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k) = \{\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + ... + \alpha_k\mathbf{v}_k | \alpha_i \in \Re\}$. These resultant vectors describing an orthonormal basis will represent axes of the reference frame for the representation of the critical dimensions of an assembly step.

A method that implements the Gram-Schmidt procedure finds the $\mathbf{q}_i$'s recursively as:

- $\tilde{\mathbf{q}}_1 := \mathbf{a}_1$

- $\mathbf{q}_1 := \tilde{\mathbf{q}}/\|\tilde{\mathbf{q}}\|$ (normalize)
- $\tilde{\mathbf{q}}_2 := \mathbf{a}_2 - (\mathbf{q}_1^T\mathbf{a}_2)\mathbf{q}_1$ (remove $\mathbf{q}_1$ component from $\mathbf{a}_2$)
- $\mathbf{q}_2 := \tilde{\mathbf{q}}_2/\|\tilde{\mathbf{q}}_2\|$ (normalize)
- $\tilde{\mathbf{q}}_3 := \mathbf{a}_3 - (\mathbf{q}_1^T\mathbf{a}_3)\mathbf{q}_1 - (\mathbf{q}_2^T\mathbf{a}_3)\mathbf{q}_1$ (remove $\mathbf{q}_1$ and $\mathbf{q}_2$ component from $\mathbf{a}_3$)
- $\mathbf{q}_3 := \tilde{\mathbf{q}}_3/\|\tilde{\mathbf{q}}_3\|$ (normalize)
- etc.

However, since $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k$ will commonly be dependent, it will be found that $\tilde{\mathbf{q}}_j = 0$ for some $\mathbf{a}_j$'s which means that $\mathbf{a}_j$ is linearly dependent on $\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_{j-1}$. In this case, it is necessary to modify the algorithm in such a way that when it encounters that $\tilde{\mathbf{q}}_j = 0$, it skips to next vector $\mathbf{a}_{j+1}$ and continue. The general Gram-Schmidt procedure is implemented as:

```
1: procedure GRAM-SCHMIDT
2:     r ← 0
3:     for i ← 1, k do
4:         ã ← a_i − Σ_{j=1}^{r} q_j q_j^T a_i
5:         if ã ≠ 0 then
6:             r ← r + 1
7:             q_r ← ã/‖ã‖
8:         end if
9:     end for
10: end procedure
```

All the *reference vectors* – the vectors representing relevant alignment constraints, which are used in the orthonormalization process –, if any exist, have to be computed before applying the Gram-Schmidt procedure. This means that the reference frames will be computed until all the nominal assembly plan had been processed, then it will be the last step of the preventive vision analysis. As will be explained in the next chapter, this process is the first step of the sensor planning method. The reference axes directions are defined by the order in which the reference vectors are processed.

3.3.2 Obtaining Reference Vectors

As mentioned before, when an assembly operation joins two or more environmental objects through contacts with the manipulated object, indirect dependencies among these environmental objects has to be represented through reference vectors.

The first step of a method for obtaining reference vectors is to identify all the contacting face pairs of the manipulated object with its environment to form two groups: one group G_m for maintained contacting face pairs and one group G_n for new contacting face pairs. Each contacting face pair describe a constraining vector. The constraining vector described by a contacting face pair is a normal vector that describe the orientation of an environment object's face. Inside each group, constraining vectors with exactly the same orientation or exactly the opposite orientation are grouped. Each group orientation describe a vector further referred as an *alignment vector*.

Determining the reference vectors for an environmental object is difficult because its participation in the insert or contact configuration could be indirect, s.t. it only restricts the feasible configurations of other environmental objects that directly or indirectly participate in the current task. Even if an environmental object directly participate in the configuration for the current task, the difficulty resides in identifying its roll in the current task, discovering its motion freedom during its last manipulation, and realizing the possible propagation of additional alignment constraints to other environmental objects.

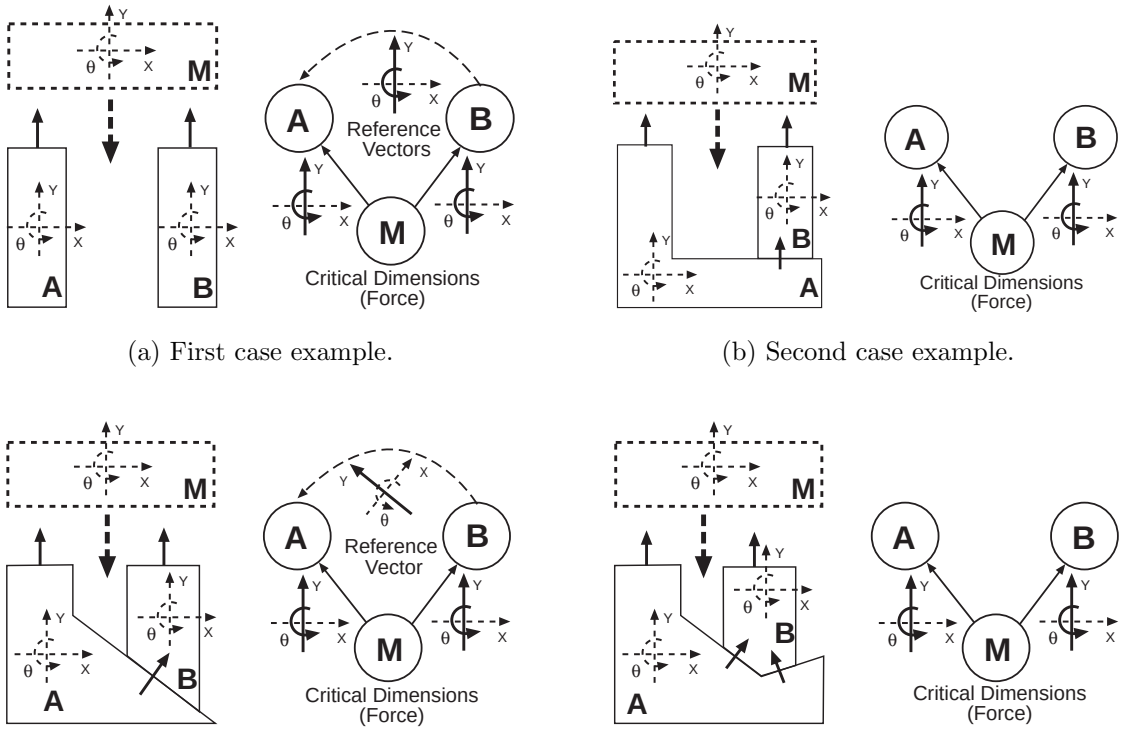
Every environmental object in the insert configuration and/or contact configuration, required by the task, has to be analyzed for reference vectors with respect to the other environmental objects that are in contact with the manipulated object, further referred as *objects of reference*. The analysis is performed in the inverse order of manipulation, s.t. the first object to analyze is the most recently manipulated object before the current task. The analysis is effected for each alignment vector $\mathbf{a}_i$ of G_n, considering one at the time.

Since, commonly, an analyzed object already participate in contact relations with other environmental objects, that conformed its own assembly environment, in its last manipulation, its feasible configurations during the current task are restricted by such relations. Then, in getting the reference vectors for this environment object the following three cases are considered:

First case: before the current task, either the environmental object does not participate in any contact with its assembly environment or all the contacts in which it participates are described by vectors $\mathbf{n}_j$ in orthogonal directions to the considered alignment vector, s.t. $\forall \mathbf{n}_j \{\mathbf{a}_i \cdot \mathbf{n}_j = 0\}$.

In this case, the alignment vector a reference vector for the environmental object. The orthogonal contacts are ignored because they do not affect the environmental position in the alignment direction, but as will be explained later, they could be used to eliminate critical dimensions for rotation.

Figure 3.9(a) depicts an example of this case since environmental Object B does not participate in any contact with other environmental objects.



(a) First case example.

(b) Second case example.

(c) Third case example.

(d) Contact configuration in which a reference vector is discarded.

Figure 3.9: Obtaining reference vectors to deduce a reference frame for the description of critical dimensions.

Second case: before the current task, at least one of the contacts in which the environmental object participates is described by a vector $\mathbf{n}_j$ in a parallel direction to the considered alignment vector, s.t. $\exists \mathbf{n}_j \{\mathbf{a}_i \cdot \mathbf{n}_j = \|\mathbf{a}_i\| \|\mathbf{n}_j\|\}$.

In this case, since the position of the analyzed environmental object is completely determined with respect to the alignment direction due to the parallel contact, there is not need to include new reference vectors.

Figure 3.9(b) presents an example of this case since the contact of environmental Object B with its environmental Object A is in the same direction as the future contact with manipulated Object M.

Third case: before the current task, any of the contacts in which the environmental object participates is described by a parallel vector to the current alignment vector, although at least one of the contacts has a vector $\mathbf{n}_j$ with a component parallel to the alignment direction, i.e. $\forall \mathbf{n}_j \{(\mathbf{a}_i \cdot \mathbf{n}_j = 0) \vee (\mathbf{a}_i \cdot \mathbf{n}_j \leq \|\mathbf{a}_i\| \|\mathbf{n}_j\|)\}$.

In this case, the reference vectors are obtained as orthogonal vectors to normal vectors describing faces contacting other environmental objects when such vectors

have components in the direction of the critical vector. Then, for each contact of the environmental object, a reference vector $\mathbf{r}_i$ is computed as:

1: $\mathbf{q}_j \leftarrow \mathbf{n}_j / \|\mathbf{n}_j\|$
2: $\tilde{r}_i \leftarrow \mathbf{a}_i - (\mathbf{q}_j \cdot \mathbf{a}_i)\mathbf{q}_j$
3: $\mathbf{r}_i \leftarrow \tilde{\mathbf{r}}_i / \|\tilde{\mathbf{r}}_i\|$

where $\mathbf{r}_i \neq \tilde{0}$.

Figure 3.9(c) is an example of this case since the contact of environmental Object B with environmental Object A does not completely determine its final Y position. If the face of Object B contacting Object A would not have been considered, the resultant reference frame would be oriented as the other two examples and the only critical dimension would be that associated with its Y axis. Then, when assembling Object B a conflict between aligning and contacting requirements would appear. To fulfill both requirements a new critical dimension associated with X axis had to be defined, but that is not evident only from the new indirect dependencies produced by the future task of moving to contact of manipulated Object M. Using the new frame aligned with the contacting surface this problem is solved.

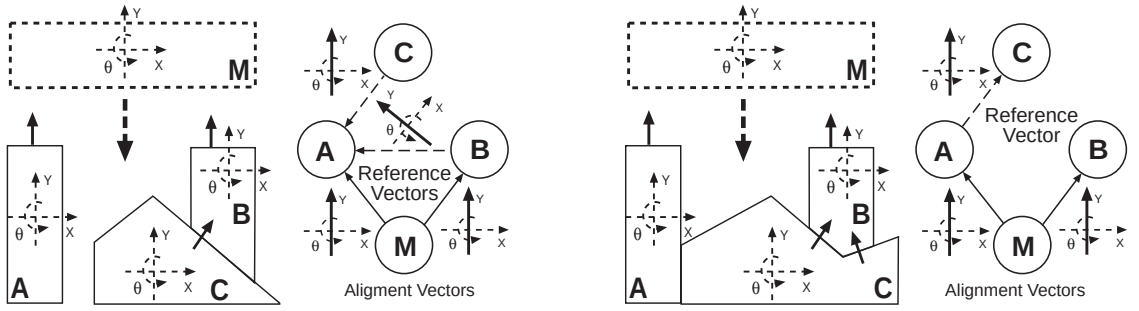
However, when in the this case, multiple contacts exist between Objects A and B, before defining or modifying their dependency relation the need of maintaining the new reference vector has to be evaluated. The reference vector should be discarded if any other contact has a component aligned with it, s.t. $\mathbf{r}_i \cdot \mathbf{n}_j \neq 0$, because this contact impedes motions aligned with such reference vector, as shown in Figure 3.9(d).

3.3.3 Additional Propagation by Configuration Conditioning

Environmental objects are analyzed for the deduction of new critical dimensions when they participate directly or indirectly in insert configurations or in contact configurations as described in Section 3.2.2. Both second case and third case of the method described above are based on the pre-existence of non-orthogonal contacts of an environmental object analyzed with other environmental objects, but this description does not specify if these contacts are with respect to the object of reference for the deduction of new indirect dependencies, or if there are contacts with other environmental objects not considered originally.

The actions described in the second and third cases apply as described when the non-orthogonal contacts of the analyzed object, Object B, happen with an object of reference, like Object A shown in Figures 3.9(b) and 3.9(c).

If in the third case, there are contacts of Object B against another environmental object,



(a) Contact requiring a new dependency arc in the ICdg.

(b) Contact configuration in which a reference vector is discarded.

Figure 3.10: Contacts requiring additional propagation of alignment constraints.

Object C, a new indirect dependency arc should be added to the ICdg between these ($B \rightarrow C$). If this arc already exists, the new reference vector should be included as part of it. In addition, in both, second case and third case, the constraint propagation process should continue, but now to analyze a possible new dependency between Objects A and C. This additional propagation of alignment constraints is needed since Object C was not considered originally because it does not participate directly as an element of the insert configuration or contact configuration that fired the propagation process. Figure 3.10(a) illustrates this situation where a new indirect dependency of Object B on Object C has been produced when analyzing a possible dependency relation with Object A; assuming that Object C was assembled before Object A, a new dependency of Object A on Object C was also deduced.

Same as before, when there are multiple contacts in the third case, before including new reference vectors in the relation between Objects B and C, the need for each has to be evaluated using the information of the other contacts, as explained before. Figure 3.10(b) presents an example in which a possible reference vector between Object B and Object C has been discarded; in this example, it is assumed that Object A was assembled before than Object C. The additional propagation of alignment constraints is not affected when reference vectors are discarded.

The additional propagation of alignment constraints between Objects A and C requires of the recursive execution of the following two steps:

1. Determine the new environmental object to analyze. This object, A or C, is the object that was manipulated later, the other was part of its environment in such assembly step.

2. Repeat the process of obtaining new reference vectors for the new environmental

object to analyze.

3.3.4 Rotational Considerations

Although the basis created from the reference vectors describe critical dimensions for pure translation, a rotation error could have a similar translational effect in critical directions, and has to be considered when determining the critical dimensions of an assembly operation. However, representing general rotational freedom is more difficult than representing translational freedom and consequently complicates the description of their critical dimensions.

As mentioned in Chapter 2, in this work, the followed approach takes advantage of the constraining effect of contacts for determining the rotation references. Also, the centers of rotation of polygonal objects in 2D-assembly is situated in the origin of their coordinate frame, and the critical rotation axes pass through the origin of the coordinate system of the polyhedral objects that represent the assembly parts.

Analysis for Planar Assembly

For planar assembly – assembly in 2D space – a simple contact fixes the attitude of an object. When an assembly task puts the manipulated object in contact with multiple environmental objects, its successful execution depends on the correct orientation of the most recently manipulated environmental objects respect to the orientation of the other environmental objects assembled before than them. Then after determining the environmental objects that conform the direct insert configuration or direct contact configuration for a task, these objects are inserted in a priority queue E_r. The priority of an object is proportional to the number of the last assembly step in which each it was manipulated. The recursive method to determine if the attitude of an environmental object has to be observed, for a possible adjustment with respect to other objects, work as follows:

1: **procedure** RotationDependency-2D
2: **if** E_r contains only one object **then**
3: Finish
4: **end if**
5: Extract the first object *obj* of E_r
6: Create a list C_e with other objects assembled before that *obj* contacting it.
7: **if** C_e is empty **then**
8: Add a dependency relation for *obj* with respect to each object in E_r
9: Call RotationDependency-2D to process the rest of E_r

10: **else**
11: Add the objects in list C_e into E_r, if they are not already in this priority queue
12: Call RotationDependency-2D to process the rest of E_r
13: **end if**
14: **end procedure**

Analysis for Assembly in 3D-space

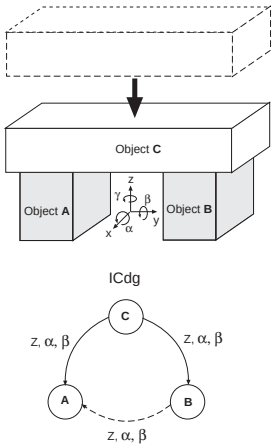
The method for determining critical rotation axes for environmental objects in 3D-assembly works in a similar way to that for planar assembly. However, some steps must realize a more complex geometric reasoning, because of the topology of spatial contacts and because in this case, the method deals with multiple rotation axes and not only a rotation center.

Since after a single face contact two possible rotation axes are fixed, in the case of propagating rotation constraints to environmental objects the following situations can apply:

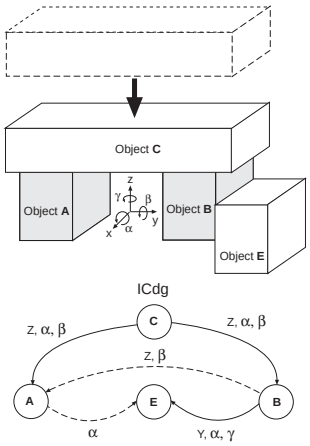
1. Before the task, the analyzed environmental object *obj* is not in contact with any other environmental object. In this case, two critical rotation axes should be defined to align its contacting faces accordingly to the alignment vector of the task. These critical rotation axes define a basis with the alignment vector.

   Figure 3.11(a) presents an example of this case. In the example, the objects are assembled in the order: $A \Rightarrow B \Rightarrow C$. Object C is the manipulated object, and objects A and B conform its direct contact configuration. Assuming that all the objects have coordinate frames aligned as depicted, the alignment vector is aligned with the Z axis. For this task vision is not required for the manipulated object, only the force compliant skills have to be used to assure the position of the object with respect to its Z position, and its orientation with respect to X and Y critical rotation axes. Since environment object B was the most recently manipulated object, and it does not participate in contacts before the task, to assure a correct environment configuration, its configuration will depend on the configuration of environmental object A; specifically its position with respect to the Z axis, and its orientation with respect to X and Y axes, which describe its critical dimensions. All the critical dependencies are illustrated in the ICdg.

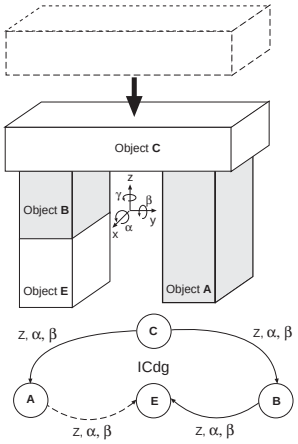
2. Before the task, *obj* only participates in contacts with other environmental objects, in its own manipulation environment, that are described by orthogonal vectors to the alignment vector of the task. Moreover, in case of multiple contacts, all of them have exactly the same direction or exactly the same opposite direction. In



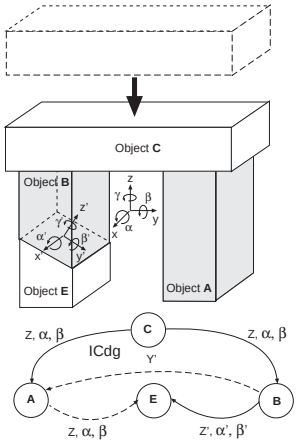
(a) First case example.



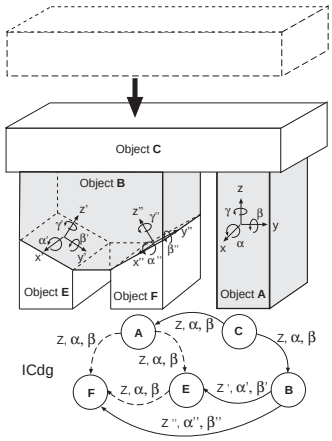
(b) Second case example.



(c) Third case example.



(d) Fourth case example.



(e) Fifth case example.

Figure 3.11: Obtaining critical rotation axes.

this case, there is only one critical rotation axis that should be aligned in the direction of the contacts.

Figure 3.11(b) exemplify this case. In this example, the task and its direct environment configuration is the same as those of the example for the first case. The important element in this example is the addition of an new environmental object, Object E, which is assumed to be assembled before Object A. Since Object E participates in a simple contact relation with Object B, and this contact is described by a vector aligned with Y axis which is orthogonal to the alignment vector of the task, the method arrive to the following conclusions:

- Use force compliance for controlling the Y position of Object B and its orientation with respect to X and Y axes, based in the configuration of Object E.
- Observe the Z position and Y orientation of Object B with respect to Object A. The X orientation which it is also critical for succeeding in the task, was eliminated because it is enforced by the contact with Object E.
- Observe the X orientation of Object A with respect to Object E because Object E fixes this orientation for Object B.

3. Before the task, *obj* participates in at least one contact with other environmental object, in its own manipulation environment, that is described by a parallel vector to the alignment vector of the task. Since this contact fixes the correct attitude of the object, there is not a critical rotation axis for this situation.

   Figure 3.11(c) presents an example of this case. In this example, the task and its direct environment configuration is also the same as those of the example for the first case. Same as the example of the second case, there is a new Object E assembled before than Object A, but in this case the contact between Objects B and E is in the same direction as the alignment vector. Then, Object E does not need to be observed because its critical position and orientation parameters are enforced by Object E. However, the complete alignment dependency is propagated resulting in a having to observe the position and orientation of Object A with respect to Object E.

4. Before the task, *obj* only participates in contacts with other environmental objects that are described by vectors that are neither orthogonal nor parallel to the alignment vector. However, in case of multiple contacts, all of them have exactly the same direction or exactly the same opposite direction. In this case, there is only one critical rotation axis that should be aligned in the direction of the contacts.

   Figure 3.11(d) illustrates an example of this case. Again, the task and direct environment configuration are the same of the other examples, and the Object E is the first assembled object that participates in a contact relation with Object B.

However, in this example the contact is neither orthogonal nor parallel to the alignment vector. Then the Z position of Object B depends on two factors: one, the Z position of Object E, and two, the Y' position of Object B with respect to the introduced coordinate frame illustrated in the tilted plane between objects B and E. This Y' position must be observed. Then, since Object A is assembled after Object E, its Z position and X and Y orientation must to be observed to verify a correct assembly configuration for the task.

5. Before the task, *obj* participates in multiple non-parallel contacts with other environmental objects that are described by vectors that are neither orthogonal nor parallel to the alignment vector. In this case, there is not a critical rotation axis for this situation, as exemplified in

   Finally, Figure 3.11(e) exemplifies this last case. This example presents the same task and direct environment configuration of the others cases and includes the environmental object E, but in addition, it includes a fifth object, Object F, which is assumed to be assembled before than Object E. In this example, the critical configuration of Object B is completely fixed by objects E and F. Only the contacts have to be verified by force, which is indicated by the solid lines between Object B and objects E and F. The relations to observe are the Z position and X and Y orientation of Object A with respect to Object E or Object F, and the same dimensions of Object E with respect to Object F.

As can be noted, these cases also appear in the analysis for critical dimensions for translation. The first and second cases of the rotation analysis correspond to the first case of the translation analysis; the third case for rotation corresponds to the second case for translation; the fourth case for rotation corresponds to the third case for translation; and the fifth case for rotation corresponds to the case of elimination of reference vectors for translation.

The process for the propagation of alignment constraints that determine new indirect rotation dependencies among environmental objects is also the same as that followed to determine indirect translation dependencies. A possible propagation has to be analyzed always that an environmental object that configure the environment of a current task is in direct contact with other environmental objects that do not.

3.4 Sensing Analysis for Robotic Assembly: Putting It Together

Summarizing all the previous sections, a method was developed to plan sensing strategies for robotic assembly. This method includes force sensing requirements to control and

correct the execution of assembly operations with objects in contact and visual sensing operations to prevent the appearance of critical deviations that would make subsequent assembly operations fail.

The sensing requirements for an object result from an analysis of the operations where it is manipulated by a robot arm, and from an analysis of its roll as an environmental object in insertion relations and contact relations with other environmental objects.

The force sensing requirements are deduced by recognizing assembly steps that include make-contact, slide, and insert assembly skill primitives. A force sensing requirement specify the type of assembly skill primitive to implement, the configuration of contacts to produce, and its critical dimensions. This information could be used by the force compliance skills.

A configuration of contacts is described as a list of pairs of faces. Every pair describe a surface-surface contact relation. In the case of a make-contact skill, the pairs describe faces that are expected to enter in contact; in the case of a slide skill, the pairs describe the sliding faces; and in the case of an insert skill, the pairs describe the constraining and constrained faces.

The visual sensing requirements are deduced by recognizing assembly steps that include insert assembly skill primitives and assembly steps that define contact relations of the manipulated object with multiple environmental objects.

A high-level outline of the sensing analysis method can be described as follows:

1. Create the CAD models for the assembly parts.
2. Load the CAD models and the nominal assembly plan to create the sequence of assembly steps.
3. For each assembly step:
   (a) Identify the new contact-state relation.
   (b) Determine the contact-state transition.
   (c) Extract force sensing requirements for the manipulated object.
   (d) Actualize the ICdg.
      - Identify new visual sensing requirements for the manipulated object.
      - Propagate new alignment constraints, if exist, that require of preventive vision for the environmental objects.
4. Compute the reference frames for each dependency arc of the ICdg.
5. Reconstruct the assembly plan including force and vision sensing requirements.

3.5 Experiments and Discussion

A computer application of the proposed method has been implemented in C++. The CAD models of the objects are created with a constructive solid modeling (*CSG*) tool known as VANTAGE [41](specifically a C++ version created at Carnegie Mellon University by the author of this dissertation and fellow George V. Paul) that gives the additional benefit of direct access to its basic data structures, the topological features of the objects, its geometric functions, and its graphical interface.

A nominal assembly plan for this application is described by a sequence of assembly steps specified in a lisp-like syntax. Every assembly step specify the type of assembly operation, the name of an assembly part to be manipulated, the name of the VANTAGE object that represent the model of the assembly part, and the motion parameters as a list of six values (three for translation and three for rotation).

The method and program were intensively tested. To test the method, the program was presented with many nominal assembly plans devised to include some assembly operations that did not required of any type of feedback, some assembly operations that did require only of force sensing operations or only of visual sensing operations, and some assembly operations that required of both types of sensing, force and vision.

In all the test cases, the method behave as expected. The application that implements the method successfully identified the assembly steps that needed of sensing feedback information and the type of force and visual feedback operations and information required.

Next, we present three cases solved using the proposed method, to illustrate its behavior. Every case and their solutions are depicted by five figures that present:

1. the VANTAGE commands used to define the models of the assembly parts involved,
2. the description of the nominal assembly plans,
3. a graphic sequence of the assembly steps (as displayed by the system),
4. an illustration of the final ICdg, and
5. the new assembly plans including the sensing feedback operations.

3.5.1 Experimental Case 1

In this case there is only one type of object involved, *body1*. *body1* is a rectangular parallelepiped (a kind of box). Its CSG model is constructed with the VANTAGE

command presented in Figure 3.12. VANTAGE produces rectangular parallelepipeds with the CUBE primitive. The origin of the object's coordinate frame is situated in its center of geometry.

```
// BODY1
(csgnode body1 cube (30 100 40) (0 0 0 0 0 0))
```

Figure 3.12: VANTAGE model of parts for test case 1

The nominal assembly plan containing the sequence of assembly steps for the case is presented in Figure 3.13. There are five objects of the same type of *body1* involved, *block1*, *block2*, *block3*, *block4*, and *block5*. An assembly plan file first specify the name of the file that contains the VANTAGE commands that construct the models of the objects, one for each kind of assembly part. Then it specify commands that describe the sequence of manipulation of the assembly elements. The current version of the program only accepts three manipulation commands: create-fixed-object, which introduces environmental object, not manipulated by the robot, and that are fixed during the full assembly; create-assembled-object, which introduces new objects manipulated by the robot; and move-object, which is used to manipulate an object that was already introduced to the assembly scene. The system assumes that an object that is manipulated by the first time comes from a completely unconstrained contact state, an S contact state.

```
// Load the models of assembly parts

load-models: case1.model.van

// Sequence of assembly

create-assembled-object: block1 body1 (-60 0 0 0 0 0)
create-assembled-object: block2 body1 (60 0 0 0 0 0)
create-assembled-object: block3 body1 (0 0 0 0 0 0)
create-assembled-object: block4 body1 (-30 0 0 0 0 0)
move-object: block1 (-120 0 0 0 0 0)
create-assembled-object: block5 body1 (30 0 0 0 0 0)
```

Figure 3.13: Nominal assembly plan for test case 1

create-fixed-object and create-assembled-object commands create new polyhedral objects from the model specified in the command. They compute the new configuration of the object by applying a 3D-transformation to the configuration of the model of reference. The 3D-transformation matrix is computed from six parameters included in the manipulation command, three for translation (the first three) and three for rotation (the last three).

In this first case, four of the five objects are assembled with a single assembly step. Then, the plan contains a sequence of six assembly steps, five described by create-assembled-object commands and one described by a move-object command.

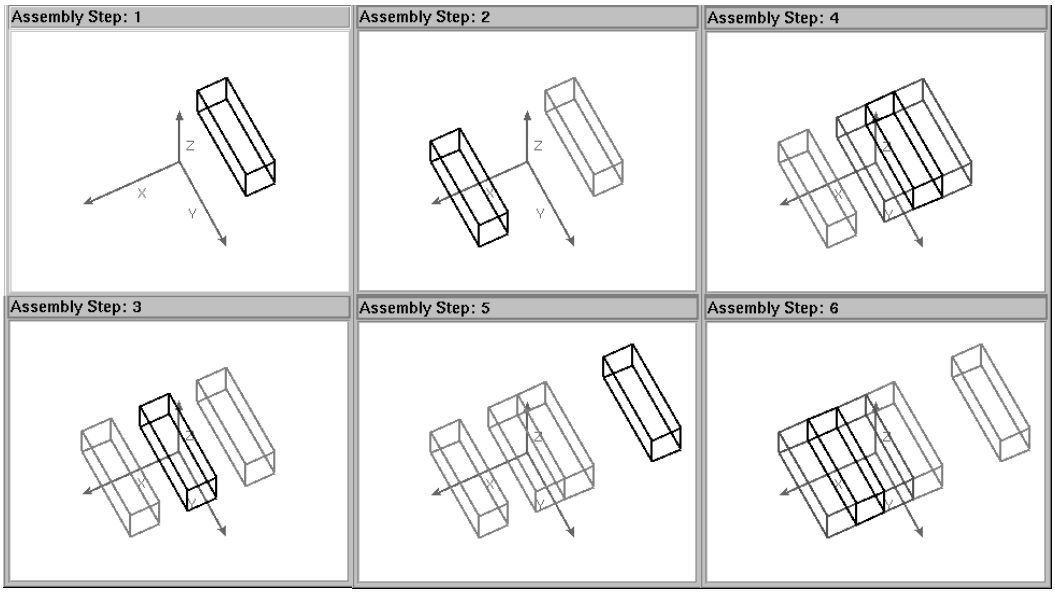


Figure 3.14: Sequence of assembly steps for test case 1

The system graphically illustrate each assembly step described in the plan. Figure 3.14 presents the sequence shown by the computer application. The assembly plan presents a kind of domino manipulation scheme where objects are juxtaposed, to form a line only through move and insert assembly skills. The figures of each step depict the object in three intensities. A black object represents the manipulated object. It also manage two gray levels to differentiate environmental objects in contact with the manipulated object (in a darker level) and other environmental objects (in a softer level).

For illustration convenience, none the test cases presented in this document includes a work table, then assuming that after the robot ungrasp an object, this object stays in its resultant pose, floating in the space. Another assumption is that all the objects have a coordinate frame aligned with the global coordinate frame for assembly.

Description and commentaries as the sensing analysis advance in each assembly step of the assembly plan for this case are:

Step 1: The *block1* object is moved to a completely unconstrained position, describing a $S \Rightarrow S$ transition. Skills graph presented in Figure 2.7 indicates that this type of assembly task requires of using the move assembly skill with respect to the three dimensions, and neither force nor vision feedback operations are required.

Step 2: The *block2* object is, similarly to *block1*, moved to a completely unconstrained position. No sensing requirements are added for this task either.

Step 3: The *block3* object is manipulated in the same manner as the preceding two objects, same conclusions in this case. However, this step completes the direct environment configuration, with *block1*, for the next assembly step.

Step 4: The *block4* object is inserted between *block1* and *block3*, then requiring of being visually observed. The step describe a $S \Rightarrow B$ transition, where the constrained dimensions is that associated with the X axis of the coordinate frames of reference. Since insertion usually requires of force control, from Table 2.1, a requirement is extracted for force compliance skills to detect contacts, keep La contacts, configuration control.

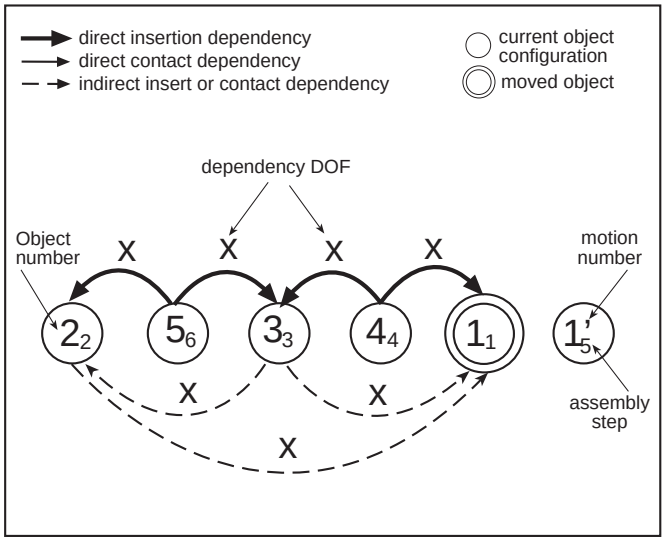


Figure 3.15: ICdg for test case 1

Vision must also be used to verify the insert configurations with respect to X DOF. Then a direct dependency relation of *block4* with respect to *block1* and *block3* is added to the ICdg of the task, as can be observed in the final ICdg illustration of Figure 3.15. This direct dependency relation is propagated to define an indirect dependency relation of *block3* with respect to *block1*. All the three new dependencies include the same critical dimensions one for position associated with a translation with respect to the X DOF, and two for orientation associated the critical rotation axes aligned with the Y and Z axes.

To reduce the cluttering of the images presented, the only critical dimensions shown are those for translation, the critical dimensions for rotation can be obtained supported by Figure 2.15.

Step 5: In this step, *block1* is detached from *block4* with a simple translation in the X direction. The assembly operation does not require of any type of sensing when ignoring that *block1* should first be grasped by the robot arm, as in fact it was. In a real situation the robot's actions without grasped objects should be considered. For these *transit* tasks, the robot hand (*gripper*) should be considered as the manipulated object, and in case of a two-fingered gripper hand as the robot used for in the experimental setup, every grasp task should be classified as an insertion task requiring of using vision and force sensing. But, in this example, this step does not include any new dependency; it only include a new node to the ICdg for *block1*.

Step 6: In this last step of the plan, *block5* is inserted between *block2* and *block3*. Again, the analysis for this step arrive to similar conclusions as those of the fourth step, but with *block5* as the manipulated object, and making *block3* to depend on the configuration of *block2* which was assembled before than it.

The most important addition of this step, as can be observed in the ICdg, is the discovery of the indirect dependency of *block2* on *block1*. This dependency, as expected, is with respect to the configuration of *block1* before than it was manipulated by the second time. Without visually verifying the right X distance between *block1* and *block2*, and its relative orientations, *block3* could not be configured to succeed in the two insertion tasks, described in steps 4 and 6. This is an example of what was called a propagation of shared constraints.

Finally, the system presents a synthesis of the sensing analysis results as a modified assembly plan, which describe the identified transition of contact states, the assembly skill primitives implied by the task, the force compliance skills to use and the visual sensing requirements. Since every assembly step performs some kind of movement, an assembly skill primitive is given for each translational DOF. The force compliance skills and visual requirements are specified with respect to the DOF that require feedback information, the critical dimensions for force and visual sensing.

Figure 3.16 present the modified assembly plan resulting from the sensing analysis over the nominal assembly plan of the first experimental case. As can be noted, since only move and insert assembly skill primitives are required by the tasks, force sensing is only required in the insertion tasks performed during assembly step 4 to insert *block4* between *block1* and *block3*, and during assembly step 6 to insert *block5* between *block2* and *block3*. Vision should be also used in this steps. And as depicted in the final ICdg, vision also must be used in the second step to verify the position and orientation of *block2* with respect to *block1*, and in the third step to verify the position and orientation of *block3* with respect to *block1* and/or *block2*. In both cases the critical dimension is associated with the X DOF for translation. The critical dimensions for rotation are

```
ASSEMBLY STEP: 1
ASSEMBLY PART: block1
Contact State Transition: S->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz

ASSEMBLY STEP: 2
ASSEMBLY PART: block2
Contact State Transition: S->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz
VISION SENSING:
  Alignment(block1: Tx)

ASSEMBLY STEP: 3
ASSEMBLY PART: block3
Contact State Transition: S->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz
VISION SENSING:
  Alignment(block1: Tx, block2: Tx)
```

```
ASSEMBLY STEP: 4
ASSEMBLY PART: block4
Contact State Transition: S->B
ASSEMBLY SKILL PRIMITIVES
  Insertion Skill: Tx
  Move Skill: Ty,Tz
FORCE SENSING:
  Tx(block1,block3): dc-skill,kc-skill,cc-skill
VISION SENSING:
  Insertion(block1,block3: Tx)

ASSEMBLY STEP: 5
ASSEMBLY PART: block1
Contact State Transition: A->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz

ASSEMBLY STEP: 6
ASSEMBLY PART: block5
Contact State Transition: S->B
ASSEMBLY SKILL PRIMITIVES
  Insertion Skill: Tx
  Move Skill: Ty,Tz
FORCE SENSING:
  Tx(block2,block3): dc-skill,kc-skill,cc-skill
VISION SENSING:
  Insertion(block2,block3: Tx)
```

Figure 3.16: assembly plan with sensing operations for test case 1

also determined, but they do not apply in the assembly plan because, for this examples, they are not included in the ICdg, either.

3.5.2 Experimental Case 2

The second test case includes three types of polyhedral objects; *body1*, *body2*, and *body3*. All of them are kind of rectangular blocks of different sizes. The model of each type of object is described in a VANTAGE file, presented in Figure 3.17, where it can be noted that each type of object is created with a single CSG command that uses the CUBE primitive.

The nominal assembly plan is composed by nine assembly steps performed over six different assembly parts. Assembly parts *block1*, *block2*, and *block3* are polyhedral of type *body1*; assembly parts *block4* and *block5* are polyhedrons of type *body2*; and assembly part *block6* is of type *body3*.

The sequence of assembly steps for this case is presented in Figure 3.18. It presents a type of puzzle task where five of the six objects are assembled in a single step. The other four steps are applied to *block1*.

```
// BODY1
(csgnode body1 cube (30 30 40) (0 0 0 0 0 0))

// BODY2
(csgnode body2 cube (30 110 40) (0 0 0 0 0 0))

// BODY3
(csgnode body3 cube (90 40 40) (0 0 0 0 0 0))
```

Figure 3.17: VANTAGE models of parts for test case 2

```
// Load the models of assembly parts

load-models: case2.model.van

// Sequence of assembly

create-assembled-object: block1 body1 (-30 -30 0 0 0 0)
create-assembled-object: block2 body1 (-30 15 0 0 0 0)
create-assembled-object: block3 body1 (30 0 0 0 0 0)
create-assembled-object: block4 body2 (-60 10 0 0 0 0)
create-assembled-object: block5 body2 (60 10 0 0 0 0)
move-object: block1 (-30 -30 90 0 0 0)
move-object: block1 (-30 15 90 0 0 0)
move-object: block1 (30 0 90 0 0 0)
create-assembled-object: block6 body3 (0 65 0 0 0 0)
```

Figure 3.18: Nominal assembly plan for test case 2

Figure 3.19 graphically illustrate the nine assembly steps. As can be seen, the objects are not form a line, as the plan of case 1, however, as depicted in its final ICdg depicted in Figure 3.20, all the critical dimension of the tasks are associated with the X DOF. The execution of the plan require of the move, make-contact, and insert assembly skill primitives. It should be noted that when a pre-assembled object is moved, it appears two times in the figure that presents the assembly step; a soft gray level image showing where it was before the task and a black image in its actual position as a manipulated object.

Description and commentaries as the sensing analysis analysis advance in each assembly step of the assembly plan for this case are:

Step 1: *block1* is moved in free space to its initial assembly position. There is not need of any kind of sensing to perform this task.

Step 2: *block2* is moved in free space to the vicinity of *block1*, but not touching it. This block is aligned with *block1*, just displaced in Y direction. Due to the absence of

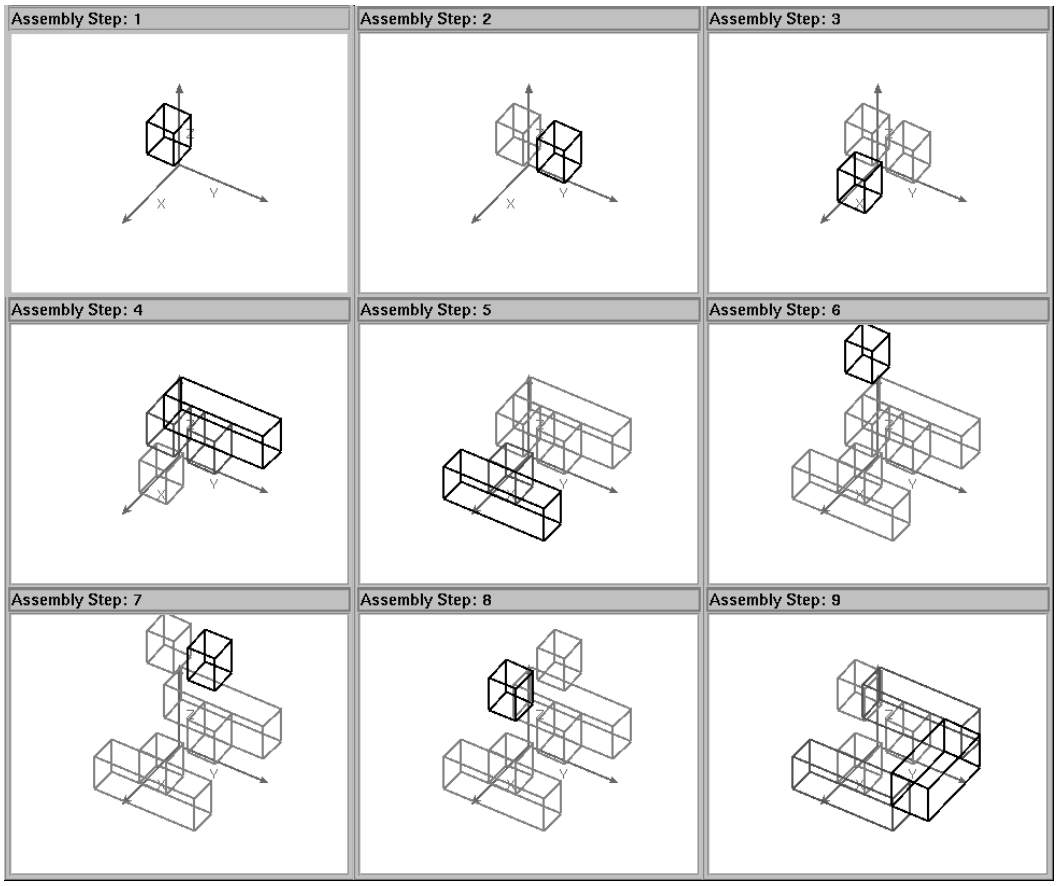


Figure 3.19: Sequence of assembly steps for test case 2

contacts, the system ignores the future significance of the X alignment of the *block1* with *block2*.

Step 3: *block3* is moved in free space to its final configuration. Again, the absence of contacts makes the system ignore the real importance of *block3* configuration.

Step 4: *block4* is moved in free space until if enters in contact with *block1* and *block2*. This task require to use the make-contact assembly skill primitive in the X direction, a guarded motion that could need of force control to correct the contact configuration. The critical dimension for translation is associated with the X DOF, while the critical dimensions for rotation are described by the Y and Z rotation axes.

This task also indirectly joins *block1* with *block2*, making significant its requirement for positional alignment with respect to X DOF, and its rotation alignment with respect to Y and Z axes. These represent critical dimensions that have to be observed when the *block2* is manipulated in the assembly step 2. This is illustrated by the dashed arrow that joins *block2* with *block1* in the final ICdg.

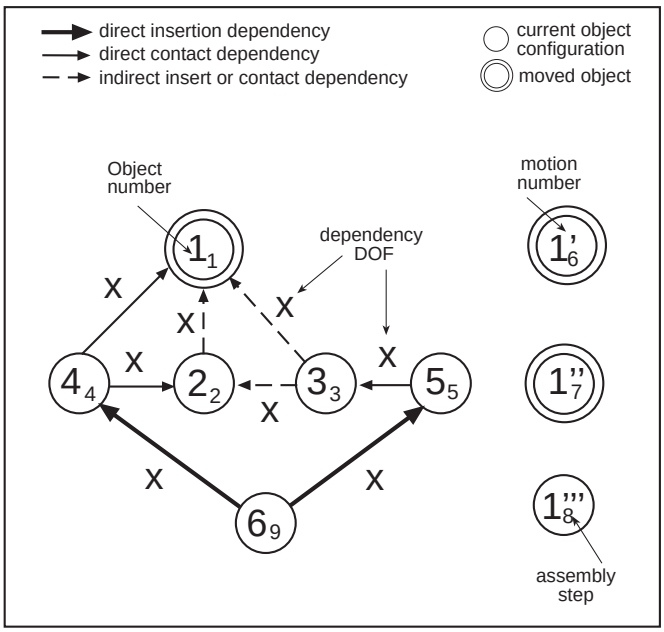


Figure 3.20: ICdg for test case 2

Step 5: *block5* is moved to contact with *block3*. The contact direction is also aligned with the X axis. This task requires of the make-contact assembly skill primitive that use force compliance skills to detect the contact and correct the contact configuration. The critical dimensions to control by force are the same as those of the previous step.

Steps 6, 7, and 8: In these steps, the *block1* is moved to different positions, but since *block1* does not participate in new contacts, and inclusive, it breaks its previous contact with *block4* in step 6, there is not need to add new force or visual sensing requirements.

Step 9: *block6* is inserted between *block4* and *block5*. Since all insertion task must be observed, this task requires of using vision to verify the correct alignment of *block6* with respect to *block4* and *block5*. And since the constrained motion due to the insertion is aligned with X axis, the X DOF represents the critical dimension for translation and the critical rotation axes are aligned with the Y and Z axes.

The most important result of the method in this task is realizing that although to succeed in the insertion *block4* and *block5* has to be aligned and positioned correctly, what has to be observed is the position and orientation of *block3* with respect to *block1* and/or *block2*. The evident reason is that the configurations of *block1* and *block2* completely fix the configuration of *block4* with respect to the critical dimensions, and that *block3* fixes the configuration of *block5*. And since the contacts are parallel to the alignment vector of the tasks in both cases, as

described for the second case of the translation analysis and for the third case of the rotation analysis, there is not need to observe *block4* nor *block5*.

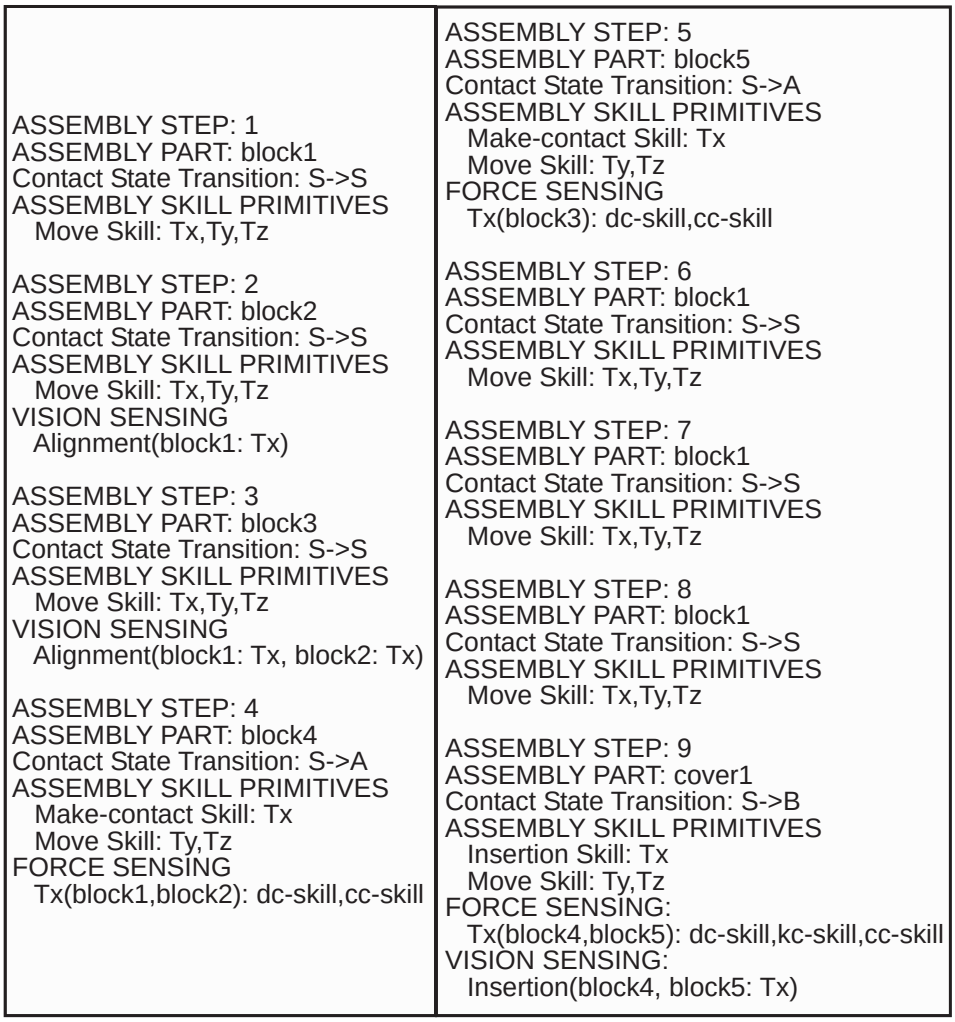


Figure 3.21: Assembly plan with sensing operations for test case 2

Figure 3.21 synthesize all the conclusions, graphically illustrated in the ICdg, of the step-by-step analysis of the assembly sequence, showing that only three steps require of visual sensing (steps 2, 3, and 9), and three steps require of force sensing (steps 4, 5, and 9). Steps 2 and 3 require of preventive vision to verify indirect alignment constraints, while step 9 requires vision for insertion. Steps 4 and 5 requires of force control realize guarded motions, while step 9 requires force control to complete an insertion operation.

3.5.3 Experimental Case 3

This third and last test case presents a more realistic assembly task that is performed with more complex parts. It includes four types of polyhedral objects. The CSG model of the objects is constructed with VANTAGE commands that use the CUBE primitive and binary operations for union and difference. Figure 3.22 presents the VANTAGE file that includes the commands to create the CSG model of the polyhedral objects.

```
// BODY
(csgnode a1 cube (170 20 20))
(csgnode a2 cube (15 15 20) (0 0 10 0 0 90))
(csgnode a3 cube (8 8 20) (-75 0 0 0 0 0))
(csgnode a4 cube (8 8 20) (75 0 0 0 0 0))
(csgnode a5 difference (a1 a2))
(csgnode a6 difference (a5 a3))
(csgnode body difference (a6 a4))

// COVER
(csgnode b1 cube (170 20 10) (0 0 -2.5 0 0 0))
(csgnode b2 cube (30 20 20) (0 0 2.5 0 0 0))
(csgnode b3 cube (15 15 20) (0 0 -7.5 0 0 90))
(csgnode b4 cube (8 8 10) (-75 0 -2.5 0 0 0))
(csgnode b5 cube (8 8 10) (75 0 -2.5 0 0 0))
(csgnode b6 union (b1 b2))
(csgnode b7 difference (b6 b3))
(csgnode b8 difference (b7 b4))
(csgnode cover difference (b8 b5))

// BAR
(csgnode bar cube (170 15 15) (0 0 0 0 0 0))

// BOLT
(csgnode d1 cube (15 15 10) (0 0 0 0 0 0))
(csgnode d2 cube (8 8 25) (0 0 -12.5 0 0 0))
(csgnode bolt union (d1 d2))
```

Figure 3.22: VANTAGE models of parts for test case 3

The first type of object, called *body*, is a rectangular block that includes two holes to its sides for inserting bolts, and a slot in the middle for inserting a bar. The body of *body* is constructed with the CSG CUBE primitive. The holes and the slot in the body are gotten through difference operations with respect to other blocks that describe the desired form and size.

The second and more complex type of object, called a *cover*, is a rectangular block with a rectangular bump and an slot in the middle, and two holes that go through the full body of the *cover* to both of its sides. The body and the form and size of the bump, of the slot, and of the holes are created with the CSG CUBE primitive. To construct the *cover* its body is united to the bump with a union binary operation, and the slot and

holes are formed as difference with the blocks that describe their form and size.

```
// Load the models of assembly parts

load-models: case3.model.van

// Sequence of assembly

create-assembled-object: body1 body (0 50 0 0 0 0)
create-assembled-object: body2 body (0 -50 0 0 0 0)
create-assembled-object: bar1 bar (0 0 50 90 0 0)
move-object: bar1 (0 0 14 90 0 0)
move-object: bar1 (0 0 10 90 0 0)
create-assembled-object: cover1 cover (0 50 50 0 0 0)
move-object: cover1 (0 50 21.5 0 0 0)
move-object: cover1 (0 50 17.5 0 0 0)
create-assembled-object: bolt1 bolt (-75 50 50 0 0 0)
move-object: bolt1 (-75 50 40 0 0 0)
move-object: bolt1 (-75 50 30 0 0 0)
move-object: bolt1 (-75 50 25 0 0 0)
```

Figure 3.23: Nominal assembly plan for test case 3

The third object, called *bar*, is a simple rectangular block created with a CSG CUBE primitive.

The fourth object, called *bolt*, is constructed as the union of two rectangular blocks created with the CSG CUBE primitive; the first block describes its elongated body, and the second its head. The nominal assembly plan , presented in Figure 3.23, describe a sequence of twelve steps that include two bodies (*body1* and *body2*), one bar (*bar1*), one cover (*cover1*), and one bolt (*bolt1*). *body1* and *body2* are assembled in single assembly operations; next, *bar1* is assembled in three steps; next, *cover1* is also assembled in three steps; and finally, *bolt1* is assembled in four steps.

Every assembly step is graphically illustrated in Figure 3.24.

Description and commentaries as the sensing analysis analysis advance in each assembly step of the assembly plan for this case are:

Step 1: *body1* is moved in free space to its final configuration. It does not require of any type of sensing operation.

Step 2: *body2* is moved in free space to its final configuration. The relevance of its alignment with *body1* is still not evident since it still does not participate in any contact relation.

Step 3: *bar1* approaches to the bodies in preparation for the insertion operation of the next step. Since this step is also in free space it does not include sensing requirements.

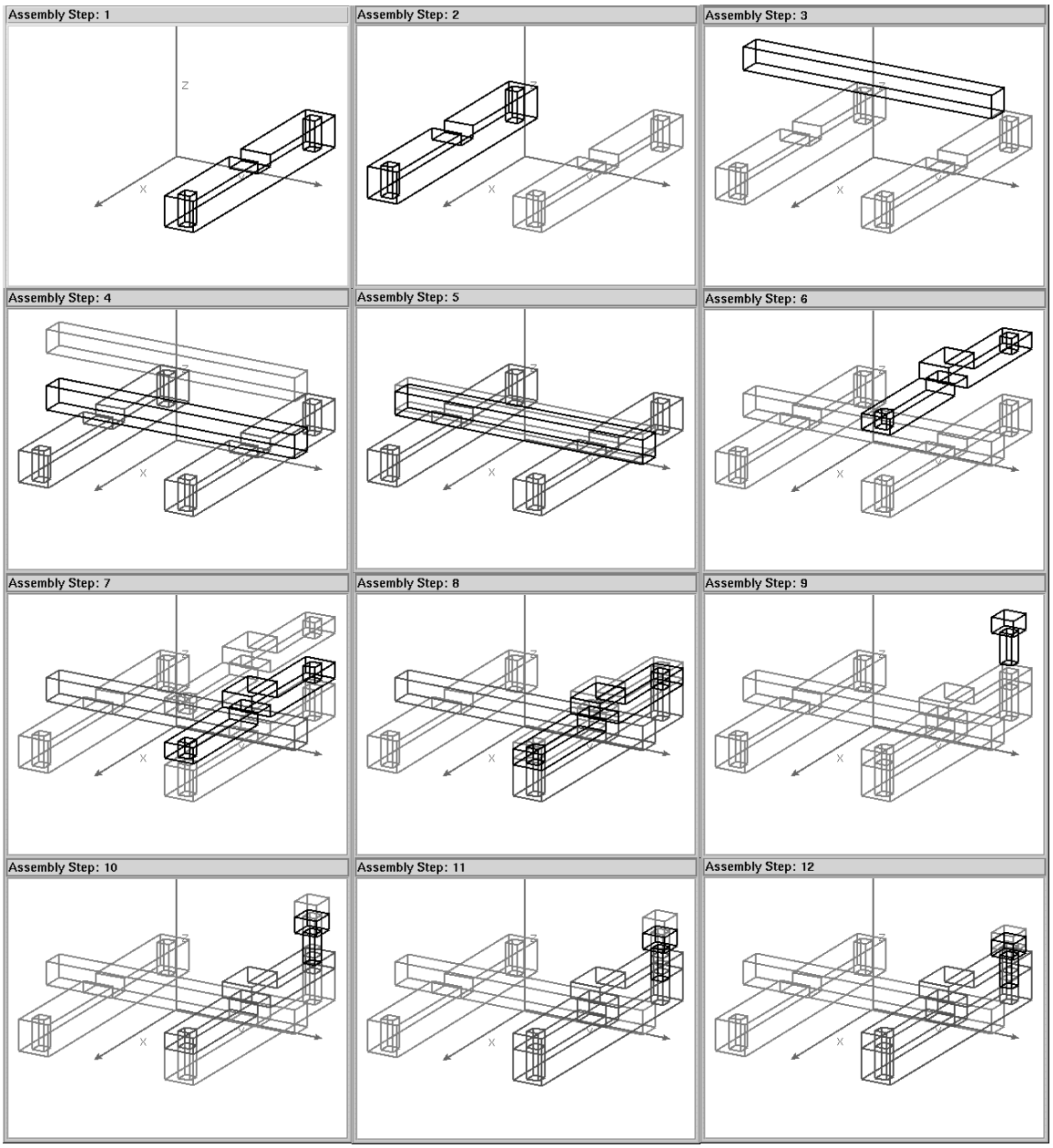


Figure 3.24: Sequence of assembly steps for test case 3

Step 4: *bar1* is inserted in the slots of *body1* and *body2*. As all insertion tasks, this step requires of visual verification with respect to the constrained DOF that in this case is aligned with the X axis. Also the orientation of the manipulated object has to be visually verified with respect to the Y and Z critical rotation axes. Force is also required to control and complete the insertion.

This step evidences the importance of aligning *body2* with respect to *body1* such that their slots get also aligned. This visual requirement is discovered by a propagation of joining constraints, that recognizes the X DOF as a critical dimension for translations and Y and Z axes as critical rotation axes, for the assembly step 2.

Step 5: This step concludes the assembly of *bar1* letting it resting over the bottom of the slots of the bodies. This operation presents a typical task that require of the slide assembly skill primitive to control the slide operation against the walls of insertion slots. It also should to perform the guarded motion required by make-contact assembly skill primitive.

The task does not require of vision, only requires of force compliance with respect to the critical dimensions which are associated with X DOF in translation with Y and Z critical rotation axes for the slide operation, and the Z DOF in translation and X critical rotation axis for the make-contact operation. The Y critical rotation axis that would be required by a manipulated object completely unconstrained manipulated object is eliminated as established by the second case of the rotation analysis.

Step 6: *cover1* approaches *bar1* in preparation for the next insertion operation. This step is in free spaces, requires of using the move assembly skill primitives, but does not introduces new sensing requirements.

Step 7: *cover1* is moved so that *bar1* gets inserted in its slot. As all the insertion tasks, this task also requires of vision with respect to the alignment reference which is the X position. As illustrated in the final ICdg of Figure 3.25, the critical dimensions for the task are the X DOF for translation and Y and Z critical rotation axes.

Step 8: In this task, *cover1* is slided down the *bar1*'s walls to rest over *body1* top side. This task is another operation including an slide assembly skill primitive with respect to X DOF and a make-contact assembly skill primitive with respect to Z DOF. The main difference between this step and step 5 is that *cover1* slides over *bar1* and finishes resting in a different object, *body1*, which is depicted by three solid lines in the ICdg, instead of the two lines used for the step 5.

Step 9: *bolt1* is moved in free space to an approach position for the next insertion operation. This step does not require of sensing operations.

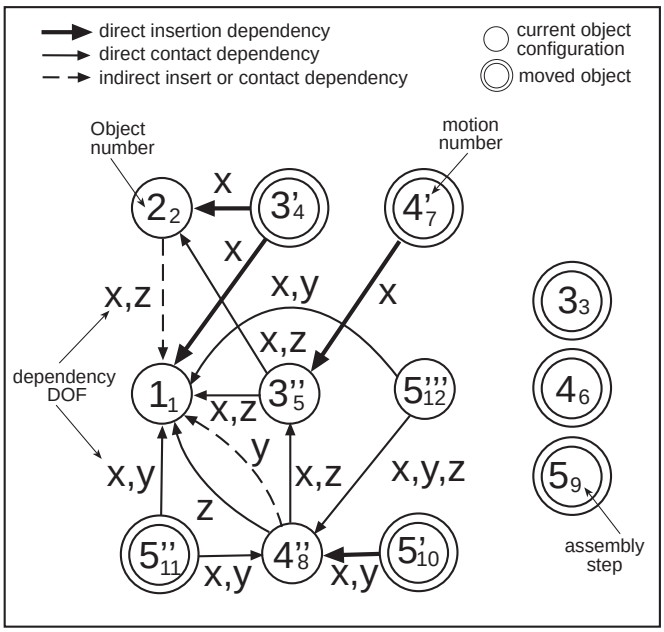


Figure 3.25: ICdg for test case 3

Step 10: The inferior part of *bolt1* is inserted in one of the holes of *cover1*. The difference of this insertion operation and all the other insertions is that it has two critical dimension for translation, associated with the X and Y DOF, and three critical rotation axes aligned with all the coordinate axes.

Step 11: *bolt1* is inserted in one hole of *body1*. This is an interesting task because as can be noted in the final ICdg, it is not considered as all the other insertion tasks. This singular insertion task does not require of using vision, it only requires of using force compliance, and this is because the insertion of *bolt1* is obviated by the alignment of *cover1* with *body1*. The hole in *body1* is implicitly considered as an extension of the hole of *cover1* because the system did not have to be programmed to deal specifically for this singular cases.

Step 12: In this last step of the assembly plan, *bolt* is slided into the holes of *cover1* and *body1* until the bottom of its head rest in the top side of the cover. This step requires only of force compliance to perform the slide and make contact tasks. The critical dimensions for translation are associated with X and Y of *body1* and X, Y, and Z DOF of *cover1*. The critical rotation axes are associated with all the coordinate axes.

Figure 3.26 synthesizes the sensing analysis results in a new assembly plan with sensing requirements. Four assembly steps (steps 1, 3, 6, and 9) do not require of any sensing; five steps (steps 2, 4, 7, 8, and 10) requires of visual operations, three of them (steps 4,

```
ASSEMBLY STEP: 1
ASSEMBLY PART: body1
Contact State Transition: S->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz

ASSEMBLY STEP: 2
ASSEMBLY PART: body2
Contact State Transition: S->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz
VISION SENSING
  Alignment(body1: Tx,Tz)

ASSEMBLY STEP: 3
ASSEMBLY PART: bar1
Contact State Transition: S->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz

ASSEMBLY STEP: 4
ASSEMBLY PART: bar1
Contact State Transition: S->B
ASSEMBLY SKILL PRIMITIVES
  Insertion Skill: Tx
  Move Skill: Ty,Tz
FORCE SENSING
  Tx(body1,body2): dc-skill,kc-skill,cc-skill
VISION SENSING
  Insertion(body1, body2: Tx)

ASSEMBLY STEP: 5
ASSEMBLY PART: bar1
Contact State Transition: B->D
ASSEMBLY SKILL PRIMITIVES
  Make-contact Skill: Tz
  Move Skill: Ty
  Slide Skill: Tx
FORCE SENSING
  Tx(body1,body2): kc-skill
  Tz(body1,body2): dc-skill,cc-skill

ASSEMBLY STEP: 6
ASSEMBLY PART: cover1
Contact State Transition: S->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz

ASSEMBLY STEP: 7
ASSEMBLY PART: cover1
Contact State Transition: S->B
ASSEMBLY SKILL PRIMITIVES
  Insertion Skill: Tx
  Move Skill: Ty,Tz
FORCE SENSING:
  Tx(bar1): dc-skill,kc-skill,cc-skill
VISION SENSING:
  Insertion(bar1: Tx)

ASSEMBLY STEP: 8
ASSEMBLY PART: cover1
Contact State Transition: B->D
ASSEMBLY SKILL PRIMITIVES
  Make-contact Skill: Tz
  Move Skill: Ty
  Slide Skill: Tx
FORCE SENSING
  Tx(bar1): kc-skill
  Tz(body1,bar1): dc-skill,cc-skill
VISION SENSING
  Alignment(body1: Ty)

ASSEMBLY STEP: 9
ASSEMBLY PART: bolt1
Contact State Transition: S->S
ASSEMBLY SKILL PRIMITIVES
  Move Skill: Tx,Ty,Tz

ASSEMBLY STEP: 10
ASSEMBLY PART: bolt1
Contact State Transition: S->E
ASSEMBLY SKILL PRIMITIVES
  Insertion Skill: Tx,Ty
  Move Skill: Tz
FORCE SENSING:
  Tx(cover1): dc-skill,kc-skill,cc-skill
  Ty(cover1): dc-skill,kc-skill,cc-skill
VISION SENSING:
  Insertion(cover1: Tx,Ty)

ASSEMBLY STEP: 11
ASSEMBLY PART: bolt1
Contact State Transition: E->E
ASSEMBLY SKILL PRIMITIVES
  Slide Skill: Tx,Ty
  Move Skill: Tz
FORCE SENSING:
  Tx(body1,cover1): kc-skill
  Ty(body1,cover1): kc-skill

ASSEMBLY STEP: 12
ASSEMBLY PART: bolt1
Contact State Transition: E->H
ASSEMBLY SKILL PRIMITIVES
  Slide Skill: Tx,Ty
  Move Skill: Tz
FORCE SENSING:
  Tx(body1,cover1): kc-skill
  Ty(body1,cover1): kc-skill
  Tz(cover1): dc-skill,cc-skill
```

Figure 3.26: Assembly plan with sensing operations for test case 3

7, and 10) for insertion; and finally, seven steps require of force compliance, three for insertion (steps 4, 7, and 10), four for sliding (steps 5, 8, 11, and 12), and three for contacting (steps 5, 8, and 12).

The system correctly identified the features that describe the critical assembly relations that, as mentioned before, are related with the most complex operations and those where most commonly failure happens during the robotic execution of an assembly plan.

Chapter 4

Sensor Planning for Visual Verification

This chapter presents a method to determine the best *sensor configurations* to use for observing an assembly part before the execution of an assembly operation. The objects to observe are determined by the sensing planner described in Chapters 2 and 3. The sensor planning method evaluates a predefined set of *viewpoints* to define an order based on the ability of each sensor configuration to maximize a measure of success an assembly step.

Known as *sensor planning* in the context of computer vision [100], the problem is to develop strategies to automatically determine a set of sensing parameter values that will achieve a task with a certain degree of satisfaction from given information about the environment as well as information about the task that the vision system is to accomplish.

The goal of a sensor planning module is gathering the best information to decide confidently if it is convenient to continue executing the plan and if any adjusting action is required to increase the possibilities of success during the execution of an assembly plan. The expected outcome from this module, as part of a system that automatically generates sensing strategies for robotic assembly, can include the sensor locations, its settings, and even its planned motions (*active vision*) [5]. Additionally, in reconfigurable environments, the illumination system can be planned [92] [67].

The biggest challenge for a sensor planner in computer vision is to understand and quantify the relationship between objects to be viewed and the sensors observing them. In this relationship between the object and the viewer, one of the most important tasks is to determine the viewpoints, because it is the viewpoint the main responsible for the quality of the image which directly affects the accuracy of the vision task.

The sensing analysis module described in Chapters 2 and 3, identify a group of op-

erations, from a nominal assembly plan, that require the use of vision to determine the configuration of some assembly elements. These configurations will be used during the execution of the assembly plan to adapt the actions of the manipulator. In the structured and well-known environments where robotic assembly plans are executed, frequently, the sensor planner can ignore the object recognition and global localization tasks, and concentrate on the refinement of the expected poses of the assembly elements. After this, the refined poses of the objects can be used to reconfigure the scene, verify the contact configurations among the objects, and accordingly, perform preventive adjustments on the configurations of manipulated objects.

In general, there are an infinite number of possible configurations of a sensor for taking images that would be used to localize target objects in an assembly scene. However, in this dissertation the potential viewpoints are limited to a finite set of viewpoints positioned on the surface of a discretized viewing sphere. This spherical representation is both simple and efficient to model the sensor configurations. A geodesic tessellation was chosen to take advantage of its uniform distribution of points. The potential viewpoints are localized in the center of each tessel and oriented through the center of the viewing sphere. An object to be observed is localized in the center of the sphere (see Figure 4.1).

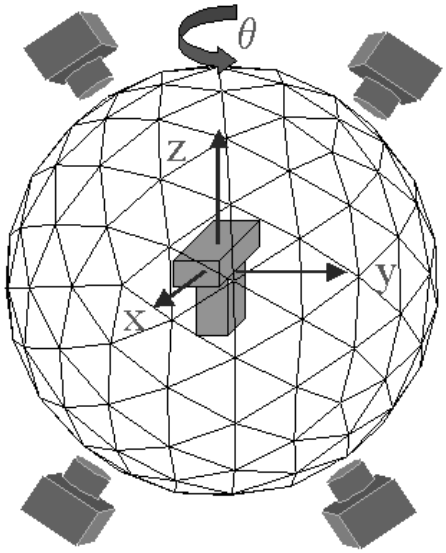


Figure 4.1: Spherical representation used to model the sensor configurations.

Since the sensing analyzer determine the visual sensing requirements from a geometric reasoning process fired by the intention to succeed in the execution of the assembly tasks, the purpose of the method described in this chapter is to determine an ordering of the finite set of viewpoints based in their ability to help in the reduction of critical errors in each assembly operation. This means that the ordering of viewpoints to see the same object can vary from task to task. The purpose of the method is not selecting the best viewpoint, because that is decided by another method that construct the complete visual

strategy for an assembly plan. Such method takes advantage of the results obtained by the sensor planner, but also consider other important factors such as external visual occlusion and sensor motion. The visual strategy planner is presented in Chapter 5.

Since uncertainty can not be completely eliminated from vision data, an strategy has been developed to quantify the expected uncertainty for every potential viewpoint.

4.1 Assembly Uncertainty

Uncertainty is ubiquitous during an assembly plan execution and its the main cause of its failure. Then, uncertainty is the main assembly factor to analyze, model, and reduce, for improving the opportunities to succeed in each assembly operation. Every assembly task has an explicit or implicit tolerance to error that depends, mainly, on the type of task and task geometry involved. Then certain quantity of uncertainty can be tolerated.

The uncertainty has an additive nature during the assembly plan execution. Thus, a natural strategy for dealing with it is: quantify the tolerance of the assembly operations and their accumulated uncertainty; if the accumulated uncertainty is bigger than the tolerance of a current task, use a sensing feedback operation to reduce the uncertainty. The sensing operation will produce information that can be used, first to reduce the accumulated uncertainty, and second to detect deviations from the plan that require of a different action, possibly to correct the deviation, select an alternative sequence of tasks, or abort the assembly execution. Thus a new plan can be obtained by introducing sensing steps and conditional branches.

But, the sensory data includes its own amount of uncertainty which means that task uncertainty can not be completely eliminated. Task uncertainty as well as sensory uncertainty are parametric, in a sense that its amount differ with respect to different parameters. For this dissertation, the most relevant parameters are those associated with the critical dimensions of a task. Then, when determining the configuration of a sensor for a sensing operation, the best viewpoints are those that allow a sensor to get information for the parameters associated with the critical dimensions, that include less potential error (uncertainty) than is tolerated. If this is not possible for some, or inclusively all, the critical dimensions, the best viewpoints are those that minimize the exceeding sensory uncertainty over task tolerance on *critical parameters* – parameters associated with the critical dimensions of the task.

4.1.1 Uncertainty in Robotic Assembly

To deal with the problem of uncertainty in the robotic execution of assembly plans, when there are not sensors to get feedback information, the robot and its environment has to be engineered so that everything will work. Tasks where uncertainty is an intrinsic property (e.g. pick-and-place) have to include additional and specific implements to eliminate it or at least reduce it to a range within the planned tolerances (e.g. using fixtures).

The cost of surrounding the robot with hard automation devices to eliminate the uncertainty factor, seriously restricts its flexibility and ability to adapt to changes and then also reduces its industrial applicability. Moreover, the hard automation devices have to be planned [17], designed and added as a new element in the assembly planning process, which is also an important problem [21].

An alternative to the elimination of uncertainty by including additional assembly accessories is to manipulate and integrate the sources of uncertainty into the assembly planning process. This means that uncertainty will persist as a planning factor and deviations from the plan will occur. The type of task and magnitude of the deviation will determine its consequences. Then, there are two main approaches to deal with it: one, uncertainty can be handled using so-called *robust methods*, or two, uncertainty can be controlled using sensors. A robust method is one that assumes a bounded amount of uncertainty in each aspect of a problem, but does not assign probabilities to values within the allowed interval [26]. A robust solution works no matter what the actual error value is, provided it is within the assumed interval.

This dissertation follows the second alternative. It uses sensors to control the uncertainty with respect to the critical dimensions of the task. How the sensors and uncertainty will be represented and used in the programming system will depend on the mechanism selected for controlling uncertainty by using the sensors in the final program.

4.1.2 Sources of Uncertainty and Errors

There are many sources of uncertainty, however, the most relevant in the case of assembly plan execution are:

- Uncertainty in the assembly device (robot arm) and its operations.
- Discrepancies between the physical objects and their geometric representations.
- Uncertainty in the sensory information obtained from the selected sensors.

Robot and Task Uncertainty

Robots as mechanical devices can perform their actions with limited precision, depending in the type of robot, type of motion, servo mechanisms, differences in operating temperature and weight, and type of applied control mechanisms. As a result, random errors – fluctuations around one value – and systematic errors – error that has constant sign and size, are involved. Good calibration techniques are needed to filter out the systematic errors.

A robot command is transformed to multiple local commands for each of its component parts. The composed discrete precision of each part will determine the global uncertainty of the robot action, if it is not applied a compensation mechanism, the uncertainty will present an additive behavior that will determine partially the resulting deviation in a commanded action as part of the assembly plan.

Additionally, the resulting uncertainty from the robot actions depend on the task uncertainty. The task uncertainty is determined by the complexity of the task and the discrepancies among the expected configurations of the assembly elements and the actual configurations when the task is performed. It is evident that some tasks are more difficult to perform for the robot, e.g. moving to place is easier than grasping an object. It has to be clear too that if the expected conditions for the task are not met, the result of its execution could differ, e.g. grasping an object that is not where it is supposed to.

When a difference between the expected configuration and an actual configuration of the assembly parts is critical, s.t. it can easily cause the failure of the task, a mechanism to control such deviations is recommended, e.g. using feeders, fixtures, etc. or sensing and potentially correcting the deviations. The second alternative was chosen in this study.

If a sensing mechanism is selected to detect configuration differences, its uncertainty will be determined by the sensing strategy. The robot action uncertainty should be added to the sensing uncertainty before computing the actual success probability of the task.

Object Representation Uncertainty

Inclusive if the assembly plan were developed using real objects, the implicit variability in the fabrication process of the assembly parts could include discrepancies between the geometric features of the reference object for planning and the actual assembled parts. There are several approaches for dealing with uncertainty in the geometry of objects in the assembly context [24][50][12].

Such variability in the shape dimensions of the assembly elements affects the uncertainty of the tasks. It can be considered as a deviation from the expected configurations for

the task, but in this case, such deviation can not be corrected.

The model uncertainty directly affects the success opportunity for the task. This opportunity is originally determined by the geometric tolerances of the task, which are computed from the geometry of the objects. Thus, a change in the geometry of the objects immediately is reflected in the success of the task. The impossibility to correct the model error of the assembled parts will in some cases make impossible to perform the task, e.g. a reduced dimension of a hole in a peg-in-hole assembly task. In such a case, failure is guaranteed and it is recommended to be detected to avoid useless correction actions. In other cases, it can improve the probabilities of success by increasing the tolerances of the task, e.g. a bigger hole for an insertion task.

Failure in assembly plan execution can be determined in different ways. A common way is trying to perform the task a limited number of times and after failing to succeed, failure is declared and the task is aborted. If the success is quantified and represented as a region, failure can be determined using sensory information when the sensed deviation minus the expected uncertainty of the sensing strategy still exceeds the full tolerance for the task in a critical parameter.

Sensory Uncertainty

Sensory uncertainty comes from the sensors used and the strategy for using the obtained information to determine the descriptors of the observed elements that describe the state of the task. The sensor have a limited and approximated capacity to get specific world features. It is limited because it is affected by factors as the illumination, that reduce its detection capability, and approximated because it is affected by several factors (e.g. calibration, noise, etc.) that modify the precision of the feature descriptors that finally produce.

After the sensor features have been obtained, they are further converted into object features. This adds uncertainty to the final result of the sensory feedback operation by propagating the sensor uncertainty to the process of object feature description. The specific way to propagate the initial sensor uncertainty and the strategy to model and minimize its effect will depend on the technique used to compute the object descriptors.

The usefulness of the sensory information will depend on its accuracy. If the degree of uncertainty in the estimated parameters is less than the detected deviation from the plan, then it can be used to perform a recovery action and improve the opportunities of succeeding in the assembly task execution.

4.2 Evaluation of Visual Sensing Strategies

In the present work, vision is used to localize rigid objects in an assembly scene. The observed pose information is used to verify the fulfillment of some alignment constraints determined as critical in the successful execution of an assembly operation. To realize this verification it is necessary to know the poses of several objects. Therefore, a *visual sensing strategy* for this purpose has to determine a sequence of the best sensor configurations to perform the object localizations.

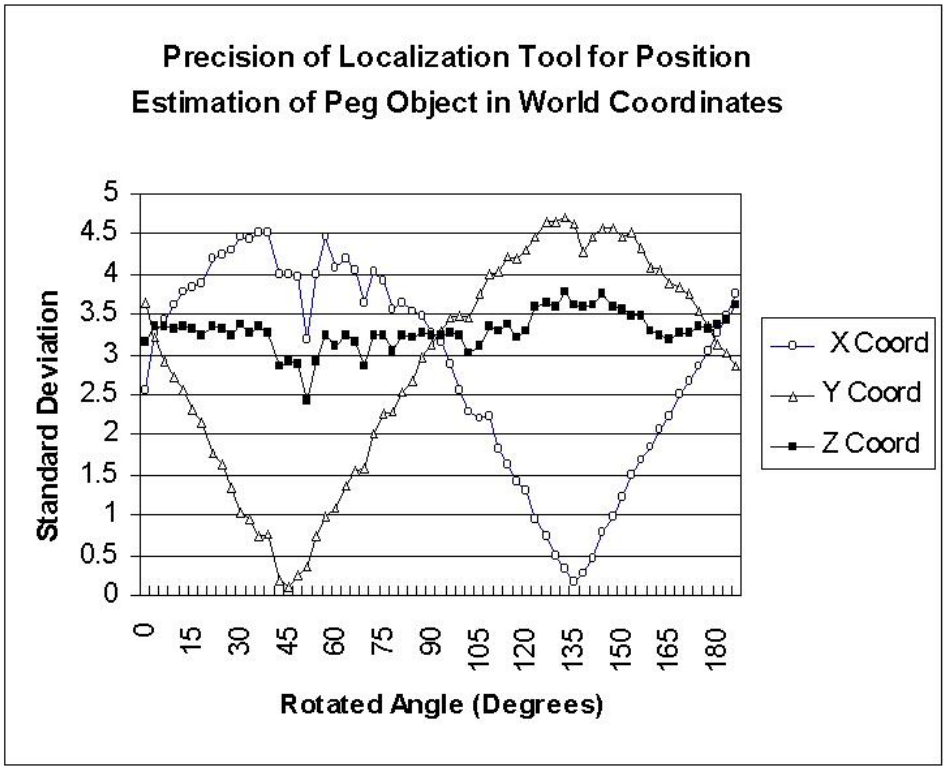


Figure 4.2: Precision of object localization tool for position estimation of a peg object.

In general the uncertainty in the results produced by a vision system depend upon several factors that make them better for observing some DOF than others. For example, Figure 4.2 depicts a graph that illustrates the precision of the object localization tool used in this dissertation when determining the position of a peg object and the camera is moved around the object, as illustrated in Figure 4.3. The graph clearly shows that the precision is different for each coordinate axis and that it changes when the viewpoint is modified.

One immediate application of the results of the preventive sensing analysis of Chapter 2 and 3 is in determining which DOF are critical for an assembly step and has to be observed in order to reduce its uncertainty. This information is used for deciding which viewpoint is the most convenient for localizing an object.

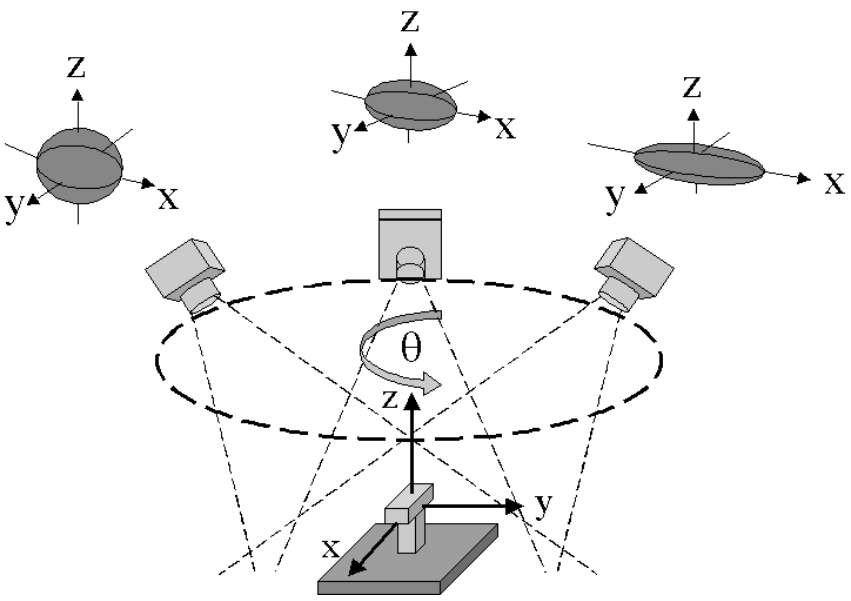


Figure 4.3: Distribution of viewpoints.

4.2.1 Evaluating Viewpoints

If the pose estimation of objects is accurate enough, adjustment actions can be proposed that will make the task succeed; if not, at least the error can be reduced making easier the job for corrective actions.

Since the goal of using vision is to succeed in the execution of an assembly task, the criteria to decide the best viewpoint to use for observing and localizing an object must tell something about its effect on the success of the task. Such success depends on the tolerance to errors in the execution of the assembly steps and the accuracy on the measures obtained from the visual localization tool.

Mating parts and assembly operations are usually devised including certain amount of clearance to support some margin of error during assembly. This tolerance to error is a dimensional phenomena, making the task better to support the error with respect to some dimensions than others. As mentioned above, errors in a task are commonly characterized by differences in the configuration and shape of the assembly elements, differences in the configuration of the environment, differences in the grasping configurations of the manipulated objects, and deviations from the planned trajectories of motion for the robot. All of them can finally be reduced to differences on expected poses of the assembly elements at determined periods of time. If the amount of error tolerated by a task is geometrically modeled for each of the pose parameters of such elements, a region of tolerated error can be obtained for every of these objects in each assembly operation. If the pose of every assembly element is kept inside of its region of tolerated error, the

assembly operation is expected to succeed.

Vision localization tools usually work on noisy, incomplete, and uncertain information, and obtain a pose estimation of an observed object that is just an approximation to the true pose. The error in the estimated pose of the observed object is another dimensional phenomena. It affects differently to different dimensions. Since the goal of object localization is to determine the pose of an object, then the approximation will be reflected by different amounts of error with respect to each pose parameter. Moreover, the magnitude of the error in each pose parameter will change even when the localization tool is used to localize the same object in images obtained with the sensor from the same viewpoint. The error do not depends only on the viewpoint. If the object localization's error with respect to each pose parameter is geometrically modeled, a region of uncertainty for the localization tool can be obtained for the sensor in each of the considered viewpoints. If a deviation from a planned pose is bigger than the error in the pose estimation of the sensor, a correction could be applied to effectively reduce it. If not, at least can be used to eliminate the error addition effect.

If the sensing uncertainty of the sensor and the error tolerance of the task are represented and quantified in the same parametric space, e.g. configuration space, a success probability for an assembly operation can be predicted for each possible sensor configuration. These probabilities can then be used to define an order of preference on the viewpoints considered in a vision operation for an assembly task.

4.2.2 Quantifying success

The underlying idea of configuration space in Robot Motion Planning is to represent the robot as a point in an appropriate space – the robot's configuration space – and to map the obstacles in this space [58]. The mapping transforms the Robot Motion Planning in a problem of finding a line in free-space – space not occupied by an obstacle – that joins an initial point (associated with an initial robot pose) with a final point (associated with a target robot pose).

In this dissertation, the underlying idea for getting a measure of success, when performing visual verification of alignment constraints, is to represent a target object for localization as a point in a space of critical dimensions – a *critical configuration space* – and to map the tolerance to error of the task and the uncertainty of the sensor in this space. This mapping transforms the problem of computing a measure of success into the problem of computing the portion of a sensing uncertainty region that falls inside of an error tolerance region for the task (see Figure 4.4).

If the problem of computing a measure of success of a visual sensing strategy is simplified to the determination of a viewpoint to localize a target object in an accurately known

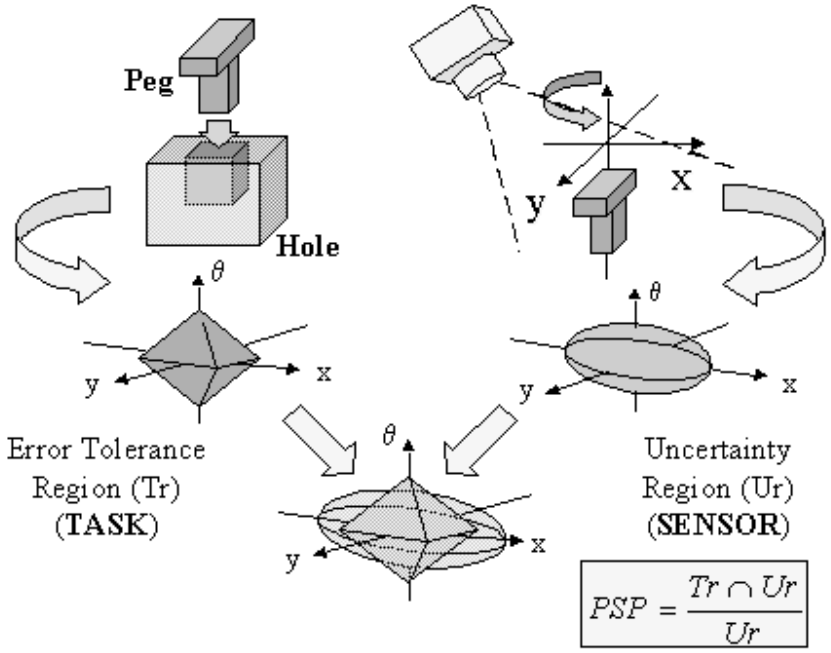


Figure 4.4: Predicted Success Probability.

environment, its computation can be formulated as follows:

Participant Objects. Let $\mathcal{M}$ be a target rigid object for localization – the *manipulated object* – in an Euclidean space $\mathcal{W}$, called the *world*, represented as $\mathbf{R}^N$, with $N = 2$ or 3. Let $\mathcal{E}_1, \ldots, \mathcal{E}_m$ be m fixed rigid objects distributed in $\mathcal{W}$ – the *environmental objects*. Assume that the locations of every $\mathcal{E}_i$ are accurately known. In addition, let $\mathcal{F}_\mathcal{M}$, $\mathcal{F}_{\mathcal{E}_1}, \ldots, \mathcal{F}_{\mathcal{E}_m}$, and $\mathcal{F}_\mathcal{W}$ be Cartesian frames embedded in $\mathcal{M}$, $\mathcal{E}_1, \ldots, \mathcal{E}_m$, and $\mathcal{W}$, respectively. Since the $\mathcal{M}$ and the $\mathcal{E}_i$'s, for all $i \in [1, m]$, are rigid by definition, every point in them has a fixed position with respect to their embedded Cartesian frames. But the position of an object's point in $\mathcal{W}$ depends on the position and orientation of its embedded frame relative to $\mathcal{F}_\mathcal{W}$.

Configuration Space. Since a specification of the position of every point in an arbitrary object $\mathcal{A}$ relative to a fixed reference frame is known as a *configuration* of $\mathcal{A}$, a specification of the position τ and orientation Θ of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_\mathcal{W}$ is a configuration $\mathbf{q}$ of $\mathcal{A}$. The *configuration space* of $\mathcal{A}$ is the space $\mathcal{C}_\mathcal{A}$ of all the configurations of $\mathcal{A}$. A point a on $\mathcal{A}$ at configuration $\mathbf{q}$ is denoted by $\mathcal{A}(\mathbf{q})$ in $\mathcal{W}$. A configuration $\mathbf{q}$ of $\mathcal{A}$ in $\mathbf{R}^N$ can be described as a list of d independent parameters, with $d = 3$ (if $N = 2$) and $d = 6$ (if $N = 3$), which corresponds to representing $\mathcal{C}_\mathcal{A}$ as $\mathbf{R}^d$.

Critical Configuration Space. In Chapter 3 was concluded that it is convenient to introduce a Cartesian frame $\mathcal{F}_t$, aligned in accordance with the task at hand, to

determine the critical dimensions for a task that requires of using vision. The critical dimensions are identified as a subset of the d independent parameters associated with $\mathcal{F}_t$. In a similar fashion to that used to describe the configuration space of $\mathcal{A}$, a critical configuration $\mathbf{q}_\mathcal{C}$ of $\mathcal{A}$ is a specification of the position and orientation of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_t$. This specification only includes the parameters associated with the critical dimensions of the task. The critical configuration space of $\mathcal{A}$ is the space $\mathcal{C}_\mathcal{A}^C$ of all the critical configurations of $\mathcal{A}$.

Task Tolerance Region. The tolerance of a task t to errors in a planned pose of an object $\mathcal{A}$ is mapped to $\mathcal{C}_\mathcal{A}^C$ as a region $\mathcal{T}_\mathcal{A}(t)$ that includes the critical configurations of $\mathcal{A}$ that guarantee the success of the task execution.

Sensor Uncertainty Region. The variability in the values of the pose parameters of an object $\mathcal{A}$, obtained by using the localization tool on images captured with the optical sensor in a particular viewpoint $\mathbf{v}$ is mapped to $\mathcal{C}_\mathcal{A}^C$ as a region $\mathcal{U}_\mathcal{A}(\mathbf{v})$ that includes the critical configurations in which $\mathcal{A}$ is observed from $\mathbf{v}$, when it really is at the origin of $\mathcal{C}_\mathcal{A}^C$.

Predicted Success Probability. A predicted success probability Psp for task t, after the pose of the manipulated object $\mathcal{M}$ is adjusted, when $\mathcal{M}$ was localized with the sensor in the viewpoint $\mathbf{v}$, is computed as

$$Psp_\mathcal{M}(t, \mathbf{v}) = \frac{\|\mathcal{T}_\mathcal{M}(t) \cap adjust(\mathcal{U}_\mathcal{M}(\mathbf{v}))\|}{\|\mathcal{U}_\mathcal{M}(\mathbf{v})\|} \tag{4.1}$$

$\cap$ is a region intersection operator and $\|\ldots\|$ is an operator that quantifies the size of the subspace occupied by a region (length in one dimension, area in two dimensions, volume in three dimensions, and so forth). $adjust(\ldots)$ is a function that maps a region of critical configurations into another resulting from correcting the pose of an object $\mathcal{A}$ assumed to be localized in each critical configuration $\mathbf{q}_\mathcal{C}$ of its $\mathcal{U}_\mathcal{A}$.

As described in Chapter 3, visual verification of assembly conditions is recommended for three cases:

1. To determine a possible preventive adjustment in the pose of the manipulated object for an insertion operation.

2. To determine a possible preventive adjustment in the pose of a manipulated object that participates, directly or indirectly, in the success of a future insertion operation, as part of the environment.

3. To determine a possible preventive adjustment in the pose of a manipulated object that participates, directly or indirectly, in the success of a future make-contact operation, as part of the environment.

Usually, an assembly operation that requires of visual verification will include inquiries about the poses of several objects: the manipulated object and some environmental objects. In the present work, a visual sensing strategy for an assembly operation is composed by a set of viewpoints to use for localizing each object participating in a visual verification task. In this way, the selection of a particular visual sensing strategy will define the objects to observe and the viewpoints to utilize for their localization.

4.3 Modeling and Quantifying Task Tolerance To Errors

In order to compute Psp values for the viewpoints on a viewing sphere of an object that has to be localized before the execution of an assembly step, a tolerated error in its pose estimation has to be represented and quantified.

As described above, a task tolerance region $\mathcal{T}$ is a representation of the allowed pose error of an assembly element during the execution of a robotic manipulation. This region is described with respect to some critical dimensions determined during the sensing analysis stage. A $\mathcal{T}$ can be calculated from the CAD models of the objects participating in a dependency relation among objects that was identified as requiring the use of visual verification.

Uncertainty and errors during the execution of an assembly operation will produce a difference between the actual pose and the planned pose of an object. Such difference from the planned pose could be tolerated by the assembly operation or not. A *tolerated object-pose error* is a distance in which an object can be displaced in a particular direction and the angle that it can be rotated with respect to a particular axis of rotation without causing the failure of an assembly operation. The object-pose error is traduced as local displacements in the topological features of the object. Success is guaranteed if the displacement of the features caused by uncertainty falls inside its tolerance region.

A $\mathcal{T}$ describes a subspace of configurations for a manipulated object. An estimation of its shape can be obtained from the geometric tolerances of the task. This $\mathcal{T}$ can be described as a system of inequalities that describe its sub-space of tolerated configurations.

The inequalities are parametric descriptions of allowed object motions. Every combination of features participating in a contact or insertion relation define a possible *redundant constraint inequality*. For example, figure 4.4 depicts the shape of this region for the case

of inserting a peg with square cross-section into a square hole; if the critical dimensions are reduced to one for rotation, which is aligned with the insertion direction, and two for translation, which are orthogonal to the insertion direction. Two of the three rotational degrees of freedom are ignored even when they would be critical in the general case. The depicted region is described by eight non-redundant inequalities, two for each vertex of the cross section of the peg [64].

An assembly state is described by a set of relations among topological features of the mating elements. Since the type of relations considered in this thesis are surface-to-surface relations, a tolerance represents a distance between a constrained surface and a constraining surface. If this distance is exceeded, features of the manipulated object will seem to penetrate into features of the environment.

4.3.1 Quantifying Error Tolerance in Critical Dimensions

In the description of the sensing analysis method of chapters 2 and 3 was mentioned that information in the ICdg will be used to construct a criteria for planning the sensor configuration, but it was not explained how. This section complements that affirmation and describe some additional information associated with the dependency arcs of an ICdg.

Each arc in an ICdg define critical alignment dependencies between two objects. This dependencies represented by reference vectors would finally represent the critical dimensions of the manipulation task for the dependent objects. When this dependencies are deduced, some additional information is also collected for helping in the sensor planning process. This information is used to compute the region of tolerance to error for the task. The following two additional information items are associated with each ICdg arc:

- Lists of points, further referred as *inequality points*, that are used to get constraint inequalities that describe the region, and
- Constraining planes which limit the displacement of the inequality points as result of errors in the configuration of the dependent object.

The tail of an ICdg arc identify the constrained object of an alignment constraint, while its head identify the constraining object. Then, if for any reason the constraining object's configuration changes, the constrained object's configuration should be adjusted accordingly. For this reason, the inequality points are located on (real or virtual) surfaces of the constrained object, while the constraining planes are planes associated with (real or virtual) faces of the constraining object.

Constraining planes and planes where inequality points reside are parallel to each other. The distance between these constraining and constrained planes describe allowed translational errors with respect to a reference vector. These translational errors are also propagated during the propagation of alignment constraints.

There is difference between the way the above information elements are obtained for insertion tasks and the way used for environmental constraining. Insertion tasks usually include some predefined level of tolerance, described as the distance between the constrained faces of the manipulated object and the constraining faces of the environment. On the contrary, environmental objects are expected to accurately conform the contacting and insertion scenes. However, if error is not tolerated in the pose of environmental objects, then there is not possibility of success ($Psp = 0$), and decisions for sensor configurations have to rely only on knowledge about the sensing uncertainty of the sensor in the different viewpoints.

For those cases, the following two alternatives were considered to evaluate the viewpoints:

1. Use only a criteria based on the size of the uncertainty region, and
2. Introduce an artificial distance describing the translational error for environmental objects.

The benefit of the first alternative is that it does not include an artificially designated quantity to compute the task tolerance region. However, this alternative has two important drawbacks: First, that it ignores the relevance of the specific geometry of the task; and second, that a mechanism has to be developed for the fusion, in some cases, of the two non-homogeneous measures obtained for a same viewpoint.

The first problem is very significant in the case of rotation errors, because the distance traveled by a point in a rigid object, when this object is rotated, is proportional to the straight distance from its rotation reference. Which means that rotation makes different points on the object translate different distances, as depicted in the example presented in Figure 4.5. Then the effect of the same rotational errors when assembling objects with different geometry in the same environment configuration, or when assembling the same objects in environments with different geometries would vary.

The second problem becomes relevant when the object to be observed is the manipulated object of a task that require of using insert assembly skill primitives and this manipulated object will participate in subsequent tasks as part of the environment. Again, the problem rises because the specific geometry of the task is ignored, and then, it is difficult to find a relation between the size of uncertainty region of the sensor, which is the only element considered when the object conform the environment of other task, and a predicted success probability for its insertion when is manipulated by a robot.

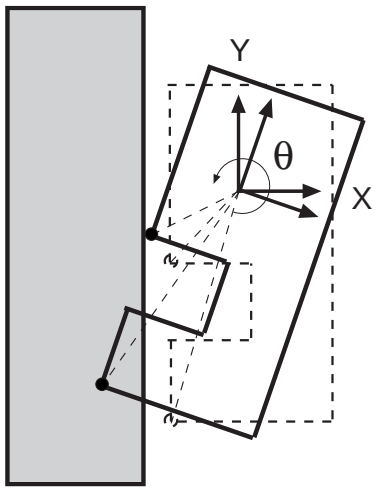


Figure 4.5: Effect of rotation errors in constraining relations.

The drawback of the second alternative is what was mentioned as the benefit of the first alternative: adding an artificial tolerance to error. However, this addition solve both drawbacks of the first alternative allowing to compute a tolerance region and use the same criteria for manipulated and environmental objects. Because the benefit of this alternative was considered to exceed its drawback, this was the chosen one. Since the purpose of this method is to determine the best viewpoints, it was considered as more important getting an homogeneous comparison base.

The obtainment of the set of inequalities that define the error tolerance region of a task is based in the displacement produced in the inequality points as a consequence of uncertainty in the pose of a manipulated object. These particular points are vertices that describe polygonal faces on restricted surfaces, of the manipulated object, due to an assembly task. The new equations of constraint for an object are generated by the two following steps:

1. Determine the inequality points and the constraining planes.

2. Obtain inequalities based on tolerated motion of inequality points.

4.3.2 Deducing Inequality Points and Constraining Planes for Insertion

In the case of an insertion operation, the task includes some pre-conceived clearance as a separation between constrained and constraining surfaces. The task tolerance region ($\mathcal{T}$) for the insertion of polyhedral objects into polyhedral slots is obtained as a function of the distance between vertices and surfaces of the participant objects in the

operation. This distance represents the planned clearance for a task and determines the translational errors associated with each reference vector.

The deduction of the inequality points for insertion is realized in the following two sub-steps:

Sub-step 1: Determining insertion planes

An insertion plane for insertion is used to formulate the criterion to select the inequality points. An insertion operation is considered successful if the set of inequality points cross the insertion planes determined for the task. An insertion plane is oriented in the direction of insertion, s.t. its normal vector, described with respect to the coordinate frame of the manipulated object, is obtained as

$$\mathbf{n}_{ins} = T_M \mathbf{d} \tag{4.2}$$

where $\mathbf{d}$ is a unitary vector in world coordinates that describe the direction of insertion, and T_M is an homogeneous transformation matrix to convert from world coordinates to the manipulated object coordinates.

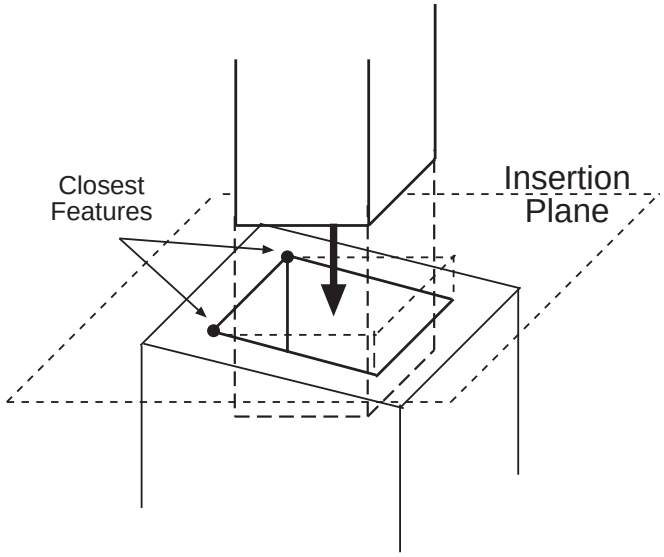


Figure 4.6: Insertion plane in an insertion operation.

To complete a formal description of an insertion plane as

$$\mathbf{n}_{ins} \cdot \mathbf{x} = o_{ins} \tag{4.3}$$

the orthogonal distance o_{ins} to the plane from the origin of the manipulated object's

frame need to be computed. This computation is based on the resulting configuration of the scene after the insertion task. Since, commonly, only a portion of the manipulated object is inserted in an assembly step, an insertion plane is placed in a position that delimits this portion in the insertion direction, as shown in the example of Figure 4.6.

The movement constraint defined by the insertion is a result of new relations between pairs of surfaces where a portion of an environmental surface almost contacts a portion of a surface of the manipulated object. These portions are computed as polygonal regions of intersection, and are described by list of points, for each new surface pair. Figure 4.7 depicts an example of this. The shadowed region in the figure, defined by the list of points p_1, p_2, p_3, p_4, describe the region of intersection of top face of Object A, labeled as S_A, with the bottom face of Object B, labeled as S_B. In the case of insertion tasks, the constraining face pairs are deduced by using a threshold that define the maximum distance between faces that make the task critical.

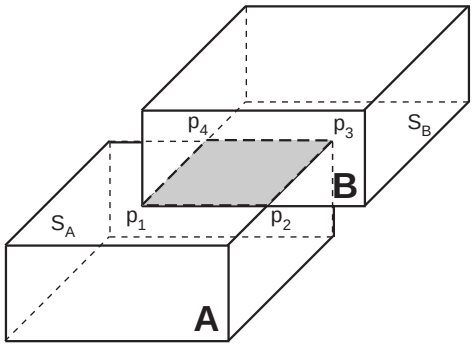


Figure 4.7: Intersection region computation for contacting planar surfaces.

An insertion plane has to be placed over the first vertex $\mathbf{v}_f$ of a set of vertices of an intersections' list, when these vertices are ordered in agreement with the insertion direction, i.e. $\mathbf{v}_f = \mathbf{v}_i$ if $\mathbf{n}_{ins} \cdot \mathbf{v}_i \leq \mathbf{n}_{ins} \cdot \mathbf{v}_j$ for all $\mathbf{v}_j \neq \mathbf{v}_i$. Having determined this special vertex, the searched orthogonal distance to the plane is computed as

$$o_{ins} = \mathbf{n}_{ins} \cdot \mathbf{v}_f. \tag{4.4}$$

An insertion plane has to be obtained for each pair of constraining surfaces because the distance between different pairs of surfaces can vary, and also the distance of the inequality points from the origin of the manipulated object's coordinate frame.

Sub-Step 2: Determining inequality points and constraining planes

Sets of inequality points should be deduced for each insertion plane. The inequality points are those vertices of new restricted surfaces in the manipulated object that have

been effectively inserted in an environmental configuration. Though, in general is difficult to determine these points exactly, in agreement with the decided form of getting the inequalities, it was decided to denote as inequality points to all vertices on the constrained face of the manipulated object and all points in the intersection, which were over the insertion plane before the insertion task and below the insertion plane after the insertion. This means that every point in this group with coordinates $\mathbf{v}_M$ before the insertion task and coordinates $\mathbf{v}'_M$ after the insertion will be considered an inequality point if

$$(\mathbf{v}_M \cdot \mathbf{n}_{ins} \leq o_{ins}) \wedge (\mathbf{v}'_M \cdot \mathbf{n}_{ins} \geq o_{ins}). \tag{4.5}$$

The alignment dependency of a manipulated object on environmental objects would not be complete if there are not alignment references fixed on those environmental objects. These alignment references are the constraining planes. A constraining plane must be defined for each set of inequality points. Since there is a set of inequality points for each pair of constraining and constrained faces, the constraining plane is described by the face of the environmental object participating in the pair.

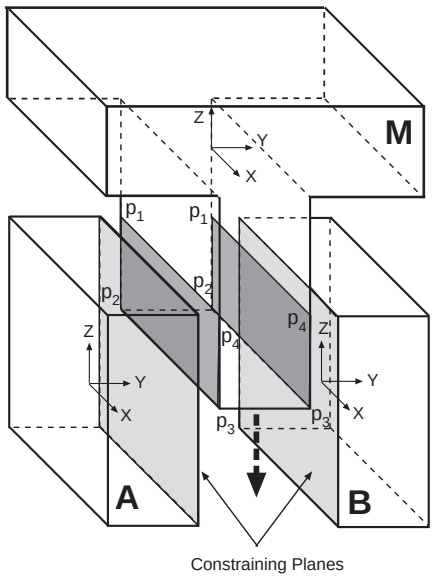


Figure 4.8: Inequality points and constraining planes for an insertion task.

Figure 4.8 illustrate the result of an insertion operation showing the inequality points as lists of four points on the dark shadowed part over the manipulated object M and the constraining planes over the surfaces of environmental objects A and B.

4.3.3 Deducing Inequality Points and Constraining Planes for Environmental Objects

The inequality points for environmental objects are found in the constraining surfaces of the environmental objects that conform the direct environment configuration for an assembly task that cause a propagation of alignment constraints. The inequality points associated with each of these objects are those that conform the list of vertices that define the intersection region of constraining face pairs.

The same inequality points are used for the environmental objects that participate in an indirect way. The rationale for this is that if an indirect alignment constraint is violated this will eventually affect the position and/or orientation of environmental objects that directly conforms the environment of the task that provoked the propagation process that added such indirect alignment constraint.

Since the inequality points of an environmental object that indirectly affects the environment configuration of a task are points over a surface of other objects, they have to be transformed to its own coordinate reference. This works like a virtual extension of its geometry by adding new faces implicitly described by the inequality points. Then this object can be treated as if it were directly participating in the critical constraining relation.

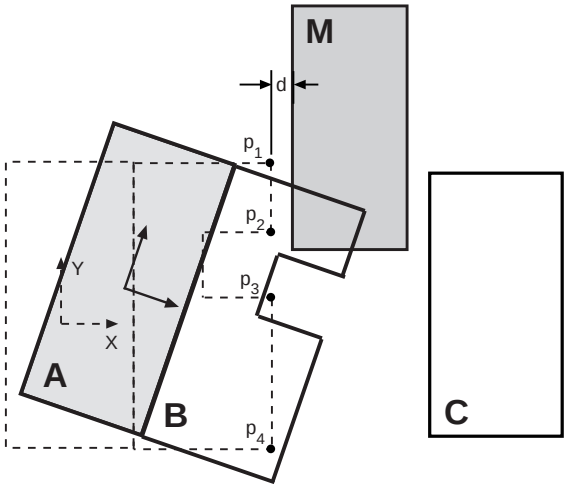


Figure 4.9: Indirect effect of violating a propagated alignment constraint.

Figure 4.9 presents an example of this situation, where an error in the configuration of Object A directly affects the configuration of Object B, which, in turn, affects the insert configuration that forms with Object C. Object B is the environmental object that directly participate in the insert configuration. This situation will eventually make that an insertion operation of manipulated object M can not be performed.

Two constraining planes for each constrained environmental surface, implicitly described

by inequality points, are situated to a constant distance defined as a global parameter for the assembly plan. One constraining plane to each side of the constrained plane. Both planes are fixed, as in the case of insertion, to the environment objects on which the analyzed object depends. This also works, like in the case of the inequality points, virtually extending the geometry of those objects.

The fixing of constraining planes to objects on which other objects depend is not strictly necessary for computing the region of tolerance to error. That step is needed for having references to compute the correction of the configuration of the object in function of the configuration of the other objects during the actual execution of the assembly plan.

4.3.4 Obtaining Task Tolerance Inequalities

Each inequality point will be associated with one constraining plane located in an environmental object. From each association one possible redundant inequality will be obtained. A constraining plane divides the assembly space into two subspaces, a constrained subspace and an unconstrained subspace. A constraining plane restricts the magnitude and form of motion of an inequality point by keeping it in the unconstrained subspace. The deduction of the following inequalities assumes that the configuration of a constraining plane is known exactly.

Getting a Task Tolerance Inequality

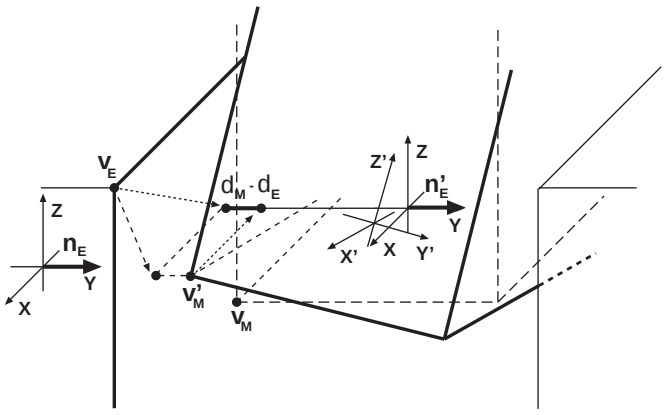


Figure 4.10: Elements of Task Tolerance Inequalities.

As depicted in Figure 4.10, the inequality point $\mathbf{v}_M$ delimits its margin of motion error by its orthogonal distance to the environmental surface S_E. Before the assembly operation, the uncertainty effect might displace the inequality point to a position $\mathbf{v}'_M$, which could be obtained from its planned position as

$$\mathbf{v}'_M = \Delta\mathbf{T}_M \mathbf{v}_M \tag{4.6}$$

where the homogeneous transformation matrix $\Delta\mathbf{T}_M$ is described by

$$\Delta\mathbf{T}_M = \begin{bmatrix} & \Delta\mathbf{R} & & \Delta\mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.7}$$

Its elements are detailed in Appendix A. This matrix quantifies the allowed rotational error, codified as matrix ΔR, and the allowed translational error, described by vector Δt, in the manipulated object pose.

Each inequality has the form

$$d_M - d_E \leq 0 \tag{4.8}$$

which represents the difference between distance d_M, representing the signed distance from the origin of the manipulated object's frame to the projection of the displaced vertex vb'_M over an axis defined by the normal vector $\mathbf{n}'_E$ to the environmental plane S_E, and distance d_E, representing the orthogonal distance from the origin of the manipulated object's frame to the environmental plane.

Distance d_M is computed as

$$d_M = \mathbf{v}'_M \cdot \mathbf{n}'_E \tag{4.9}$$

for which it is needed to apply a change of coordinate frame reference to normal vector $\mathbf{n}_E$ using

$$\mathbf{n}'_E = \mathbf{T}_M \mathbf{T}_E^{-1} \mathbf{n}_E \tag{4.10}$$

where the homogeneous transformation matrices $\mathbf{T}_M$ y $\mathbf{T}_E$ describe the manipulated object pose and the environmental object pose, respectively.

Distance d_E is computed as

$$d_E = \mathbf{v}'_E \cdot \mathbf{n}'_E \tag{4.11}$$

where $\mathbf{v}'_E$ is any vertex on the environmental plane, usually a vertex, subject to a change of coordinates similar to the normal vector case, i.e. its new coordinates are computed by

$$\mathbf{v}'_E = \mathbf{T}_M \mathbf{T}_E^{-1} \mathbf{v}_E. \tag{4.12}$$

Considering Critical Dimensions of a Task in 2D-Assembly

In general, the form exposed in the preceding section for computing a constraining inequality is correct; however, it does not consider the critical dimensions of the task. The critical dimensions of an assembly operation are important when obtaining its task tolerance inequalities because they describe the most sensitive DOF of the step and those that should be corrected.

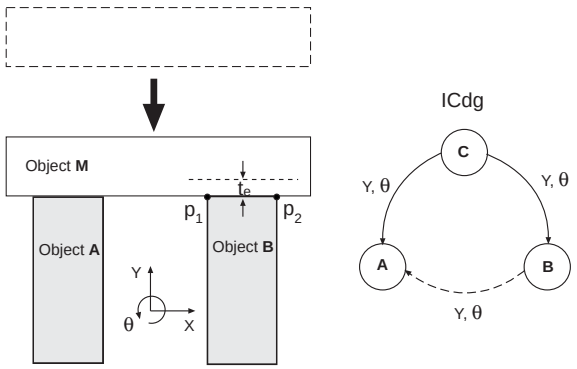


Figure 4.11: A multiple contact configuration where the dependent object is completely unconstrained.

To illustrate this, the process for obtaining constraint inequalities in 2D-assembly is analyzed. The constraining relation caused by a make-contact operation in which a manipulated object enters in contact with multiple environmental objects is used as the illustration instrument. Every critical relation between a constrained plane described by inequality points and a constraining plane describing the tolerance to error can be processed in the same way.

Figure 4.11 presents an example of this. In this figure, there are three objects, the manipulated object M, and the environmental objects A and B. All objects share the same coordinate frame of reference, F_w, and the contacting surfaces are aligned with Y axis. The assembly order adds Object A before than Object B, then when Object M is moved to contact, a joining constraint dependency is generated for Object B with respect to Object A. The critical dimensions, the Y DOF for translation and the origin of the reference frame for rotation, are presented in the ICdg also in Figure 4.11. Since in this case, the dependency is between environmental objects, an artificial constraining plane is defined to a t_e distance from the plane described by the inequality points, p_1 and p_2.

In this case, the constraint inequality for a inequality point $\mathbf{p}$ is computed as

$$(\mathbf{p}' - \mathbf{p})_y \leq t_e \tag{4.13}$$

where $\mathbf{p}'$ is inequality point $\mathbf{p}$ perturbed by uncertainty. The subscript y refers to the Y coordinate of the difference, since the contact is oriented in the Y direction. Since $\mathbf{p}'$ coordinates are computed by

$$\mathbf{p}' = \Delta\mathbf{T}\mathbf{p} = \Delta\mathbf{R}\mathbf{p} + \Delta\mathbf{t} = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x\cos(d\theta) - y\sin(d\theta) + dx \\ x\sin(d\theta) + y\cos(d\theta) + dy \end{bmatrix} \tag{4.14}$$

where x, y, and z are the position coordinates of $\mathbf{p}$; $d\theta$ is the rotation error angle caused by uncertainty; and dx and dy are the translation error displacements caused by uncertainty. The constraint inequality is obtain as

$$(\Delta\mathbf{T}\mathbf{p} - \mathbf{p})_y = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} (\Delta\mathbf{T}\mathbf{p} - \mathbf{p}) \leq t_e \tag{4.15}$$

Then, the constraint inequality generated from $\mathbf{p}$ for this case is:

$$x\sin(d\theta) + y\cos(d\theta) + dy - y \leq t_e. \tag{4.16}$$

The error that should be corrected is described by $y' - y$ in translation and $\theta' - \theta$ in rotation.

Figure 4.12 depicts two examples of task tolerance regions for an environmental object B in a make-contact operation as that illustrated in Figure 4.11. Two constraining planes are symmetrically located to both sides of the constrained plane described by two inequality points. Each plane is located at a distance t_e of 1. Then, each region is described by four constraint inequalities. The regions have a curved rhombus shape. The smaller region (darker rhombus region) is described by inequality points in positions $(5, 5)$ and $(-5, 5)$ which produce the following inequality constraints

$$\begin{aligned} CstrIneq\ 1 &= 5\sin(d\theta) + 5\cos(d\theta) + dy - 5 \leq 1 \\ CstrIneq\ 2 &= -5\sin(d\theta) + 5\cos(d\theta) + dy - 5 \leq 1 \\ CstrIneq\ 3 &= 5\sin(d\theta) + 5\cos(d\theta) + dy - 5 \leq -1 \\ CstrIneq\ 4 &= -5\sin(d\theta) + 5\cos(d\theta) + dy - 5 \leq -1. \end{aligned} \tag{4.17}$$

The bigger region is described by inequality points in positions $(2, 5)$ and $(-2, 5)$ which produce the following inequality constraints

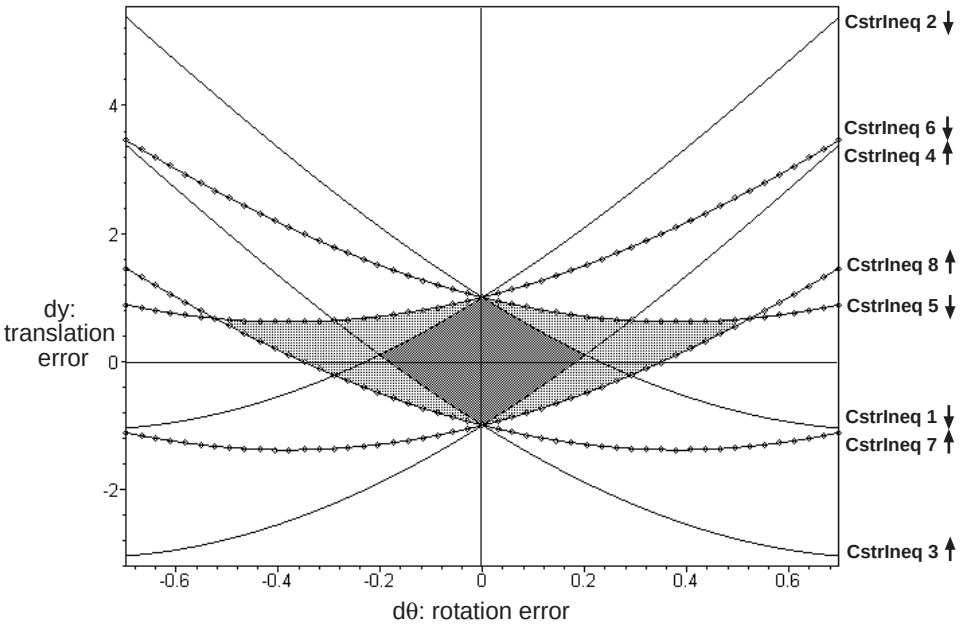


Figure 4.12: Examples of Task Tolerance Regions for two initially unconstrained environmental objects in a make-contact operation.

$$\begin{aligned} CstrIneq\ 5 &= 2\sin(d\theta) + 5\cos(d\theta) + dy - 5 \leq 1 \\ CstrIneq\ 6 &= -2\sin(d\theta) + 5\cos(d\theta) + dy - 5 \leq 1 \\ CstrIneq\ 7 &= 2\sin(d\theta) + 5\cos(d\theta) + dy - 5 \leq -1 \\ CstrIneq\ 8 &= -2\sin(d\theta) + 5\cos(d\theta) + dy - 5 \leq -1. \end{aligned} \tag{4.18}$$

As noted, the only difference in the two situations is the X position of the inequality points. This demonstrate the relevance of the position of inequality points and illustrates the redundancy of some constraint inequalities; if the four inequality points were used to construct a single task tolerance regions, inequalities $CstrIneq$ 5, $CstrIneq$ 6, $CstrIneq$ 7, and $CstrIneq$ 8 become redundant.

A slight complication could appears when each object have its own coordinate frame, and there is a coordinate frame for the assembly, as it effectively is in this work. The pose of each object is described by a homogeneous transformation with respect to the assembly frame.

In this case, to get the tolerance error t_e the constraining plane have to be described with respect to coordinate frame of object B by applying a change of reference with transformation matrix

$$\mathbf{T}_{AB} = \mathbf{T}_B \mathbf{T}_A^{-1} \tag{4.19}$$

where $\mathbf{T}_B$ is the transformation matrix that describe the pose of Object B and $\mathbf{T}_A^{-1}$ is the inverse of the transformation matrix that describe the pose of Object A.

If, additionally, the constrained and constraining planes are not orthogonal to any of the coordinate axis of Object B, the alignment should be forced in order to get the constraint inequality. In this case the convention is to align the planes with respect to the Y axis. This alignment is done after computing the uncertainty-perturbed coordinates of the inequality points. The new constraint inequality equation is

$$(\mathbf{p}'_{aligned} - \mathbf{p}_{aligned})_y = (\mathbf{R}_{align}(\psi)[\mathbf{p}' - \mathbf{p}])_y = (\mathbf{R}_{align}(\psi)[\Delta\mathbf{T}(d\theta, d\mathbf{t})\mathbf{p} - \mathbf{p}])_y \leq t_e \quad (4.20)$$

where $\mathbf{R}_{align}$ is a rotational matrix that align the unit normal vector $\mathbf{n}_c$ of the constrained plane with the Y axis of the perturbed object by rotating it an angle

$$\psi = \cos^{-1}(\mathbf{j} \cdot \mathbf{n}_c) \quad (4.21)$$

where $\mathbf{j}$ is unit vector in the direction of Y axis.

Developing Inequality 4.20, the resulting constraint inequality is

$$\begin{array}{l} \sin(\psi)(x\cos(d\theta) - y\sin(d\theta) + dx - x) + \\ \cos(\psi)(x\sin(d\theta) + y\cos(d\theta) + dy - y) \quad \leq t_e \end{array} \quad (4.22)$$

It can be seen that Inequality 4.22 is consistent with Inequality 4.16 for $\psi = 0$.

In the cases covered above, Object B is not in contact with any other environmental object, and that is why the critical dimensions are the translation with respect to Y axis and the orientation. Since a simple contact in 2D-assembly fix the orientation of an object, if Object B is in contact with another object, as illustrated in Figure 4.13, its critical dimensions for vision will decrease.

In Figure 4.13, the contact between Object B and the third environmental object, Object C is parallel to the constrained and constraining planes. This, as explained by the third case of the method for the propagation of alignment constraints of Chapter 3, means that both critical dimensions for Object B are fixed by its contact relation with Object C. This is the reason for the absence of a direct arc between nodes A and B in the ICdg illustration. This absence means that if Object C is exactly aligned with Object A, there is not possibility of error in the Y position and orientation of Object B.

The above conclusion can be realized from an analysis of the effect of errors in the *potential critical dimensions* of a task. Also from Chapter 3 description, it should be understood that a critical dimension for vision is associated with a DOF that was

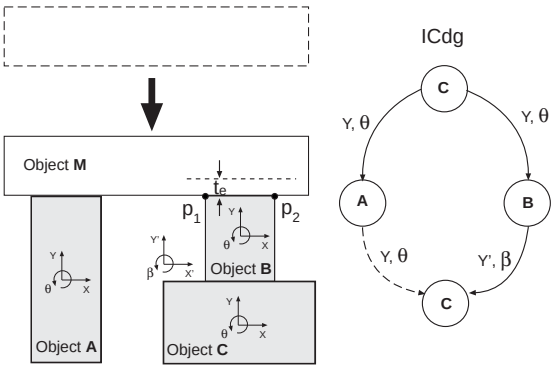


Figure 4.13: A multiple contact configuration where the dependent object critical configuration is completely fixed by a third environmental object.

completely unconstrained before a task and gets completely constrained after the task. Then, a relatively small error with respect to this DOF could violate a condition for successfully achieving a required contact configuration. Then, the potential critical dimensions of a task are all the unconstrained DOF.

In the example illustrated in Figure 4.13, Object B can only move, without breaking the contact with Object C, with respect to the X axis. Then X DOF would be the only potential critical dimension for vision. Since a motion in a X direction would not perturb the position of the inequality points in the Y direction, their constraint could not be violated. Then the X DOF is not a critical dimension for this task.

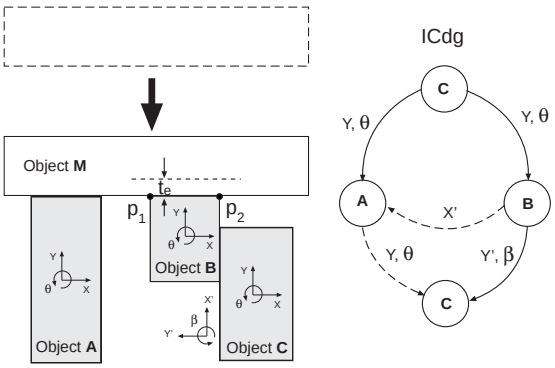


Figure 4.14: A multiple contact configuration where the dependent object is in a contact relation which is orthogonal to the constraining plane.

Figure 4.14 presents another example where the configuration of environmental object B, an object that directly participate in a multiple contact configuration, is restricted by the third environmental object C. But in this case, the contact is aligned with the X axis of the coordinate frame of Object B. Here, the only potential critical dimension is the Y DOF. Since a motion in such direction could violate the constraint, Y is a critical

dimension, and its constraint inequality is

$$dy \leq t_e \tag{4.23}$$

As can be noted, Inequality 4.23 does not include variables related with coordinates of the inequality points. In such cases, only one constraint inequality is required, since the rest of the inequality points will produce redundant inequalities.

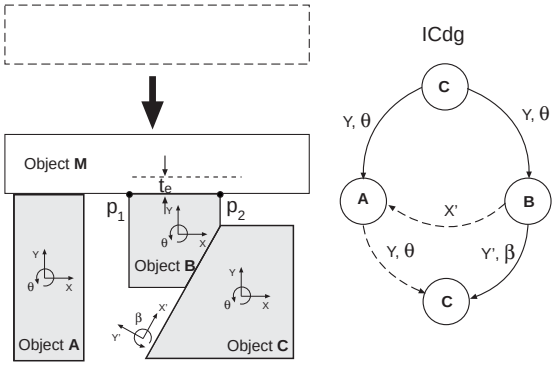


Figure 4.15: A multiple contact configuration where the dependent object is in a contact relation which is inclined to the constraining plane.

Figure 4.15 presents another example of constrained configuration for environmental object B, but in this case the contact with Object C is not parallel nor orthogonal to the constrained and constraining planes. Then, the only potential critical dimension is aligned with the X' axis of a new coordinate frame, further referred as a *contact frame* because is aligned with the contact between objects B and C. X' DOF is a critical DOF because is oblique to the constraining plane and an error in its value could violate the recognized constraint.

To get the constraint inequality for this case, an error with respect to X' DOF has to be converted to an error with respect to Y DOF. This is done as

$$\Delta\mathbf{T} = \mathbf{R}_{correct}^{-1}\Delta\mathbf{T}'\mathbf{R}_{correct} \tag{4.24}$$

where $\mathbf{R}_{correct}$ is a transformation matrix that describe the rotation required to align the coordinate frame of Object B with the contact frame, and $\Delta\mathbf{T}'$ is a transformation matrix describing the effect of translation with respect to X' axis. Then

$$\begin{aligned}\Delta\mathbf{T} &= \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & dx' \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & dx'\cos(\phi) \\ 0 & 1 & -dx'\sin(\phi) \\ 0 & 0 & 1 \end{bmatrix}\end{aligned} \tag{4.25}$$

where ϕ is the angle that the constrained and constraining planes should be rotated to be aligned with the Y axis of the contact frame, and dx' is a potential error with respect to X' DOF.

The resulting constraint inequality for oblique constraining contacts with respect to the constraining planes is

$$-dx'\sin(\phi) \leq t_e \tag{4.26}$$

Again, the constraining relation of Object B with respect to Object C only allows translation of inequality points, and then, only one constraint inequality is required because all points produce the same inequality.

Inequality 4.26 can also be used to describe the parallel case ($\phi = 0$ in Figure 4.13) and the orthogonal case ($\phi = -\pi/2$ in Figure 4.14).

Finally, if neither the constrained and constraining planes nor the contact between an analyzed object and another environmental object are aligned with Y axis of the coordinate frame of the analyzed object, the constraint inequality is obtained as

$$dx'\cos(\phi)\sin(\psi) - dx'\sin(\phi)\cos(\psi) \leq t_e \tag{4.27}$$

where as mentioned before ψ is the angle that align the constrained and constraining planes with the Y axis.

Considering Critical Dimensions of a Task in 3D-Assembly

The analysis for the assembly in 3D-space is similar to the analysis presented above for the 2D case. The main difference is due to the complexity added by the rotation with respect to sets of orthogonal rotation axes. The general form of a 3D rigid transformation matrix is presented in Appendix A.

For getting the constraint inequalities in 3D-assembly, the convention is to align the constrained and constraining planes with the Z axis of the analyzed object. The general form of a 3D constraint inequality is

$$(\mathbf{p}' - \mathbf{p})_z = (\Delta\mathbf{T}\mathbf{p} - \mathbf{p})_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} (\Delta\mathbf{R}\mathbf{p} + \Delta\mathbf{t} - \mathbf{p}) \leq t_e \tag{4.28}$$

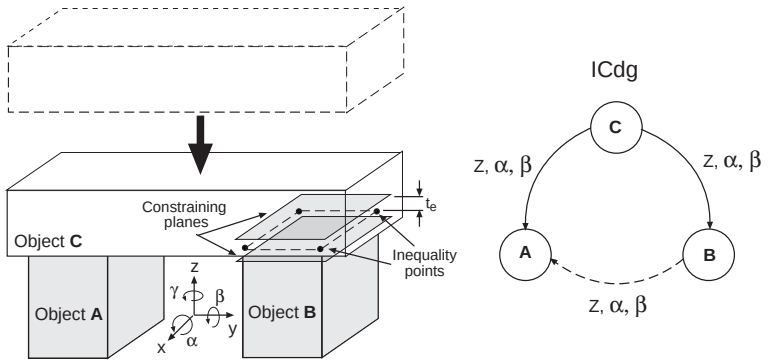


Figure 4.16: A multiple contact configuration where the dependent object is completely unconstrained.

For an object that does not participate in contacts with other environmental objects, as Object B in Figure 4.16, all the DOF are potentially critical; however, only three are really critical: translation with respect to Z axis and rotation with respect to X and Y axes. The constraint inequality for this case is

$$-x\sin(d\beta) + y\cos(d\beta)\sin(d\alpha) + z\cos(d\beta)\cos(d\alpha) + dz - z \leq t_e \tag{4.29}$$

where $d\alpha$ is the error angle with respect to X rotation axis; $d\beta$ is the error angle with respect to Y rotation axis; dz is the translation error displacement in the Z direction; and x, y, and z are the position coordinates of an inequality point.

Figure 4.17 depicts examples of task tolerance regions for an environmental object B in a make-contact operation as that illustrated in Figure 4.11. Two constraining planes are symmetrically located to both sides of the constrained plane described by four inequality points. The inequality points are located in positions $(5, 5, 5)$, $(-5, 5, 5)$, $(5, -5, 5)$, and $(-5, -5, 5)$. Each plane is situated at a distance t_e of 2. Then, Task Tolerance Region is described by the following eight constraint inequalities

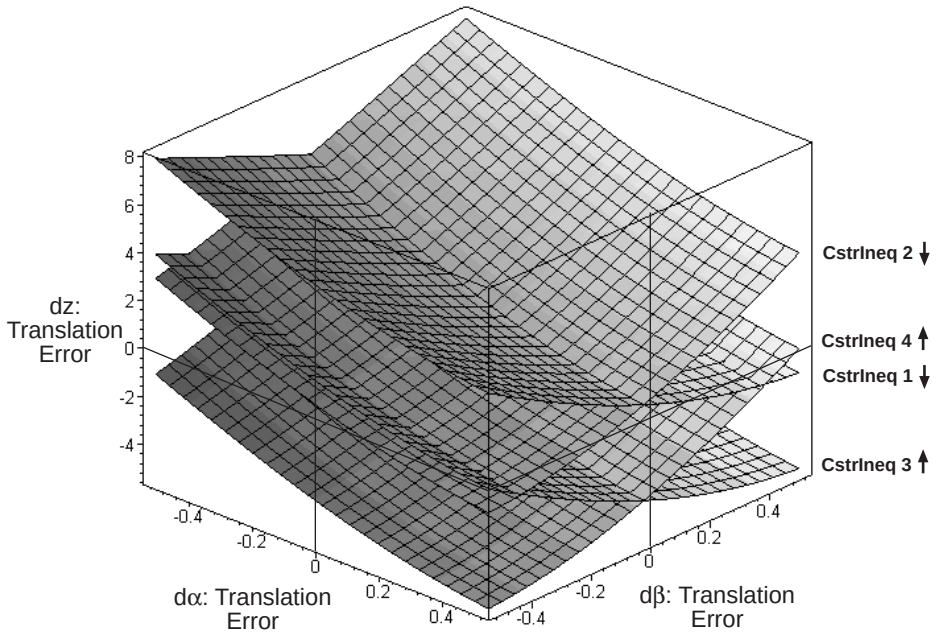


Figure 4.17: Examples of constraint inequalities for an initially unconstrained environmental object in a make-contact operation.

$$
\begin{aligned}
CstrIneq\ 1 &= -5\sin(d\beta) + 5\cos(d\beta)\sin(d\alpha) + 5\cos(d\beta)\cos(d\alpha) + dz - 5 \leq 2 \\
CstrIneq\ 2 &= 5\sin(d\beta) + 5\cos(d\beta)\sin(d\alpha) + 5\cos(d\beta)\cos(d\alpha) + dz - 5 \leq 2 \\
CstrIneq\ 3 &= -5\sin(d\beta) + 5\cos(d\beta)\sin(d\alpha) + 5\cos(d\beta)\cos(d\alpha) + dz - 5 \leq -2 \\
CstrIneq\ 4 &= 5\sin(d\beta) + 5\cos(d\beta)\sin(d\alpha) + 5\cos(d\beta)\cos(d\alpha) + dz - 5 \leq -2 \\
CstrIneq\ 5 &= -5\sin(d\beta) - 5\cos(d\beta)\sin(d\alpha) + 5\cos(d\beta)\cos(d\alpha) + dz - 5 \leq 2 \\
CstrIneq\ 6 &= 5\sin(d\beta) - 5\cos(d\beta)\sin(d\alpha) + 5\cos(d\beta)\cos(d\alpha) + dz - 5 \leq 2 \\
CstrIneq\ 7 &= -5\sin(d\beta) - 5\cos(d\beta)\sin(d\alpha) + 5\cos(d\beta)\cos(d\alpha) + dz - 5 \leq -2 \\
CstrIneq\ 8 &= 5\sin(d\beta) - 5\cos(d\beta)\sin(d\alpha) + 5\cos(d\beta)\cos(d\alpha) + dz - 5 \leq -2
\end{aligned}
\tag{4.30}
$$

To reduce the complexity of the image, only the first four constraint inequalities are shown in Figure 4.17. The resulting region is a curved polyhedron with eight curved triangular faces. The illustrated four constraints inequalities describe a kind of tube with a cross section similar to that of the task tolerance region for the 2D-case. The other four constraints describe a similar tube that intersects the first one to get the task tolerance region for this case.

As in 2D-assembly if the constrained and constraining planes are not orthogonal to the

Z coordinate axis of the analyzed object, Object B in the examples, the alignment should be forced in order to get the constraint inequality. This alignment is done after computing the uncertainty-perturbed coordinates of the inequality points. The new constraint inequality equation is obtain by

$$\begin{aligned}(\mathbf{p}'_{aligned} - \mathbf{p}_{aligned})_z &= (\mathbf{R}_{align}(\psi_x, \psi_y, 0)[\Delta\mathbf{T}\mathbf{p}' - \mathbf{p}])_z \\ &= (\mathbf{R}_{align}(\psi_x, \psi_y, 0)[\Delta\mathbf{R}(d\alpha, d\beta, d\gamma)\mathbf{p} + \Delta\mathbf{t} - \mathbf{p}])_z \leq t_e\end{aligned} \tag{4.31}$$

where $\mathbf{R}_{align}$ is a rotation matrix that aligns the normal vector $\mathbf{n}_c$ of the constrained plane with the Z axis of the perturbed object. $\mathbf{R}_{align}$ is obtained as product of two rotation matrices. First, a rotation matrix that rotates $\mathbf{n}_c$, with respect to the X axis, to align it with the XZ plane. The rotation angle is

$$\psi_x = \tan^{-1}\left(\frac{y_{\mathbf{n}_c}}{z_{\mathbf{n}_c}}\right) \tag{4.32}$$

where $y_{\mathbf{n}_c}$ and $z_{\mathbf{n}_c}$ are Y and Z coordinates of $\mathbf{n}_c$. Then, a rotation matrix to finish the alignment with the Z axis by using the rotation angle

$$\psi_y = \frac{\pi}{2} - \cos^{-1}(x_{\mathbf{n}_c}) \tag{4.33}$$

where $x_{\mathbf{n}_c}$ is the X coordinate of $\mathbf{n}_c$.

Developing Inequality 4.31, the resulting constraint inequality is

$$\begin{gathered}-\sin(\psi_y)(x\cos(d\gamma)\cos(d\beta) + y(-\sin(d\gamma)\cos(d\alpha) + y\cos(d\gamma)\sin(d\beta)\sin(d\alpha)) \\ +z(\sin(d\gamma)\sin(d\alpha) + \cos(d\gamma)\sin(d\beta)\cos(d\alpha)) + dx - x) + \\ \cos(\psi_y)\sin(\psi_x)(x\sin(d\gamma)\cos(d\beta) + y(cos(d\gamma)\cos(d\alpha) + \sin(d\gamma)\sin(d\beta)\sin(d\alpha)) \\ +z(-\cos(d\gamma)\sin(d\alpha) + \sin(d\gamma)\sin(d\beta)\cos(d\alpha)) + dy - y) + \\ \cos(\psi_y)\cos(\psi_x)(-\sin(d\beta)x + y\cos(d\beta)\sin(d\alpha) + z\cos(d\beta)\cos(d\alpha) + dz - z) \leq t_e\end{gathered} \tag{4.34}$$

Inequality 4.34 is quite more complex than Inequality 4.29, and reduces to this when $\psi_x = 0$ and $\psi_y = 0$. Something that should be noted is that the inequality expression also includes $d\gamma$ which is the error angle with respect to the Z rotation axis, and dx and dy which are the translation error displacements with respect to axes X and Y. The reason for this is that the object localization result describe errors with respect to the object's coordinate frame, and since the constraining relation is not aligned with it, errors in any configuration parameter of the object can displace the inequality points in a critical direction.

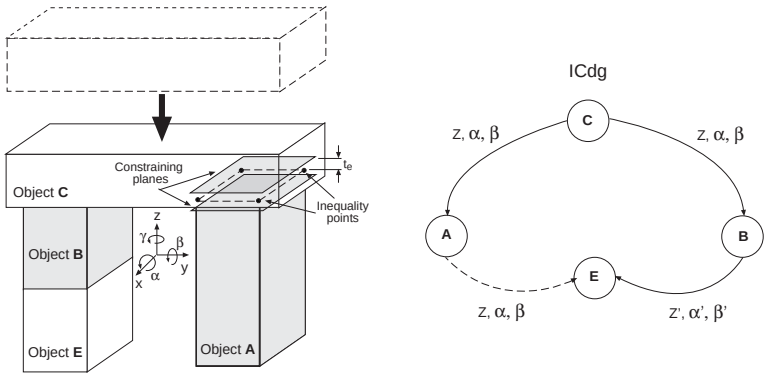


Figure 4.18: A multiple contact configuration where the dependent object critical configuration is completely fixed by a third environmental object.

In 3D-assembly, when a constrained object is in contact with another environmental object, as Object B is in contact with Object E in the example of Figure 4.18, three of its six possible DOF are fixed. In the illustrated case, the Z position and the α and β orientation of Object B are defined by Object E, which is assumed that was assembled before than Object B. Then the only possible critical dimensions for Object B are its X and Y position and its orientation with respect to the Z axis.

Now, since the constrained and constraining planes are orthogonal to the Z axis and none of the possible critical dimensions for object B appear in Inequality 4.29, object B does not have critical dimensions, and do not require of preventive vision.

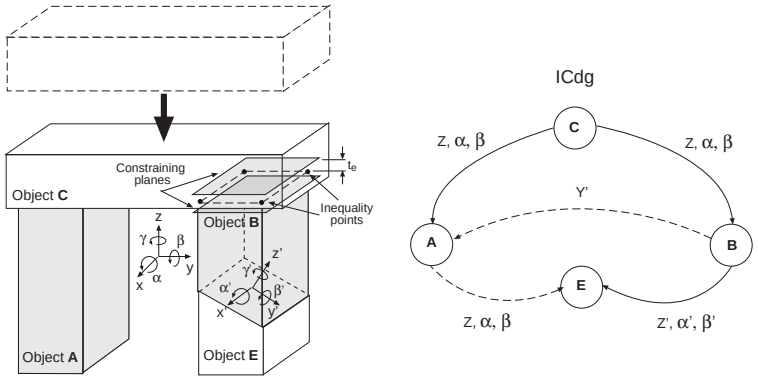


Figure 4.19: A multiple contact configuration where the dependent object is in a contact relation which is inclined to the constraining plane.

Figure 4.19 presents another example of constrained configuration for environmental object B, but in this case the contact with Object C is not parallel nor orthogonal to the constrained and constraining planes. Then, assuming a contact frame aligned with the contact, as illustrated in the example, the potential critical dimensions are related

with displacements in the X' and Y' directions and rotations with respect to the Z' axis. The X' and Y' axes can be oriented arbitrarily, but should form a basis with Z' axis, which is described by the contact direction between objects B and C.

To get the constraint inequality for this case, the errors with respect to the contact frame have to be translated to errors with respect to coordinate frame of Object B. This is done as

$$\Delta\mathbf{T} = \mathbf{R}_{correct}^{-1}\Delta\mathbf{T}'\mathbf{R}_{correct} \tag{4.35}$$

where $\mathbf{R}_{correct}$ is a transformation matrix that describe the rotation required to align the coordinate frames, and $\Delta\mathbf{T}'$ is a transformation matrix describing the effect of errors in the critical dimensions.

$\mathbf{R}_{correct}$ can be obtained in two steps: first, the unit vector representing the Z axis is aligned with the Z' axis (the contact direction) using the combined rotation strategy described before to align the constrained plane with the Z axis. ϕ_x, the alignment angle for the X' rotation, is obtained as ψ_x, and ϕ_y, the alignment angle for the Y' rotation, is obtained as ψ_y. To compute these values, the coordinates of a unit vector in the Z direction has to be converted to contact frame coordinates.

The second alignment step, should perform a rotation with respect to the Z' axis to align the rest of the axes, X and Y with X' and Y', respectively. This is done by using the rotation angle

$$\phi_z = -\tan^{-1}\left(\frac{y'_{\mathbf{X}}}{x'_{\mathbf{X}}}\right) \tag{4.36}$$

where $x'_{\mathbf{X}}$ and $y'_{\mathbf{X}}$ are the coordinates of the unit vector describing the X axis after applying the alignment rotation for the Z axis.

Then, the constraint inequality for this cases is obtained by

$$\left(\mathbf{R}_{correct}^{-1}(\phi_x,\phi_y,\phi_z)\begin{bmatrix}\cos(d\gamma') & -\sin(d\gamma') & 0 & dx'\\ \sin(d\gamma') & \cos(d\gamma') & 0 & dy'\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\mathbf{R}_{correct}^{-1}(\phi_x,\phi_y,\phi_z)\mathbf{p}-\mathbf{p}\right)_z \leq t_e \tag{4.37}$$

where $d\gamma$ is a potential error angle with respect to the Z' axis, dx' and dy' are the potential error displacements with respect to the X' and Y' axes, respectively.

The resulting constraint inequality for oblique constraining contacts with respect to the constraining planes is

$$
\begin{array}{l}
-x\cos(\phi_y)\sin(\phi_x)\sin(d\gamma') + x\cos(\phi_y)\cos(\phi_x)\sin(\phi_y)\cos(d\gamma') - y\sin(\phi_y)\sin(d\gamma') \\
-y\cos(\phi_x)\sin(\phi_x)\cos(d\gamma')\cos(\phi_y)^2 + dy'\cos(\phi_x)\sin(\phi_y)\sin(\phi_z) \\
+dx'\cos(\phi_x)\sin(\phi_y)\cos(\phi_z) + y\sin(\phi_x)\cos(\phi_y)^2\cos(\phi_x) + z\cos(\phi_x)^2\cos(\phi_y)^2 \\
+dx'\sin(\phi_x)\sin(\phi_z) + z\cos(d\gamma') - x\sin(\phi_y)\cos(\phi_x)\cos(\phi_y) \\
-dy'\sin(\phi_x)\cos(\phi_z) - z\cos(\phi_x)^2\cos(d\gamma')\cos(\phi_y)^2 - z \leq t_e
\end{array}
\tag{4.38}
$$

The convenience of choosing good alignments for the contact frame can be realized from the reduction of the complexity in the formula of the constraint inequality for aligned cases. For example, in the case presented in Figure 4.19 the X and X' axes are already aligned, then to align the rest of the axes is only necessary to rotate with respect to X' axis. The constraint inequality for this case is

$$
\begin{array}{c}
-x\sin(\phi_x)\sin(d\gamma') + y(-\cos(\phi_x)\cos(d\gamma')\sin(\phi_x) + \sin(\phi_x)\cos(\phi_x)) \\
+z(\sin(\phi_x)^2\cos(d\gamma') + \cos(\phi_x)^2) - dy'\sin(\phi_x) - z \leq t_e
\end{array}
\tag{4.39}
$$

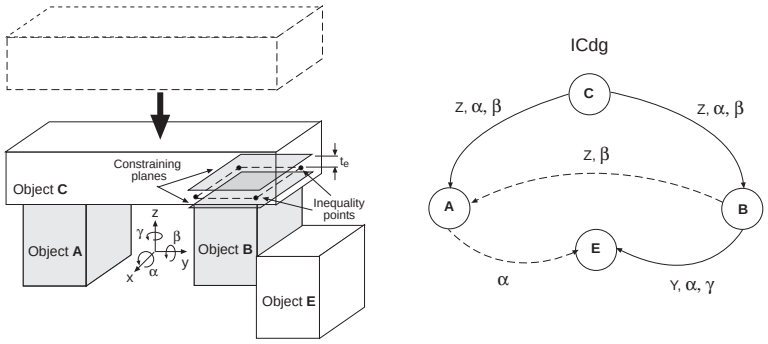


Figure 4.20: A multiple contact configuration where the dependent object is in a contact relation which is orthogonal to the constraining plane.

And, if additionally the contact is orthogonal to the Z axis, $\phi_x = \pi/2$, as in the example depicted in Figure 4.20, the constraint inequality gets reduced to

$$
-x\sin(d\gamma') + z\cos(d\gamma') - dy' - z \leq t_e. \tag{4.40}
$$

When the configuration of an analyzed object is constrained by two of more aligned contacts, its motion freedom is reduced to simple translation. The constraint inequality for this case is

$$
dx'\ cos(\phi_x)\sin(\phi_y)\cos(\phi_z) + dx'\sin(\phi_x)\sin(\phi_z) \leq t_e \tag{4.41}
$$

where the contact frame is aligned in such a way that the potential translation error is aligned with the X' axis. Something to note is that this inequality does not include the coordinate values of the inequality points, which means that only one inequality is necessary to represent the full constraint.

Finally, if neither the constrained and constraining planes nor the contact between an analyzed object and another environmental object are aligned with Z axis of the coordinate frame of the analyzed object, the general constraint inequality is obtained as

$$(\mathbf{R}_{align}[\mathbf{R}_{correct}^{-1}\Delta\mathbf{T}'\mathbf{R}_{correct}\mathbf{p} - \mathbf{p}])_z \leq t_e \tag{4.42}$$

but its developed form is no added in this document for obvious reasons.

4.4 Modeling and Quantifying Sensing Uncertainty

For this dissertation, the sensing uncertainty region is a representation of the expected error in the description of the pose of an object when obtained from a visual sensing mechanism. This region is a product of the variability on the results of the pose refinement process realized using a localization tool with images obtained from a sensor in predefined configurations. The size and shape of this uncertainty region depends on the type, configuration and calibration of sensor; the type, structure, and brightness of illumination; the shape, size, and photometric properties of the objects to localize; and the technique used for localization. Is a highly dimensional problem as has been demonstrated experimentally.

4.4.1 Experimental Testbed

In this dissertation, a description of the visual uncertainty of the sensor was developed to compute the *Psp* criteria used for ordering potential viewpoints for localizing the objects in assembly tasks. An empirical approach was selected to quantify a model of uncertainty for the used object localization method. The experimental testbed for this task is described below.

The Sensor

An active-camera mechanism composed by a pan-tilt computer-controllable camera and a Cartesian robot-arm was selected to perform the visual sensing strategies generated by the sensing analyzer using the best feasible viewpoints determined by the sensor planner(see Figure 5.1). In this mechanism, a camera is fixed to the robot hand. The

robot moves the camera without rotation to achieve its target position. The pan-tilt mechanism of the camera is used to achieve its target orientation.

A detailed description of the active sensor, its configuration, calibration, and use will be presented in Chapter 5.

The Illumination

The output's quality of a system depends on the quality of its input. A vision system then depends on the quality of its images. The distribution of light sources is one of the fundamental factors that determine the image's quality of a three-dimensional object. Other factors include the object's shape and reflectance properties, its attitude with respect to the imaging system, and its pose in space.

The detection of specific features on an image rely on the intensity of the picture cells (*pixels*) that forms it. The intensity of a pixel is determined by the amount of light falling in a specific region (irradiance) of the camera's captor surface. The irradiance of a point in the image is related to the amount of light radiated by the objects in the scene (radiance).

This dissertation does not deals with illumination planning, but recognizes that it is a very important factor that affects the accuracy of the localization process; importance realized in hundreds of experiments performed under many illumination conditions. Most of the experiments with real images where performed putting the target object for localization within a reconfigurable illumination environment, where lights were uniformly distributed over an illumination sphere. The sources of illumination were specifically (manually) selected to enhance the relevant features of the observed objects (its edges).

The Object Localization Tool

To deduce the configuration of the assembly objects in the scene a model-based localization tool (*2DTM*) is used. 2DTM localize 3D-object models in one or multiple 2D-intensity images by performing M-estimation [33] with dynamic correspondences. This tool was originally developed and programmed by Mark D. Wheeler as part of is Ph.D. Dissertation at Carnegie Mellon University [108]. However, the version used in this dissertation, include adjustments and modifications to both the software and procedures to get the models of the objects to look for in an assembly scene.

The models of the objects are described as collections of 3D points on the object's surface which often create edges when visible in intensity images; these points are further referred as *edgel generators*. The edgel generators are matched to identified 2D contrast points that compose intensity edges in the image, further referred as *intensity edgels*.

Using these correspondences, an error measure for the pose of the model is defined by measuring a 3D distance between the matching features. The error is minimized by using a robust statistical M-estimator that adjust the values for the model pose parameters. A process of pose refinement is performed by dynamically modifying the correspondences and minimizing the new error measures.

A more detailed description of the 2DTM tool, its adjustment and modeling methods are given in appendix C.

4.4.2 Modeling and Quantifying Sensing Uncertainty

To generate a measure of the expected uncertainty in the results of the localization tool, the first step is to understand its behavior based in the recognition of the most important variables and their interdependencies. A starting point was the precision graph showed in Figure 4.21. The graph presents a common uncertainty pattern when localizing 3D objects from single intensity images of the world scene. The results describe good accuracy with respect to coordinate axes that are parallel to the image plane and high uncertainty with respect to directions aligned with the optical axis of the camera.

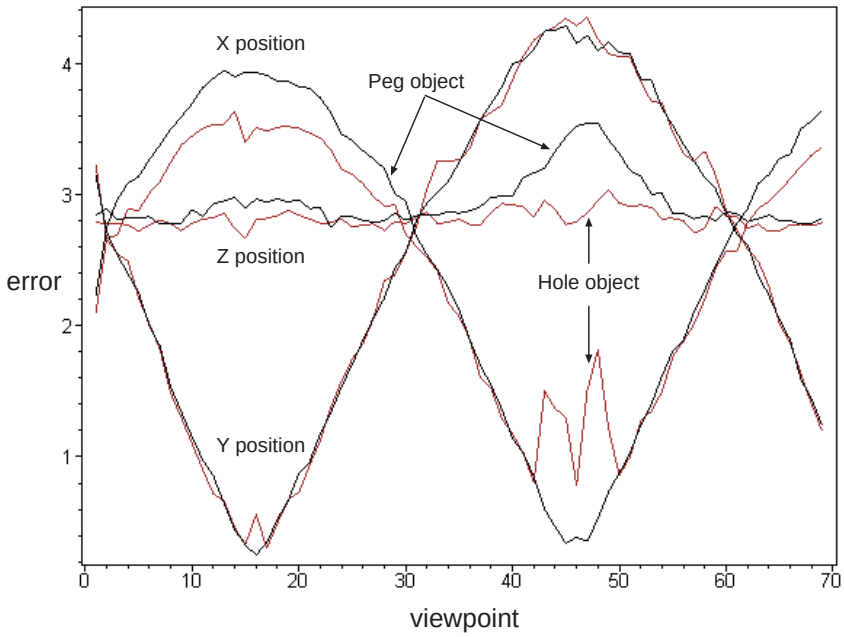
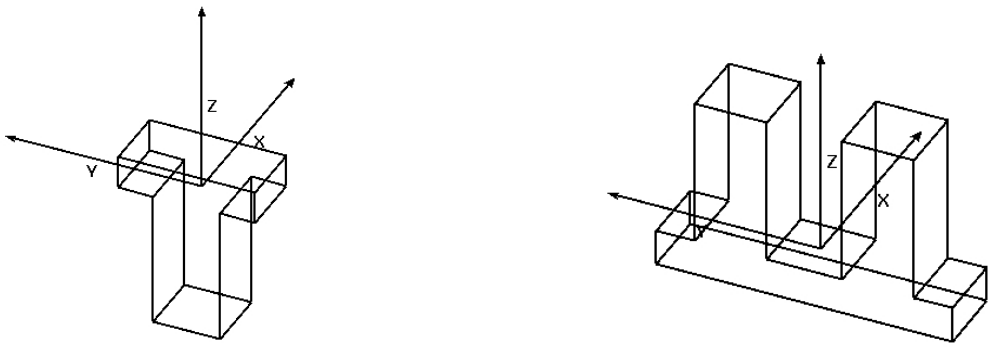


Figure 4.21: Precision in the localization of peg and hole objects.

In this respect, an extensive experimentation was performed with 2DTM to understand its behavior under different conditions and using different objects. Localization experiments were realized using real images taken with the active camera mechanism moving

the camera around the target object, and many more using a rotary table to rotate the target object with a stationary camera. Most of the experiments were realized over two real objects, a peg object and a hole object; their VANTAGE images are shown in Figure 4.22.



(a) Peg object. (b) Hole object.

Figure 4.22: Real experimental objects.

Also, the localization tool was tested with synthetic images generated directly from the CAD models of the peg, the hole, and other objects. Finally, the system was tested using simple patterns of lines to analyze the specific behavior over similar directed elements. The basic experimenting strategy was to situate a target object in a predefined pose and then apply the localization system to randomly perturbed initial pose estimates. The perturbation was realized within predefined bounded values of translational and rotational uncertainty. The uncertainty space was sampled a predefined number of times (around 200 times for each pose) and statistical descriptors were calculated.

From the results of the experiments, it was realized that, even with a localization technique utilizing robust M-estimators, the system keeps its highly-dimensional nature that makes it very difficult to understand and model analytically. However, from an analysis of its behavior, the following qualitative conclusions have been reached:

- The attributes used in the determination of the correspondences have demonstrated to be insufficient to overcome the problems emerging from the 2D nature of the sensor, as illustrated by the uncertainty pattern of Figure 4.21. The system has the ability to accurately resolve for object features that lie in a plane parallel to the image plane, but less accurately for the depth of the features. The graph was obtained from experiments realized as illustrated in Figure 4.3. The camera was located at 70 different viewpoints while rotated 180 degrees around the target objects. The stable error pattern of coordinate Z is explained because all the

viewpoints were localized at the same height. When the camera moves around the target object, what changes drastically is the orientation of the camera with respect to axes X and Y. The highest errors for a coordinate estimation, say X coordinate, happen when the axis of the other coordinate, say Y, is orthogonally aligned with the optical axis of the camera, and viceversa. Contrariwise, the smallest errors in a coordinate estimation happen when the axis of such coordinates are orthogonal to the optical axis of the camera.

- The pose refinement process realized by the system can also correct the orientation errors within planes parallel to the image plane better than the orientation errors in planes that include the center of projection of the camera.

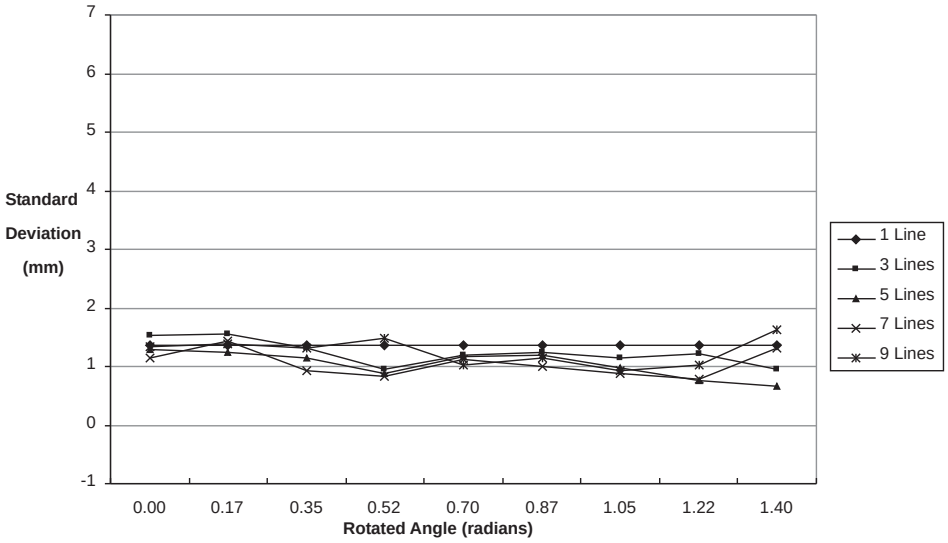


Figure 4.23: Precision of 2DTM with respect to the axis parallel to the direction of multiple lines.

- A set of experiments with simple patterns of parallel lines showed that the error with respect to an axis in a parallel direction to the lines has an static nature (Figure 4.23) while the error with respect to an axis in an orthogonal direction grows with the number of lines (Figure 4.24).

- The effect of the distance among lines depends on the size of the expected uncertainty in the initial pose estimate, showing a maximum when proportional distances are half the size of the expected error (Figure 4.25). For distances bigger than the expected uncertainty, the error tends to disappear. Bigger distances among the lines also reduces the uncertainty in depth.

- The uncertainty grows when the focal length of the camera is reduced and diminishes when the camera gets closer to the objects. The expected uncertainty grows when the error in the initial pose estimate increase (Figure 4.26).

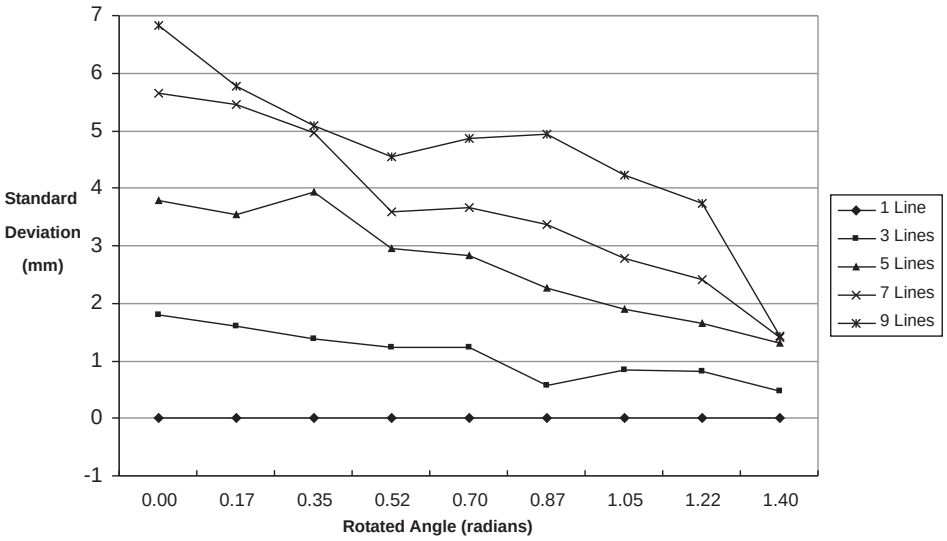


Figure 4.24: Precision of 2DTM with respect to the axis orthogonal to the direction of multiple lines.

Summarizing, the quantity of error in the final estimation depend on many factors related mainly with the size of the expected error in the initial pose estimate, the illumination conditions, the level of occlusion, the configuration of the edge features in the object geometry, the object position and attitude with respect to the camera, the focal length and distance from the camera, and the type and parameter values of the robust estimator.

The realization of most of the experiments that derived in the above qualitative analysis of the localization tool uncertainty was motivated by a search for a method to analytically compute an expected visual uncertainty for the sensor. The features considered were basically geometric features extracted from models of target objects for localization, and intrinsic and extrinsic parameters of the camera calibration model. An approach using aspects, counters of visible edgels, distances and orientation of lines, and camera calibration data, was tried, but no consistent patterns were recovered.

Since the most persistent pattern recognized was that illustrated in the graph of Figure 4.21, it was decided to use an n-dimensional ellipsoid in configuration space to approximate and represent the sensing uncertainty. The ellipsoid parameters are obtained from the statistical analysis of precision descriptors obtained by using 2DTM to localize objects in the center of a viewing sphere, with a camera in each potential viewpoint. Since the precision on the localization results depend on the pose relation between the observed object and the camera, the size and shape of the uncertainty ellipsoid changes with the sensor configuration.

The method used to get the uncertainty ellipsoid is based in principles of orthogonal

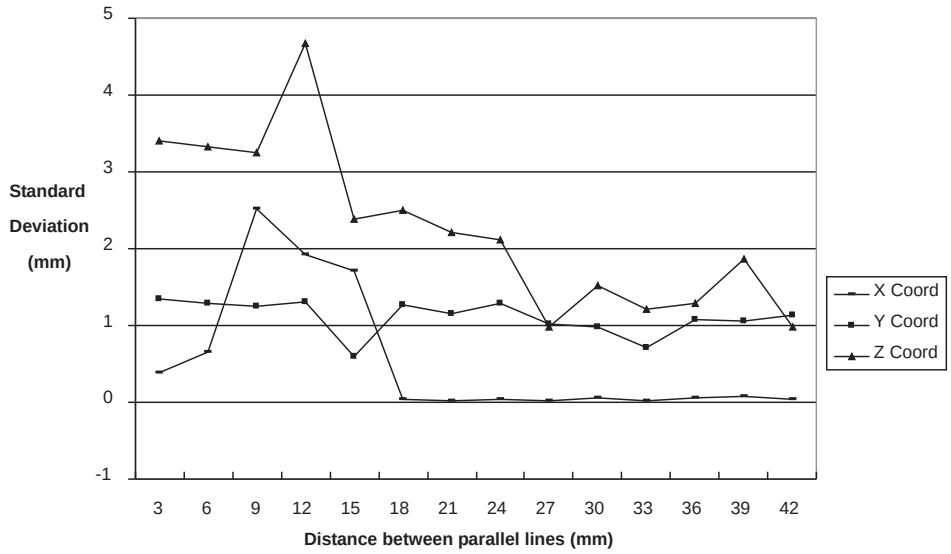


Figure 4.25: Precision of 2DTM for position estimation of 3 lines when the distance among lines changes.

regression. The axes of the ellipsoid are aligned with lines that are the best approximation to the data in the sense that the sum of the squares of the orthogonal distances to the line is minimized. The method is also known as *eigenvector fit* and is a result of a principal component analysis. Next, the foundations of the method are explained for getting the best fit line to a set of n 2D-points. The method can be easily extended to n-dimensional space.

The method establishes the fact that any line minimizing the sum of orthogonal distances must pass through the mean (or center of gravity) of the points. Then without loss of generality, it can be assumed that the set of n given points has zero mean. If every ith point (x_i, y_i) is denoted by the vector $\mathbf{p}_i$, and its perpendicular distance to the line is denoted by d_i. The line can be characterized through the origin by its unit normal vector $\mathbf{N}$. Then the method must minimize the error denoted by the sum of squares of the perpendicular distances. Since a perpendicular distance is computed as $\mathbf{N}^t\mathbf{p}_i$, the method must minimize

$$\sum_{i=1}^{n} d_i^2(\mathbf{N}) = \sum_{i=1}^{n} (\mathbf{N}^t\mathbf{p}_i)^2 = \mathbf{N}^t \sum_{i=1}^{n} (p_i p_i^t)\mathbf{N} \tag{4.43}$$

The symmetric matrix

$$\mathbf{S} = \sum_{i=1}^{n} (p_i p_i^t) \tag{4.44}$$

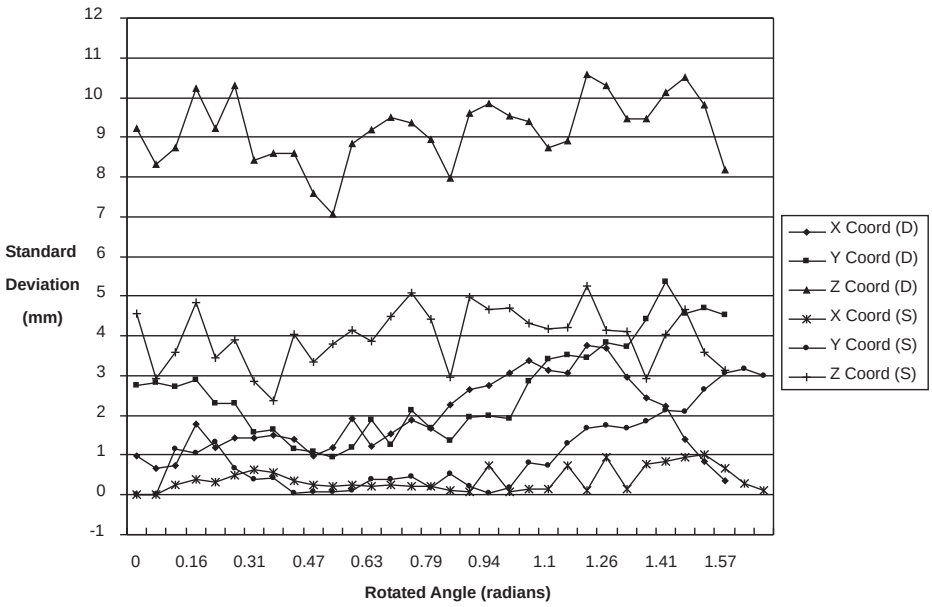


Figure 4.26: Precision of 2DTM when the initial error is duplicated.

is the *scatter matrix* of the n given points. The best line is therefore characterized by the unit normal vector $\mathbf{N}$ that minimizes $\mathbf{N}^t\mathbf{S}\mathbf{N}$. This quadratic form is minimized by taking $\mathbf{N}$ to be the eigenvector of $\mathbf{S}$ associated with the smallest eigenvalue. Since distinct eigenvectors of symmetric matrix are orthogonal, the best fitting line is in the direction of the principal eigenvector of $\mathbf{S}$, that is, the eigenvector associated with the largest eigenvalue.

The same strategy is used to obtain the rest of the principal axes of the ellipsoid as the eigenvectors associated with the rest of the eigenvalues. Additionally, the radii of the ellipsoid are calculated as scaled values of the standard deviations of the error points in the directions of the principal axes.

The algorithm for getting the description of the uncertainty ellipsoid, then, is the following:

1. Perform a series of n object localization experiments over a target object to get n estimated poses from a viewpoint.

2. Subtract the true pose of each estimated pose to get a set of n error vectors.

3. Compute the mean of each vector element for the set of n error vectors.

4. Standardize the error vectors by subtracting the means of each element.

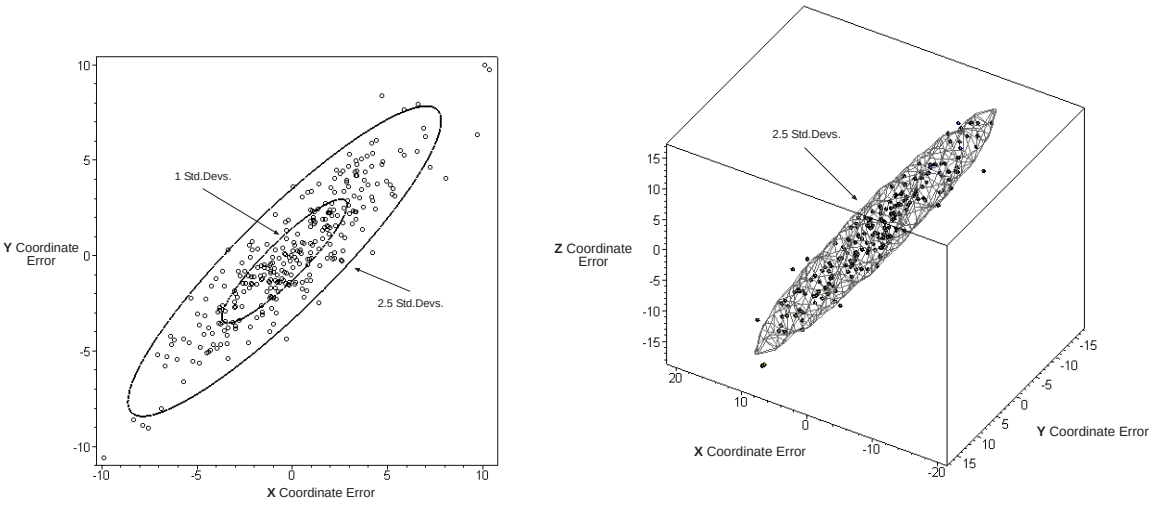
5. Compute the scatter matrix with the set of standardized error vectors.

6. Find the eigenvectors of the scatter matrix.

7. Rotate each standardized error vector to align the basis described by the eigenvectors of their scatter matrix with the coordinate frame of the target object.

8. Compute the standard deviation with respect to each axis.

The same process is repeated for each different viewpoint. The description data required for the computation of the criterion used to order the viewpoints are the principal axes of the ellipsoid (the eigenvectors) and the standard deviations. The standard deviations are used as scale factors for defining the size of the ellipsoid. These scale factors are also used to describe a bounding box that contains the uncertainty ellipsoid. This bounding box is used for the computation of the Psp values.



(a) 2D fitted ellipses.

(b) 3D fitted ellipsoid.

Figure 4.27: Uncertainty ellipsoids.

Figures 4.27(a)-(b) presents examples of 2D and 3D ellipsoids fitted to sets of 2D and 3D error vectors, by following the proposed algorithm.

To align a 2D tilted ellipsoid (an ellipse) with the coordinate frame of the target object, the error vectors must be rotated an angle computed as

$$\theta = \tan^{-1}\left(\frac{e_y}{e_x}\right) \tag{4.45}$$

where e_x and e_y are the X and Y coordinates of the unit eigenvector associated with the largest eigenvalue.

The form of the tilted uncertainty ellipse is

$$\frac{(x\,\cos(\theta) - y\,\sin(\theta))^2}{k\,\sigma_x^2} + \frac{(x\,\sin(\theta) + y\,\cos(\theta))^2}{k\,\sigma_y^2} = 1 \tag{4.46}$$

where σ_x and σ_y are the standard deviation with respect to each coordinate of the aligned ellipse, and k is a scale factor. Experimentally was observed that a $k = 2.5$ gave good results.

Aligning a 3D ellipsoid is more difficult. It has to be done in two steps: first, the axis described by the eigenvector associated with the largest eigenvalue is aligned with a coordinate axis of the target object, say the Z axis, with a sequence of a rotation with respect to X axis followed by a rotation with respect to the Y axis. And second, complete the alignment of the other two axes through a rotation with respect to Z axis. The rotation angles are computed as

$$\begin{aligned} \alpha &= \tan^{-1}\left(\frac{e_y}{e_z}\right) \\ \beta &= -\tan^{-1}\left(\frac{e_x}{\sqrt{e_y^2+e_z^2}}\right) \\ \gamma &= -\tan^{-1}\left(\frac{e'_{2y}}{e'_{2x}}\right) \end{aligned} \tag{4.47}$$

where α, β, and γ are the X, Y, and Z rotation angles, respectively; e_x, e_y, and e_z are the X, Y, and Z coordinates of the eigenvector associated with the largest eigenvalue; and e'_{2x} and e'_{2y} are the rotated X and Y coordinates of the eigenvector associated with the second largest eigenvalue. The rotated coordinates are computed by multiplying the coordinates with the rotation matrix $\mathbf{R}_y(\beta)\,\mathbf{R}_x(\alpha)$.

The equation of the tilted 3D ellipsoid is

$$\begin{gathered} \frac{(\cos(\gamma)\,\cos(\beta)\,x + (-\sin(\gamma)\,\cos(\alpha) + \cos(\gamma)\,\sin(\beta)\,\sin(\alpha))\,y + (\sin(\gamma)\,\sin(\alpha) + \cos(\gamma)\,\sin(\beta)\,\cos(\alpha))\,z)^2}{k\,\sigma_x^2} + \\ \frac{(\sin(\gamma)\,\cos(\beta)\,x + (\cos(\gamma)\,\cos(\alpha) + \sin(\gamma)\,\sin(\beta)\,\sin(\alpha))\,y + (-\cos(\gamma)\,\sin(\alpha) + \sin(\gamma)\,\sin(\beta)\,\cos(\alpha))\,z)^2}{k\,\sigma_y^2} + \\ \frac{(-\sin(\beta)\,x + \cos(\beta)\,\sin(\alpha)\,y + \cos(\beta)\,\cos(\alpha)\,z)^2}{k\,\sigma_z^2} = 1 \end{gathered} \tag{4.48}$$

where σ_x, σ_y, and σ_z are the standard deviations with respect X, Y, and Z coordinates of the aligned ellipsoid.

In the case of the 6-dimensional ellipsoid, the solution used was to construct two 3D ellipsoids, one for translation and one for rotation.

4.5 Predicted Success Probability

The decision for the best viewpoints to actually use during the visual verification of the assembly conditions has to wait until the assembly setup is ready to perform the assembly. The workspace of the specific sensing camera mechanism to use for the vision tasks is limited by a set of kinematic constraints. This limitation eliminates some of the modeled viewpoints considered during the sensor planning stage.

A Psp value express the goodness of a particular viewpoint for performing an object localization task to verify critical alignment relations of its target object with respect to another reference in its assembly environment.

A program to implement the proposed strategy for the evaluation of visual sensing strategies was developed. In this section, the strategy is tested for a peg-in-hole operation. For this experiment, the assembly conditions were simplified by assuming that the exact position of the hole object is known. Then for the realized experiments, the goal was reduced to determine the actual pose of the peg. In the case that the hole pose were unknown, before determining the adjustment of the peg's pose, the pose of the hole object had to be deduced from observation.

4.5.1 Peg-in-Hole Experiments

A peg-in-hole robot operation was performed using a t-shape peg object with an inserting square cross-section. The peg is inserted into square holes of different dimensions.

Experimental Process

- Getting the correct inserting parameters:
  1. Define a pose for the hole object and fix it to the work table.
  2. Move the peg object to its correct insertion pose. This pose is where the peg is positioned and aligned in such a way that it can be successfully inserted into the hole object by a simple straight descending motion.
  3. Store the hole and the peg poses.
- Computing the sensing uncertainty descriptors:
  1. Select the camera poses to be used during the experiments.
  2. Calibrate the camera in each selected pose.
  3. Move the peg object to its correct insertion pose.
  4. Take an image of the scene from each camera pose.

5. Compute the uncertainty descriptors for each camera pose by localizing the peg object in the scene image using the localization tool.

- Performing the peg-in-hole experiments (for each selected camera pose):

1. Move the camera to the selected pose.
2. Repeat the following n times:
   (a) Move the peg to the correct insertion pose.
   (b) Perturb randomly the peg's pose. The new pose is obtained by computing random bounded values for translation and rotation parameters and then applying them to the correct pose. The bounding limits are decided based on an expected initial pose error.
   (c) Take an image of the new scene.
   (d) Find the peg in the scene image with the localization tool.
   (e) Compute the difference between the expected (correct) and observed (sensed) poses.
   (f) Compute and apply a corrective motion.
   (g) Perform the descending insertion operation as a guarded motion. The task is perform using velocity control and sensing continuously force data. As soon as a change in the force pattern is detected, the insertion task is aborted.
   (h) Annotate whether the insertion operation succeeded or failed.
3. Compute and store the success ration for the current camera pose.

Calibration of the Assembly Relations

The correct execution of the above experiments depend on a quantified knowledge of the relations among the experimental elements: robots, camera, parts to assemble, and the assembly reference (world). Then a set of calibration processes must be performed. Each calibration giving values to a parameterized relation.

The tasks are established with respect to an special assembly reference known as the world coordinate system. However, since their execution is performed through commanded motions of the assembly robot, their relations have to be parameterized and the value of the parameters obtained by calibration.

To perform the assembly, a revolute robot of six degrees of freedom, a FAROT DD Fujitsu robot arm was used. This is a quite precise robot that allows commanding motions in several modalities with respect to several coordinate references. It was decided to command the robot's motions by six parameters $(x, y, z, \alpha, \beta, \gamma)$ that define the absolute

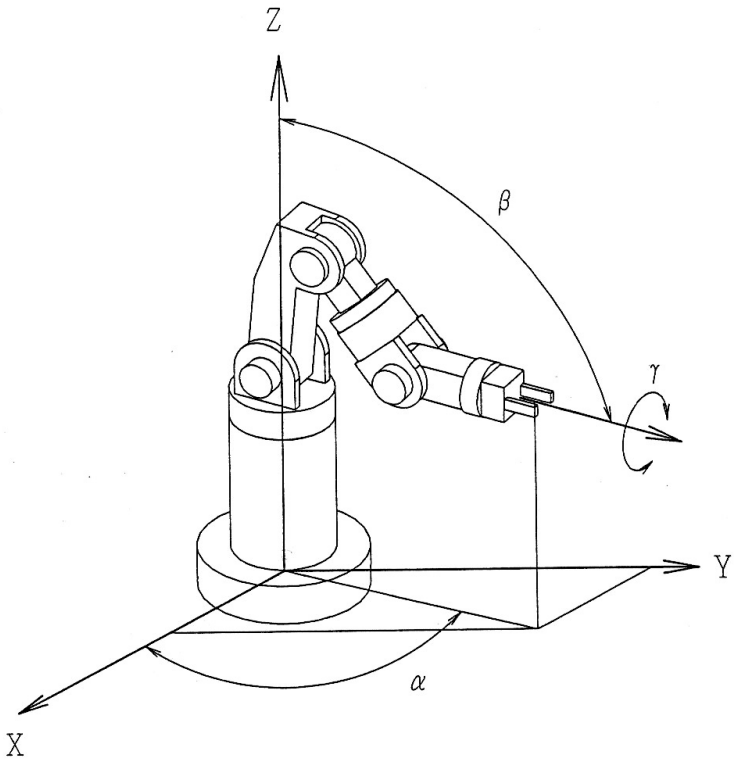


Figure 4.28: FAROT DD Fujitsu arm and its base coordinate system.

position and orientation of the robot hand (a parallel jab gripper) with respect to the robot local coordinate frame located on its base (see Figure 4.28).

The robot transformation is computed as

$$\mathbf{p}_r = \mathbf{T}_r \, \mathbf{p}_g = \mathbf{R}_r \, \mathbf{p}_g + \mathbf{t}_r \tag{4.49}$$

where $\mathbf{R}_r$ is a rotation matrix that re-orients the local coordinate system of the gripper with respect to the base frame of the robot, $\mathbf{p}_g$ is a 3D point with coordinates defined with respect to the local gripper frame, and $\mathbf{t}_r$ is a 3D vector that defines the final position of the hand.

When the hand is in its home position $(0, 0, 0, 0, 0, 0)$ the axis for γ rotation is assumed to be perfectly aligned with the Z axis of the robot frame. Then $\mathbf{R}_r$ can be calculated as

$$\mathbf{R}_r = \mathbf{R}_z(\alpha) \, \mathbf{R}_y(\beta) \, \mathbf{R}_z(\gamma) \tag{4.50}$$

which gives the matrix

$$\begin{bmatrix} \cos(\alpha)\cos(\beta)\cos(\gamma) - \sin(\alpha)\sin(\gamma) & -\cos(\alpha)\cos(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\sin(beta) \\ \sin(\alpha)\cos(\beta)\cos(\gamma) + \cos(\alpha)\sin(\gamma) & \cos(\alpha)\sin(\gamma) - \sin(\alpha)\cos(\beta)\sin(\gamma) & \sin(\alpha)\sin(beta) \\ -\sin(\beta)\cos(\gamma) & \sin(\beta)\sin(\gamma) & \cos(\beta) \end{bmatrix}. \tag{4.51}$$

From the above matrix, the rotation angles can be extracted as

$$\alpha = \arctan 2(r_{12},\ r_{02}) \tag{4.52a}$$
$$\beta = \arctan 2(\sqrt{r_{02}^2 + r_{12}^2},\ r_{22}) \tag{4.52b}$$
$$\gamma = \arctan 2(r_{21},\ -r_{20}). \tag{4.52c}$$

Now the problem is how to parameterize the important relations to transform required object poses in world coordinates to robot commands. To do this in general, it is needed to determine the following relations:

- World to assembly robot transformation ($\mathbf{T}_{wr}$),
- Gripper to robot transformation ($\mathbf{T}_{gr}$),
- Object to world transformation ($\mathbf{T}_{ow}$), and
- Object to gripper transformation ($\mathbf{T}_{og}$).

Now, if some relations can be fixed, it is possible to reduce at least some relations. For example, for these experiments, the peg-object was fixed to the gripper and, then, the requirements to compute $\mathbf{T}_{gr}$ and $\mathbf{T}_{og}$ were reduced.

Object to World Transformation

The $\mathbf{T}_{ow}$ transformation is obtained directly from the required object pose. In the performed experiments, the object pose which were determined by the localization tool is represented by a seven elements vector $\mathbf{p}_e = (x, y, z, u, v, w, \theta)^T$ where the first three $\mathbf{p} = (x, y, z)^T$ defines its world position, $\mathbf{r} = (u, v, w)^T$ a unit vector representing a rotation axis, and θ a rotation angle defines its world attitude.

In this case the conversion from the pose parameters to the homogeneous transform matrix $\mathbf{T}_{ow}$ is performed in three steps:

1. Its attitude representation is converted into a unit quaternion $\mathbf{q}$ [29] as

$$\mathbf{q} = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ \mathbf{r}\,\sin(\frac{\theta}{2}) \end{bmatrix} \tag{4.53}$$

2. The unit quaternion $\mathbf{q} = (q_s, q_u, q_v, q_w)^T$ is converted into matrix form as

$$\mathbf{R}_{og} = \begin{bmatrix} q_s^2 + q_u^2 - q_v^2 - q_w^2 & 2\,(q_u\,q_v - q_s\,q_w) & 2\,(q_u\,q_w + q_s\,q_v) \\ 2\,(q_u\,q_v + q_s\,q_w) & q_s^2 - q_u^2 - q_v^2 - q_w^2 & 2\,(q_v\,q_w - q_s\,q_u) \\ 2\,(q_u\,q_w - q_s\,q_v) & 2\,(q_v\,q_w + q_s\,q_u) & q_s^2 + q_u^2 - q_v^2 - q_w^2 \end{bmatrix} \tag{4.54}$$

3. $\mathbf{T}_{og}$ is composed as

$$\mathbf{T}_{og} = \begin{bmatrix} \mathbf{R}_{og} & \mathbf{t}_{og} \\ \hat{0} & 1 \end{bmatrix} \tag{4.55}$$

   where $\mathbf{t}_{og}$ is the 3D column vector containing the object position $\mathbf{p}$.

World to Robot Transformation

The $\mathbf{T}_{wr}$ relation is calibrated by localizing the peg object in different poses with respect to the world reference and using the parameters commanded to the robot for moving the peg.

To perform the peg localization, an implicit stereo approach was used as follows:

1. Two camera poses were selected and calibrated with respect to the world coordinate system.

2. A set of robot motions were selected and stored taking care that the peg is inside the visible region of both cameras.

3. An image of the peg was taken for each pose from each camera pose.

4. The localization tool was used to localize the peg in each par of images and the pose was stored.

A set of points on each standard axis of the object were selected for calibration. The world coordinates for each point are computed as:

$$\mathbf{p}_{wi} = \mathbf{T}_{ow}\,\mathbf{p}_{oi} \tag{4.56}$$

where $\mathbf{p}_{wi}$ and $\mathbf{p}_{oi}$ are the ith corresponding points in world and object coordinates, respectively.

The world to robot relation is modeled as usual by

$$\mathbf{T}_{wr}\,\mathbf{p}_{wi} = \mathbf{R}_{wr}\,\mathbf{p}_{wi} + \mathbf{t}_{wr} = \mathbf{p}_{ri} \tag{4.57}$$

where $\mathbf{R}_{wr}$ is a rotation matrix, $\mathbf{t}_{wr}$ is a translation vector, and $\mathbf{p}_{ri}$ is the ith point in robot coordinates.

Since the values for the points with respect to the robot reference are not known, one of the object poses (e.g. the first one) can be used as auxiliary to eliminate the unknowns in which it is not interested in, the robot points in this case.

The n object poses and the selected m object points generates a system of $m\,(n-1)$ equations of the form

$$\mathbf{R}_{ri}^{-1}\,(\mathbf{R}_{wr}\,\mathbf{p}_{wi} + \mathbf{t}_{wr} - \mathbf{t}_{ri}) = \mathbf{R}_{r0}^{-1}\,(\mathbf{R}_{wr}\,\mathbf{p}_{w0} + \mathbf{t}_{wr} - \mathbf{t}_{r0}) \tag{4.58}$$

where $\mathbf{p}_{wi}$ and $\mathbf{p}_{w0}$ are the corresponding object points in world coordinates in the (i+1)th and first object poses; and $\mathbf{R}_{ri}$, $\mathbf{t}_{ri}$, $\mathbf{R}_{r0}$, and $\mathbf{t}_{r0}$ are the rotation matrices and translation vectors obtained from the commanded robot motions for the (i+1)th and first object poses, respectively.

The system is solved by minimizing the squared error calculated from the distance between the resultant points in both sides of the equations.

Experimental Results and Discussion

A generate-and-test approach was used to perform the experiments. A set of potential viewpoints were explicitly selected and the strategy was applied. Figure 4.29 presents the success ratios for inserting the peg object in holes of increasing sizes. This values were obtained by performing the task with real objects. The success ratio is computed by actually counting the number of times that the task succeed. A task is considered successful if the peg is displaced a predefined distance without detecting a force pattern associated with a contact in the opposite direction of the insertion. The graph shows as is expected that a bigger hole allows better success ratios.

The Figure 4.30 depicts the Psp values computed by constructing the uncertainty ellipsoids from the actual variability observed in the peg pose during the real experiments. Two and a half standard deviations were used to compute the radii of the principal axes describing the uncertainty ellipsoid. Then these were used to compute the Psp values

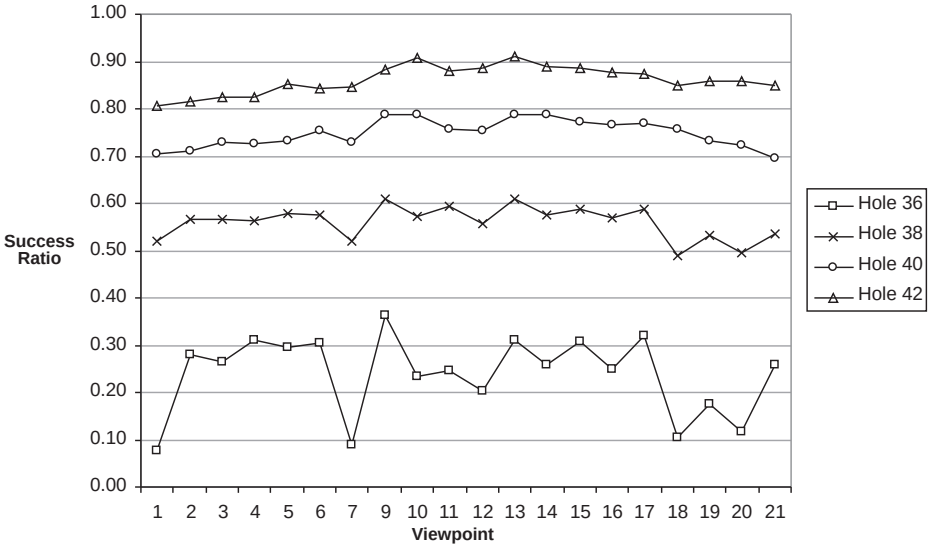


Figure 4.29: Success Ratio from Real Experiments.

shown in the graph.

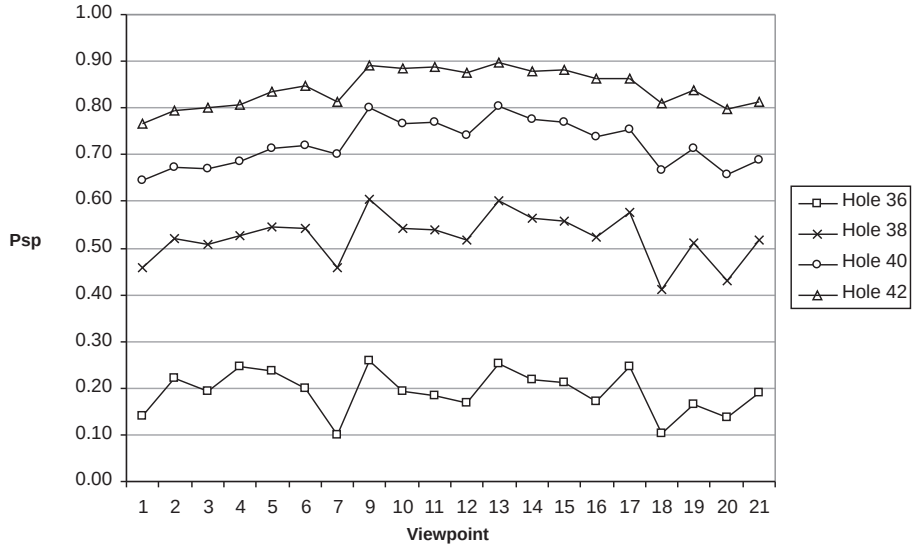


Figure 4.30: *Psp* from Real Experiments by using 2.5 Standard Deviations.

As it can be clearly appreciated the proposed *Psp* measure presents a similar pattern to that of the real success ratio with approximated magnitudes. Even in the cases where the magnitudes of the experimental and theoretic measures differ, the important relative relations among camera poses, which will be used to order the viewpoints, are maintained.

The same experiments were repeated using synthetic images instead of real intensity

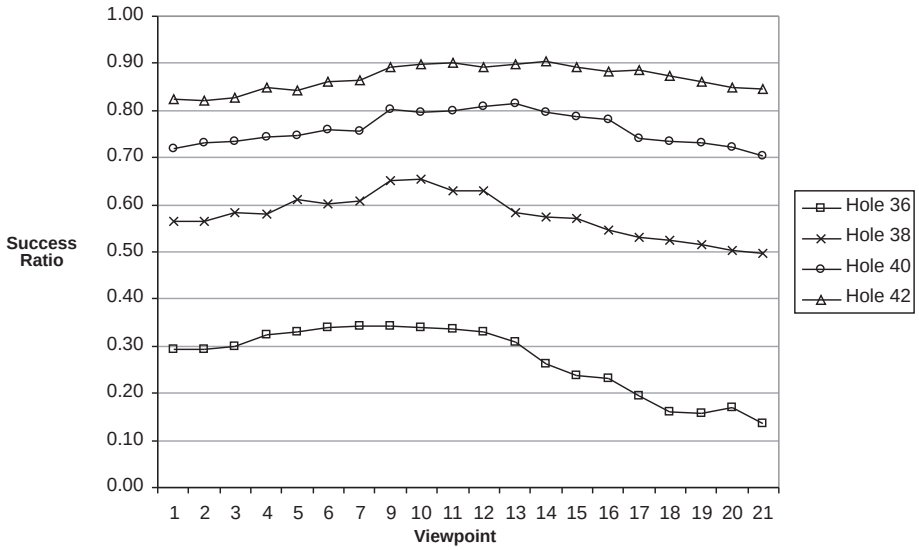


Figure 4.31: Success Ratio from Experiments with Synthetic Images.

images. Figures 4.31 and 4.32 present the success ratio and Psp values obtained in this cases. Again, the Psp measures show a similar behavior to the computed success ratio, but this time computed by simulating the execution of the insertion task. Though the synthetic results show more relaxed patterns, the tendencies are comparable to those of the real case. This even when strong assumptions where made to create the synthetic images, which considered perfect illumination, absence of noise, reduced occlusion, and empty surroundings (background).

Finally, Figure 4.33 presents the results of similar insertion experiments performed by placing a peg object in the center of a tessellated viewing sphere. The viewpoints were located in the center of each triangle and oriented through the center of the sphere. Four sizes of square holes were analyzed. The uncertainty ellipsoids were computed from synthetic images of the peg object, and the insertion experiments were simulated.

The Psp results are shown in a gray-scale coded representation in Figures 4.33(b)-(e), where lighter colors are used to represent higher predicted success probabilities. Perfect white triangles represent viewpoints where its uncertainty region is completely inside of the error tolerance region of the task.

Even under the simulation assumptions, as would be expected in real situations, the best viewpoints start in positions orthogonal to the insertion plane and the Psp values decrease when the angle between the viewing direction and the inserting direction grows.

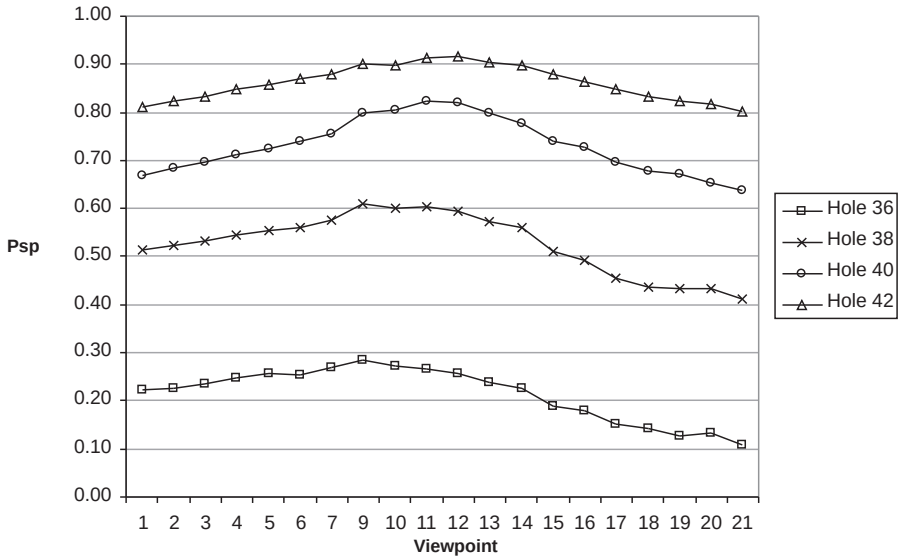
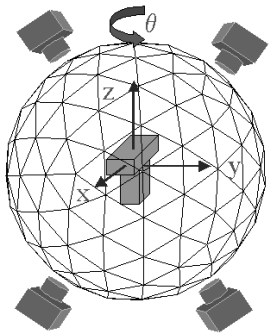
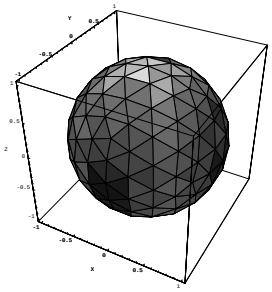


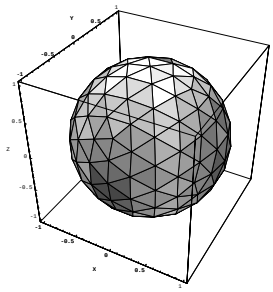
Figure 4.32: Psp from Experiments with Synthetic Images by using 2.5 Standard Deviations.



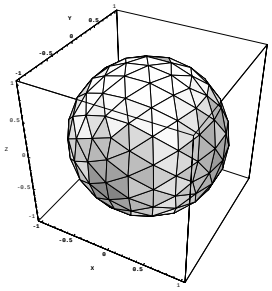
(a) Viewing Sphere.



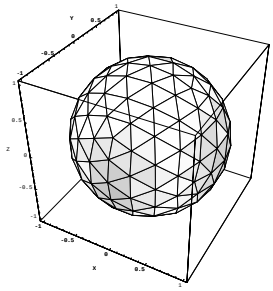
(b) Hole: 36x36 mm.



(c) Hole: 38x38 mm.



(d) Hole: 40x40 mm.



(e) Hole: 42x42 mm.

Figure 4.33: Predicted Success Probability for Peg-in-Hole Task.

Chapter 6

Conclusions

This thesis has investigated several issues related to the use of geometric information about the models of component objects of assemblies and the process of contact formation among such objects for tackling the automatic planning of sensing strategies.

The studies and experiments conducted during this research have led to the development of novel methods for enabling robots to detect critical errors and deviations from a nominal assembly plan during its execution. The errors are detected before they cause failure of assembly operations, when the objects that will cause a problem are manipulated. Having control over these objects, adjustment actions could be commanded expecting to correct the errors.

Chapters 2 and 3 propose a sensing analyzer that determine which tasks require of using force and vision feedback operations. It also determines what is the critical information that has to be collected and which objects contain such information. Chapter 4 proposes a sensor planner that evaluates potential viewpoints for the camera to be used for localizing the objects that contain critical information for the assembly tasks. The criteria used is computed partially by an analytical method and partially by an empirical method. This order is one of the main factors for constructing a full visual sensing strategy for robotic assembly. Chapter 5 presents an active camera mechanism that was developed to perform visual localization tasks and a method that is proposed to obtain the actual preventive visual strategy for the complete assembly plan.

6.1 Contributions

The main contributions of this thesis are the following:

- A method to systematically deduce assembly skill primitives and force compliance skills requirements for assembly tasks.

- Introduction of a graph representation that codes the sensing requirements of an assembly plan.
- A method to propagate direct and indirect translational and rotational dependencies among environmental objects.
- A method to analytically describe a region of tolerated error for assembly tasks.
- An algorithm to quantify and represent the uncertainty of a template matching localization tool that works on intensity-images.
- A method to evaluate potential viewpoints to locate the camera for localization tasks.
- Development of an active camera mechanism for visual verification tasks.
- A method to construct preventive vision strategies for a complete assembly plan.

Below, each contribution is expanded in more detail.

6.1.1 Automatic Deduction of Skills

In this thesis, a method is proposed for systematically deduce force compliance skills required for monitoring and controlling the execution of tasks that involve contacts between the object manipulated by the robot arm in the task and the objects that conform its direct environmental configuration.

First, the method recognizes the assembly tasks from transitions in the contact relations of the manipulated object. For this purpose each DOF of the manipulated object is cataloged as maintaining DOF, detaching DOF, or constraining DOF. Then, it identify and associate an assembly skill primitives with each DOF in accordance with a possible change in their constraining conditions. Knowing the assembly skill primitives, the force compliance skills are extracted from the mapping table 2.1.

Even though this research does not deals with assembly task planning, the deduction and use of four assembly skill primitives obviated the force compliance skill deduction and the deduction of some preventive vision requirements. Whereas the force compliance skills set was reduced to a minimum of three, the selected skills were considered enough for the type of tasks expected in the nominal assembly plans and for the purpose of reducing the potential critical dimensions for vision. However, they are associated with each DOF of the manipulated object, which means that for some tasks several skills have to be performed concurrently. Such combination of tasks is what makes force control difficult to implement, and that is the reason why the force compliance skills could be

further decomposed into multiple force compliance primitives. Implementing the force compliance skills was not part of the goals for this thesis, however, a basic facility was implemented to perform the real experiments for getting the success ratio data in the sensor planning module.

6.1.2 Representation of Sensing Requirements

In this thesis, the ICdg representation has been introduced to express direct and indirect critical dependencies among assembly elements, caused by insertions and contacts between the manipulated objects of the assembly operations and their environments.

The ICdg is constructed while analyzing the contact formation process. The addition of new constraint dependencies to the ICdg is fired by the recognition of insertion operations and multiple-contact operations. Then, direct constraint dependencies are propagated among the environmental objects. The final ICdg codes the complete set of critical alignment constraints, and then expresses the force and vision sensing requirements for the analyzed assembly plan.

This structure simplified the deduction of the real critical dimensions of a task by making explicit the true relevance of certain alignment relations between objects in the environment. It is also concise and easy to visualize. The most important information is stored in its dependency arcs. Such information not only indicates which tasks need of some type of sensing, it also contain the data required to construct the criteria for evaluating the sensor configurations. The ICdg is the central data structure also during sensing execution because it describe the dependencies in relative terms. If an object, on which some other objects depend, does not finish in its absolute prescribed pose, all the other objects depending in it should be adjusted accordingly.

The utility of the ICdg was demonstrated during the programming of the sensing analyzer by simplifying the codification and debugging process and during the experiments as an instrument for verifying the correctness of the results.

6.1.3 Propagation Scheme of Critical Dependencies

In this thesis, a method is proposed to propagate alignment constraint dependencies among the objects that conform the assembly environment. The method extends the strategy proposed in [64] which limited its preventive vision strategy to verify the configuration of inserted objects.

Recognizing the importance of having a correct environment configuration to succeed in the execution of a task that involve multiple objects, the propagation of critical

dependencies allow to anticipate potential problems that could irremediable affect the success execution of subsequent assembly operations.

This propagation scheme represents the heart of this dissertation work because it provides the basis for the rest of the contributions and work. The main purpose of this thesis is to help in the successful execution of robotic assemblies by automatically planning sensing strategies to prevent critical errors. Then, the propagation of critical dependencies makes explicit the most important tasks in an assembly plan. This helps to redefine its success criteria not only in terms of its current effects in a single operation but in terms of its global effects in the success of the plan execution.

The results of this propagation process can not only be used to prescribe sensing requirements, it also can be used to judge when is recommendable to use some support devices for assembly and even when to perform assembly re-planning. The idea is to reduce the sensing requirements, specially of preventive vision, and assembly plans that require a lot of visual verification tasks denote re-sequencing opportunities.

The propagation method was extensively evaluated against test cases presented to its computer implementation. Many of the cases were purposely designed to confront the system with complex interrelationships even for humans. The system behave correctly in all the test cases. Inclusive, in some occasion it finished with unexpected results, but after a deeper analysis it was realized that its decisions where correct.

6.1.4 Analytical Description of Regions of Tolerated Error

A method has been proposed to analytically deduce the descriptions of inequalities that implicitly describe a region of tolerated error. When an error displaces a manipulated object to a pose that still is inside of this region, the task is expected to succeed.

The proposed method computes sets of inequality points that are used to describe critical constraining relations for analyzed objects. It also recognize, for the case of insertion tasks, and define, for the case of environmental object configurations, constraining references that together with the inequality points are used to obtain the critical inequalities that implicitly describe the task tolerance region.

Though uncertainty affects all visual parameters, the constraining effect of contacts and insertion configurations is used to concentrate only in the effects of uncertainty in the critical dimensions of a task because this are related to the DOF that have to be adjusted.

A detailed analysis was realized for 2D-assembly (planar case) and 3D-assembly. The analysis was facilitated by decomposing the constraining effect of multiple constraining conditions to simple constrained plane to constraining plane cases. General Inequality

formulas were developed and special cases were reviewed. The special cases helped to test the consistency of, in some cases, very complex formulas.

6.1.5 Modeling and Quantification of Sensing Uncertainty

An algorithm that implements an empirical method to determine the form and orientation of six-dimensional ellipsoids is proposed to model and quantify the uncertainty of a tool that localize 3D objects from single intensity-images of the assembly scene.

The sensing strategy use a robust technique that works much better for free-form objects than for simple polyhedral objects, as those involved in the present work. An orthogonal regression method is used to determine the orientation of a commonly tilted ellipsoid, and a statistical precision estimator is used as a scaling factor to describe the size of its principal axes.

Extensive experimentation over real images of test objects obtained from a testbed including an active camera mechanism, a computer-controllable rotary table, and a computer-controllable reconfigurable illumination environment; and over simulated images of multiple objects and patterns of lines was realized. The experimental results showed that the problem of uncertainty modeling for the selected sensor and sensing strategies is highly dimensional. As consequence it was decided to approximate its shape and size using an empirical algorithm. The error pattern described by the object pose refinement experiments supported the decision.

6.1.6 Ordering of Camera's Viewpoints

A method is proposed to compute a criterion – the predicted success probability or Psp – for ordering a set of potential viewpoints in which a camera can be located to perform object localization tasks. The measure is the result of a combination of the region of error tolerance of a task and the uncertainty ellipsoid associated with a particular viewpoint located in the center of a tessel of a discretized viewing sphere.

The method segments the uncertainty ellipsoid into homogeneous elements converting the computation of the Psp in a counting problem. The method basically computes the portion of uncertainty ellipsoid that falls inside the region of tolerated error for a task.

For the experiments, the number of segments for DOF was selected arbitrarily as a constant, and only one point is selected for each elementary partition. This specific point is tested to fall inside of the uncertainty ellipsoid and in the unconstrained subspace described by the set of constraining inequalities. If the point pass this test, it is counted, consequently increasing the Psp for the viewpoint. The complexity of the task tolerance

region description and the high dimension of the uncertainty ellipsoid supported the decision for the method.

Real experiments of assembly operations were realized to test the Psp criterion. The ordering of viewpoints described by the success ratio obtained in the experiments seems agreed with the ordering of viewpoints described by the Psp values. Obviously, the exact Psp value differ from the exact success ratio because it is obtained by an approximated method that additionally depends in a scaling factor statistically decided. However, the important product is the relative ordering among viewpoints. Experiments with simulated images were also realized over each tessel of the discretized viewing sphere and intuition agrees with the ordering results.

6.1.7 An Active Camera Mechanism

An active vision mechanism is proposed to perform visual verification tasks. This mechanism allows the camera move around the assembly scene to recollect visual information required to prevent errors and deviations that could cause failure in the execution of assembly operations. The active camera was also used during the experimentation phase.

A method is proposed to calibrate the different elements of the active camera and move the camera to commanded poses. A calibration model for the active camera was developed accordingly.

The features of each component of the active mechanism has to be modeled and calibrated to reduce the error in locating the optical center of the camera over a planned viewpoint. Object pose estimation errors will be directly affected by the active camera calibration model, its quantification, the uncertainty on the actions of the robot that translate the camera and the uncertainty on the pan/tilt mechanism. Exact analytical formulations were developed to adjust the calibration model and pose of the active camera; however, the approximated nature of the calibration models added to errors in the tasks.

It is also important to have in mind that errors caused by approximated camera calibration models is compensated by changes in values of intrinsic parameters, which means that a change only in the camera pose will need of adjustment on both the extrinsic parameters and the intrinsic parameters. This was evident in the experiments performed. Also from the experiments, it was realized that a more elaborated model of the pan/tilt mechanism result in more complex adjustment formulations. And for the camera utilized, this complication was not justified by the improvement in the accuracy of the camera.

6.1.8 Computation of Preventive Vision Strategies

Finally, a method is proposed to construct a complete visual strategy for an assembly plan. This method decide the specific sequence of viewpoints to use for localizing the objects that were specified by the visual sensing analyzer.

The proposed method transforms the problem of deciding a sequence of camera motions into a multi-objective optimization problem that is solved in two phases: a local phase that reduce the set of potential viewpoints to small sets of viewpoints with the best Psp values of the kinematically feasible viewpoints for the active camera; and a global phase that decides a single viewpoint for each object in a task and then stitch them together to form the visual sensing strategy for the assembly plan. Additionally to the Psp values, the multi-objective optimization method try to minimize a measure of the level of occlusion from the considered viewpoints and the distance that the camera has to be moved. A dynamic programming approach was selected to perform the optimization process.

6.2 Future Directions

We conclude with a discussion of the limitations and suggested improvements of the work presented in this dissertation.

6.2.1 Sensing Planning

The limitations of the sensing planner proposed in this dissertation are related with the assumptions about the type of objects, the type of contacts, and the type of nominal assembly plans. The objects are assumed to be rigid polyhedral objects participating in face contact relations, and the nominal assembly plans are described as single sequences of mating operations involving only one manipulated object at the time.

An obvious extension in this respect is one that allows curved objects. To extend the approach to curved objects it is recommendable to extend the type of allowed contact configurations. The usual strategy is to describe the contacts by sets of contact points. This will require to increase the number of contact-state relations and to consider possible singular configurations as those described in [99]. The number and type of contact-state transitions should increase accordingly.

Today, there is a tendency through multi-agent collaborative systems where several autonomous entities perform tasks concurrently, which initially would seem to require of a much more complex strategy than the strategy described in this document. How-

ever, if some of the obvious potential interference problems are resolved, the approach could be applied as described. Additional complexity sources could come from the inclusion of several active cameras, grasping stability problems, and model uncertainty in subassemblies.

6.2.2 Sensor Planning

The limitations of the sensor planning approach is also determined by the assumptions in the geometry of the objects for the computation of the task tolerance region, and the type of sensor and sensing strategy in the computation of the sensing uncertainty region.

To extend the approach to include curved objects, a new analysis and formulation of constraint inequalities should be performed; however, since the geometry of mechanic parts tends to standardized shapes, some specific formulations could be easily added to describe regions composed by usual curved objects with common curved shapes, e.g. cylinders.

Something that was not realized during this research was an study of the effect of using viewpoints different from those selected in the tessellated viewing sphere. Checking the common continuous and soft variation patterns in the uncertainty of the object localization tool it is expected that uncertainty in viewpoints located in the vicinity of an studied viewpoint have similar uncertainty regions. Then a relaxation method could be a possible strategy to transform the finite number of viewpoints into an continuous sphere. Unfortunately, this is not always the case. Then, the problem would be to determine how many viewpoints to study and where to locate such viewpoints to interpolate the uncertainty regions for intermediate viewpoints. The same phenomena appears in the case of the success ratios and Psp values.

Another interesting question is related with the effect of using multiple images from different positions of the camera, or even, from different cameras or optical sensors. The problems becomes in one that requires of sensor fusion solutions. If the images are taken with the same camera, the problem becomes in the usual problem of active vision of deciding which is the best next sensor position.

6.2.3 Sensing Execution

Most of the ideas and proposed methods were tested individually. An interesting experimental study and something that would be good to see pursued is related to practical sensing decisions during the actual execution of complete assembly plans. A representation of the actual poses of the involved objects has to be maintained during execution.

Such representation would also be uncertain, then a model of actual uncertainty has also be maintained. These models together with the critical dependency relations codified in the ICdg, should be used to decide when is a good idea to perform some sensing operations to actualize the models because the adjustments in the configurations of manipulated objects will depend on a confidence measure of its outcome. This is also important to decide which objects to use as a reference for adjustment when an object depends on several objects over the same critical dimensions. But, before of doing this a more elaborated facility to perform force control must be implemented.

Bibliography

[1] Special issue: Sensor fusion and integration. *Journal of Robotics Society of Japan*, 8(6), 1990.

[2] Steven Abrams and Peter K. Allen. Swept volumes and their use in viewpoint computation in robot work-cells. In *IEEE Proc. Intl. Symposium on Assembly and Task Planning*, pages 188–193. IEEE, August 1995.

[3] Steven Abrams, Peter K. Allen, and Konstantinos A. Tarabanis. Computing camera viewpoints in a robot work-cell. In *Proc. 1996 IEEE Intl. Conference on Robotics and Automation*, pages 1972–1979, Minneapolis, Minnesota, USA, April 1996. IEEE.

[4] Peter Allen. Integrating vision and touch for object recognition tasks. *International Journal of Robotics Research*, 7(6):15–33, 1988.

[5] J. Y. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active vision. In *Proc. of 1st. Intl. Conf. in Computer Vision (ICCV)*, pages 35–54. IEEE, 1987.

[6] J. P. Baartman. *Automation of Assembly Operations on Parts*. PhD thesis, Faculty of Mechanical Engineering and Marine Technology, Delft University of Technology, Amsterdam, NL, 1995.

[7] L. Basañez, R Suárez, and J. Rosell. Contact situations from observed reaction forces in assembly with uncertainty. In *Proc. of the 13th IFAC World Congress*, pages 331–336, 1996.

[8] Jagdeep S. Basran and Emil M. Petriu. Translation of task-level instructions to sensing/actuation skills by a robotic assembling agent. In *IEEE Instrumentation and Measurement Technology Conference*, pages 593–598, Ottawa, Canada, May 1997. IEEE.

[9] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Appearance based active object recognition. *Image and Vision computing*, 18:715–727, 2000.

[10] Amy J. Briggs and Bruce R. Donald. Robust geometric algorithms for sensor planning. In *Proc. of the Second International Workshop on Algorithmic Foundations of Robotics*, pages 1–16, Toulouse, France, July 1996.

[11] Randy C. Brost and Matthew Mason. Graphical analysis of planar rigid-body dynamics with multiple frictional contacts. In *Fifth International Symposium of Robotics Research*, 1989.

[12] F. Cazals and J. C. Latombe. Effect of tolerancing on the relative position of parts in ans assembly. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 606–611. IEEE, 1997.

[13] S. Y. Chen and Y. F. Li. Optimum viewpoint planning for model-based robot vision. In *Proc. of the 2002 Congress on Evolutionary Computation*, volume 1, pages 634–639, May 2002.

[14] S. Y. Chen and Y. F. Li. Automatic sensor placement for model-based robot vision. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(1):393–408, February 2004.

[15] Cregg K. Cowan and Peter D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):407–416, May 1988.

[16] Gordon Dakin and Robin Popplestone. Fine-motion planning in the contact space of narrow-clearance assemblies. CS Technical Report 93-60, University of Massachusetts, Computer Science Department, July 1993.

[17] Ding Dan, Liu Yun-Hui, M. Y. Wang, and Wang Shuguo. Automatic selection of fixturing faces and fixturing points for polyhedral workpieces. *IEEE Transactions on Rootics and Automation*, 17(6):833–841, December 2001.

[18] J. Denzler and C. M. Brown. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):145–157, February 2002.

[19] Rajiv S. Desai and Richard A. Volz. Identification and verification of termination conditions in fine motion in presence of sensor errors and geometric uncertainties. In *Proc. 1989 IEEE Intl. Conf. on Robotics and Automation*, pages 800–807. IEEE, May 1989.

[20] G. N. DeSouza and A. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[21] Y. Ding and P. Kim. Optimal design of fixture layout in multistation assembly processes. *IEEE Transactions on Automation Science and Engineering*, PP(99):1–13, 2004.

[22] Bruce Donald. Planning and executing robot assembly strategies in the presence of uncertainty. Technical Report 89-1060, Cornell University, Department of Computer Science, November 1989.

[23] Bruce R. Donald. Planning multi-step error detection and recovery strategies. *International Journal of Robotics Research*, 9(1):3–60, February 1990.

[24] H. F. Durrant-White. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4(1):23–31, 1988.

[25] B. Eberman. A model-based approach to cartesian manipulation contact sensing. *The International Journal of Robotics Research*, 16(4):508–528, 1997.

[26] Michael Erdmann. On motion planning with uncertainty. Technical Report 810, MIT, AI Laboratory, 1984.

[27] David A. Forsyth and Jean Ponce. *Computer Vision: a modern approach*. Prentice Hall, 2003.

[28] J. H. Friedman, J. L. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[29] Janez Funda, Russell H. Taylor, and Richard P. Paul. On homogeneous transforms, quaternions, and computational efficiency. *IEEE Transactions on Robotics and Automation*, 6(3):382–387, June 1990.

[30] J. D. Geeter, J. De Schutter, H. Bruyninckx, H. V. Brussel, and M. Decrton. Tolerance-weighted l-optimal experimental design: a new approach to task-directed sensing. *Advanced Robotics*, 13(4):401–416, 1999.

[31] A. J. Goldman and A. W. Tucker. Linear inequalities and related systems. *Annals of Math. Studies*, 38:19–39, September 1956.

[32] X. Gu, M. M. Marefat, and F. W. Ciarallo. A robust approach for sensor placement in automated vision dimensional inspection. In *Proc. of the 1999 IEEE Intl. Conference on Robotics and Automation*, pages 2602–2607, Detroit, Michigan, USA, May 1999. IEEE.

[33] Robert M. Haralick, Hyonam Joo, Chung-Nan Lee, Xinhua Zhuang, Vinay G. Vaidya, and Man Bae Kim. Pose estimation from corresponding point data. *IEEE Transactions on systems, Man, and Cybernetics*, 19(6):1426–1446, 1989.

[34] T. Hasegawa, T. Suehiro, and K. Takase. A model-based manipulation system with skill-based execution. *IEEE Transactions on Robotics and Automation*, 8(5):535–544, 1992.

[35] M. Hashimoto. Real time estimation of contact position based on force/velocity sensor fusion. In *Proc. of the 7th International Conference on Advanced Robotics*, pages 651–654, 1995.

[36] J. Hill and W. T. Park. Real time control of a robot with a mobile camera. In *Proc. of the 9th ISIR*, pages 233–246, Washington, D.C., March 1979.

[37] S. Hirai and H. Asada. Kinematics and statics of manipulation using the theory of polyhedral convex cones. *The International Journal of Robotics Research*, 12(5):434–447, 1993.

[38] S. Hirai and K. Iwata. Recognition of contact state based on geometric model. In *Proc. of the 1992 IEEE International Conference on Robotics and Automation*, pages 1507–1512. IEEE, 1992.

[39] K. Hirana, T. Suzuki, and S. Okuma. Optimal motion planning for assembly skill based on mixed logical dynamical system. In *Proc. of the 7th International Workshop on Advanced Motion Control*, pages 359–364, July 2002.

[40] H. Hirukawa, T. Matsui, and K. Takase. Automatic determination of possible velocity and applicable force of frictionless objects in contact from a geometric model. *IEEE Transactions on Robotics and Automation*, 10(3):309–320, June 1994.

[41] R. Hoffman, K. Ikeuchi, P. Balakumar, J. C. Robert, and T. Kanade. Vantage: A frame-based geometric/sensor modeling system - programmer/user's manual v1.0. Technical report, Carnegie Mellon University, Pittsburgh, PA, 1991.

[42] N. Hogan. Impedance control: An approach to manipulation: Part i- theory; part ii- implementation; part iii-applications. *Transactions ASME Journal on Dynamic Systems, Measurement, and Control*, 107(1):1–24, 1985.

[43] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.

[44] Seth A. Hutchinson and Avinash C. Kak. Planning sensing strategies in a robot work cell with multi-sensor capabilities. *IEEE Transactions on Robotics and Automation*, 5(6):765–783, December 1989.

[45] Katsushi Ikeuchi and Martial Hebert. Task oriented vision. In *Proc. 1992 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 2187–2194. IEEE, July 1992.

[46] Katsushi Ikeuchi and Takeo Kanade. Towards automatic generation of object recognition programs. Technical Report CMU-CS-88-138, Carnegie Mellon University, School of Computer Science, May 1988.

[47] Katsushi Ikeuchi, Masato Kawade, and Takashi Suehiro. Assembly task recognition with planar, curved and mechanical contacts. In *Proc. 1993 IEEE Intl. Conf. on Robotics and Automation*, pages 688–694. IEEE, May 1993.

[48] Katsushi Ikeuchi and Takashi Suehiro. Towards an assembly plan from observation: Task recognition with polyhedral objects. Technical Report CMU-CS-91-167, Carnegie Mellon University, School of Computer Science, August 1991.

[49] Katsushi Ikeuchi, Takashi Suehiro, Peter Tanguy, and Mark Wheeler. Assembly plan from observation. Annual research review 1990, Carnegie Mellon University, The Robotics Institute, 1990.

[50] M. Inui, M. Miura, and F. Kimura. Positioning conditions of parts with tolerances in an assembly. In *Proc. IEEE Intl. Conf. on Robotics and Automation*, pages 2202–2207. IEEE, 1996.

[51] T. Ishikawa, S. Sakane, T. Sato, and H. Tsukune. Estimation of contact position between a grasped object and the environment based on sensor fusion of vision and force. In *Proc. of the 1996 IEEE/SICE/RSJ Intl. Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 116–123. IEEE/SICE/RSJ, 1996.

[52] M. Kaiser, A. Girodana, and M. Nuttin. Integrated acquisition, execution, and tuning of elementary skills for intelligent robots. In *Artificial Intelligence in Real Time control*, pages 117–122, International Federation of Automatic Control, 1994.

[53] A. C. Kak, K. L. Boyer, C. H. Chen, R. J. Safranek, and H. S. Yang. A knowledge-based robotic assembly work cell. *IEEE Expert*, 1:63–83, 1986.

[54] Lydia E. Kavraki and Mihail N. Kolountzakis. Partitioning a planar assembly into two connected parts is np-complete. *Information Processing Letters*, 55(3):159–165, August 1995.

[55] K. W. Khawaja, A. A. Maciejewski, D. Tretter, and C. A. Bouman. Camera and light placement for automated assembly inspection. In *Proc. of the 1996 IEEE Intl. Conference on Robotics and Automation*, pages 3246–3252. IEEE, April 1996.

[56] K. Kitagaki, T. Ogasawara, and T. Suehiro. Methods to detect contact state by force sensing in and edge mating task. In *Proc. of the 1993 IEEE International Conference on Robotics and Automation*, pages 701–706. IEEE, 1993.

[57] H. W. Kuhn and A. W. Tucker. Linear inequalities and related systems. *Annals of Math Studies*, 39, 1956.

[58] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[59] Z. Liu and T. Nakamura. Skill-based micromanipulation system for assembly operation. In *Proceedings of 2003 International Symposium on Micromechatronics and Human Science*, pages 197–203, October 2003.

[60] Tomás Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[61] Tomás Lozano-Pérez, Matthew T. Mason, and Russel H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24, Spring 1984.

[62] Matthew T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11:418–432, June 1981.

[63] B. J. McCarragher, G. Hovland, P. Sikka, P. Aigner, and D. Austin. Hybrid dynamic modeling and control of constrained manipulation systems. *IEEE Robotics and Automation Magazine*, pages 27–44, June 1997.

[64] Jun Miura and Katsushi Ikeuchi. Task-oriented generation of visual sensing strategies in assembly tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(2):126–138, February 1998.

[65] J. D. Morrow, B. J. Nelson, and P. K. Khosla. Vision and force driven sensorimotor primitives for robotic assembly skills. In *Proc. of the 1995 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, volume 3, pages 234–240, August 1995.

[66] H. Mosemann and F. M. Wahl. Automatic decomposition of planned assembly sequences into skill primitives. *IEEE Transactions on Robotics and Automation*, 17(5):709–718, October 2001.

[67] H. Murase and S. K. Nayar. Illumination planning for object recognition using parametric eigenspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-16(12):1219–1227, December 1994.

[68] A. Nakamura, T. Ogasawara, T. Suehiro, and H. Tsukune. Skill-based backprojection for fine motion planning. In *Proc. of the 1996 IEEE Intl. Conference on Intelligent Robots and Systems*, pages 526–533. IEEE, 1996.

[69] United Nations. World robotics 2003 - statistics, market analysis, forecasts, case studies and profitability of robot investment, 2003.

[70] S. K. Nayar and R. M. Bolle. Reflectance based object recognition. *International Journal of Computer Vision*, 17(3):219–240, 1996.

[71] Bradley J. Nelson and Pradeep K. Khosla. Force and vision resolvability for assimilating disparate sensory feedback. *IEEE Transactions on Robotics and Automation*, 12(5):714–731, 1996.

[72] G. Olague and R. Mohr. Optimal camera placement to obtain accurate 3d point positions. In *Proc. of the 14th Intl. Conf. on Pattern Recognition*, pages 8–10, 1998.

[73] L. Paletta and A. Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31:71–86, 2000.

[74] George V. Paul. *Kinematics of Objects in Contact using Dual Vectors and its Applications*. Ph.d thesis, Carnegie Mellon University, The Robotics Institute, November 1997.

[75] E. Paulos and J. Canny. Accurate insertion strategies using simple optical sensors. In *Proc. 1994 IEEE Intl. Conf. on Robotics and Automation*, volume 2, pages 1656–1662. IEEE, May 1994.

[76] Marc H. Raibert and John J. Craig. Hybrid position/force control of manipulators. *ASME Journal of Dynamic Systems, Measurement, and Control*, 102:126–133, June 1981.

[77] M. K. Reed and P. K. Allen. Constraint-based sensor planning for scene modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1460–1467, December 2000.

[78] J. Rosell, L. Basañez, and R. Suárez. A probabilistic method to analyze ambiguous contacts situations. In *Proc. of the 6th IFAC Symposium on Robot Control, SYROCO'2000*, volume 2, pages 415–420, 2000.

[79] N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation - mobile robot navigation with uncertainty in dynamic environments. In *Proc. of the IEEE Intl. Conference on Robotics and Automation*, volume 1, pages 35–40. IEEE, May 1999.

[80] S. J. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Prentice Hall, 2003.

[81] S. Sakane, R. Niepold, T. Sato, and Y. Shirai. Illumination setup planning for a hand-eye system based on an environmental model. *Advanced Robotics*, 6(4):461–482, 1992.

[82] S. Sakane and T. Sato. Automatic planning of light source and camera placement for an active photometric stereo system. In *IEEE Intl. Conference on Robotics and Automation*, pages 1080–1087, Sacramento, CA, USA, April 1991. IEEE.

[83] B. Schiele and J. L. Crowley. Transinformation for active object recognition. In *Proc of the Sixth Intl. Conference in Computer Vision*, 1998.

[84] K. Selke, K. G. Swift, G. E. Taylor, A. Pugh, S. N. Davey, and G. E. Taylor. Knowledge-based robotic assembly - a step further towards flexibility. *Computer-Aided Engineering Journal*, 4:62–67, February 1987.

[85] Robert M. Haralick Seungku Yi and Linda G. Shapiro. Automatic sensor and light source positioning for machine vision. In Steven A. Shafer Lawrence B. Wolff and Glenn E. Healey, editors, *Physics-Based Vision, Principles and Practice*, volume RADIOMETRY, pages 394–398. 1992.

[86] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[87] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, 1963.

[88] W. Sheng, N. Xi, M. Song, and Y. Chen. Cad-guided sensor planning for dimensional inspection in automotive manufacturing. *IEEE/ASME Transactions on Mechatronics*, 8(3):372–380, September 2003.

[89] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5:99–108, 1973.

[90] M. Skubic and R. A. Volz. Acquiring robust, force-based assembly skills from human demonstration. *IEEE Transactions on Robotics and Automation*, 16(6):772–781, December 2000.

[91] T. Smithers and C. Malcolm. Programming robotic assembly in terms of task achieving behavioural modules. In *DAI Research Paper No. 417*, 1988.

[92] Fredric Solomon. *Illumination Planning for Photometric Measurements*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, June 1996.

[93] M. Spreng. A probabilistic method to analyze ambiguous contacts situations. In *Proc. of the 1993 IEEE International Conference on Robotics and Automation*, pages 543–548, USA, May 1993. IEEE.

[94] D. B. Steward, D. E. Schmitz, and P. K. Khosla. The chimera ii real-time operating system for advanced sensor-based control systems. *IEEE Transactions o Systems, Man and Cybernetics*, 22(6):1282–1295, 1992.

[95] Gilbert Strang. *Introduction to Linear Algebra*. Wellesleyh-Cambridge Press, 1993.

[96] T. Suehiro. Study of an advanced manipulation system. Technical Report 912, Researches of the Electrotechnical Laboratory, 1990. in Japanese.

[97] Takashi Suehiro and Katsushi Ikeuchi. Towards an assembly plan from observation. part ii: Correction of motion parameters based on fact contact constraints. In *Proc. 1992 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, pages 2095–2102. IEEE, July 1992.

[98] R. Sutton and A. Barto. *Reinforcement Learning, An Introduction*. MIT, 1998.

[99] Jun Takamatsu. *Abstraction of Manipulation Tasks to automatically Generate Robot Motion From Observation*. PhD thesis, Graduate School of Information Science and technology, The University of Tokyo, Tokyo, Japan, 2003.

[100] Konstantinos A. Tarabanis, Peter K. Allen, and Roger Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11(1):86–104, February 1995.

[101] Konstantinos A Tarabanis, Roger Y. Tsai, and Peter K. Allen. The mvp sensor planning system for robotic vision tasks. *IEEE Transactions on Robotics and Automation*, 11(1):72–85, February 1995.

[102] Roger J. Thien and Simon D. Hill. Fusion of force and vision data for intelligent assembly. In *IEEE Region 10 Conference, Tencon 92*, pages 257–261, Melbourne, Australia, November 1992. IEEE.

[103] H. Tominaga and K. Ikeuchi. Acquiring manipulation skills through observation. In *Proc. of the 1999 IEEE Intl. Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 7–12, Taipei,Taiwan,R.O.C., August 1999. IEEE.

[104] W. O. Troxell. A robotic assembly description language derived from task-achieving behaviors. In *Proc. of Manufacturing International'90*, pages 25–28, Atlanta,GA, March 1990.

[105] E. Trucco, M. Umasuthan, A. M. Wallace, and V. Roberto. Model-based planning of optimal sensor placements for inspection. *IEEE Transactions on Robotics and Automation*, 13(2):182–194, April 1997.

[106] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[107] Y. von Collani, C. Scheering, J. Zhang, and A. Knoll. A neuro-fuzzy solution for integrated visual and force control. In *Proc. of the 1999 IEEE Intl. Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 135–140, Taipei, Taiwan,R.O.C., August 1999. IEEE.

[108] Mark D. Wheeler. *Automatic Modeling and Localization for Object Recognition.* Ph.d thesis, Carnegie Mellon University, School of Computer Science, October 1996.

[109] D. Whitney. Force feedback control of manipulator fine motions. *Trans. of ASME, Journal of Dynamics, Systems, Measurement and Control*, pages 91–97, June 1977.

[110] Jing Xiao. Automatic determination of topological contacts in the presence of sensing uncertainties. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, volume 1, pages 65–70. IEEE, 1993.

[111] Jing Xiao and Lixin Zhang. An efficient algorithm (fapric) for finding the principal contacts possibly established due to uncertainties. In *Proc. of the 1995 IEEE International Conference on Robotics and Automation*, pages 427–432. IEEE, 1995.

[112] Jing Xiao and Lixin Zhang. Toward obtaining all possible contacts - growing a polyhedron by its location uncertainty. *IEEE Transactions on Robotics and Automation*, 12(4):553–565, 1996.

[113] Jing Xiao and Lixin Zhang. Contact constraint analysis and determination of geometrically valid contact formations from possible contact primitives. *IEEE Transactions on Robotics and Automation*, 13(3):456–466, 1997.

[114] Hong Zhang. Two-dimensional optimal sensor placement. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(5):781–792, May 1995.

Appendix A

Representation of Rigid Transformations

A.1 2D Rigid Transformations

2D rigid transformations can be represented by using 3x3 homogeneous matrices as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \bar{0} & 1 \end{bmatrix} \tag{A.1}$$

where $\mathbf{t}$ is the 2D translation vector $(t_x, t_y)^T$ and

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{A.2}$$

which performs a rotation around the origin of the coordinate frame. Substituting, we obtain the full 2D rigid transformation matrix as

$$\mathbf{T} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

.

A.2 3D Rigid Transformations

3D rigid transformations can be represented by using 4x4 homogeneous matrices as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \bar{0} & 1 \end{bmatrix} \tag{A.4}$$

where $\mathbf{t}$ is the 3D translation vector $(t_x, t_y, t_z)^T$ and

$$\mathbf{R} = \mathbf{R}_z(\gamma) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) \tag{A.5}$$

is a 3x3 rotation matrix computed from the basic 3D rotational matrices:

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{A.6}$$

which performs a rotation around the X coordinate axis,

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{A.7}$$

which performs a rotation around the Y coordinate axis, and

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.8}$$

which performs a rotation around the Z coordinate axis. Substituting, we obtain the full rigid transformation matrix as

$$\mathbf{T} = \begin{bmatrix} c\beta\, c\gamma & s\alpha\, s\beta\, c\gamma - c\alpha\, s\gamma & c\alpha\, s\beta\, c\gamma + s\alpha\, s\gamma & t_x \\ c\beta\, s\gamma & s\alpha\, s\beta\, s\gamma + c\alpha\, c\gamma & c\alpha\, s\beta\, s\gamma - s\alpha\, c\gamma & t_y \\ -s\beta & s\alpha\, c\beta & c\alpha\, c\beta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.9}$$

where c stands for cosine, s stands for sine, and α, β, and γ are the rotation angles with respect to the X, Y, and Z axis respectively.

Appendix B

Assembly Planning Terminology

The terms and definitions below are mostly personal preferences. Many are not standard.

Binary plan A binary plan is one which requires two hands, i.e., no step requires three or more parts or subassemblies to move in different directions simultaneously.

Component A component is a part or a subassembly. Other authors either allow a subassembly to be a single part, allow a part to be a subassembly, or always say "part or subassembly". I find a separate term more intuitive.

Global or extended motion A global motion is a motion over an infinite distance. A part or a subassembly is said to be **globally free** if it can be removed to infinity.

Linear plan A linear plan inserts parts one at a time. That is, it never forms subassemblies. Obviously any planner that generates linear plans can be used to generate nonlinear ones if the user selects the subassemblies.

Local motion A local motion is a motion over an infinitesimal distance. A part or a subassembly is said to be locally free if it can move an infinitesimal distance without colliding with other parts.

Mating operations A mating operation consists of putting together two (or more) parts or subassemblies. This is a slightly higher-level operation description than pick-and-place operations. Usually some relative motion by which the parts can be joined will be known, but it may not necessarily be known which of the mating parts is held and which is moved.

Monotone plan A sequential plan is monotone if each step ends with all moved parts in their final position i.e. each subassembly, once constructed, is final. This means that parts are never put into temporary positions. This constraint makes

planning a lot easier, both because it avoids the problem of finding useful temporary positions, and because it vastly decreases the search space size. However non-monotone plans are quite commonly used in building assemblies with moving parts. Warning: "Non-monotone planning" means something completely unrelated to "non-monotone plans" in the AI literature. Do not call a planner which generates non-monotone plans is a non-monotone planner.

Number of hands The number of hands required to perform a given step in an assembly sequence is defined as the number of parts or subassemblies that are moving with respect to one another. For example, if a single part or subassembly is being removed from the rest of the assembly, then two hands are required: one for the moving part or subassembly, and one for the fixed partial assembly. If the fixed partial assembly is merely resting on a table, then, the table acts as a "hand" in this context.

Part A part is one of the primitives that must be combined into the assembly in the plan. Some parts may actually be assemblies, like ball-bearings, which have already been built when supplied to us. But since they are already built, and we don't intend to disassemble them, we treat them as parts and call them parts. Note that some authors do not count fasteners (e.g., nuts, bolts, rivets) as full-fledged parts but only as attributes on a connection between two parts.

Partial assembly Any subset of parts that occurs at any point in time during an assembly plan is called a partial assembly. Single parts may be partial assemblies.

Pick-and-place operations A pick-and-place operation moves a part to a given goal position. Usually the trajectory along which the part is moved will be described.

Plan Obviously a plan is a description of how an assembly is to be built. It is important to note, however, that plans generally do not describe all details of the assembly process. Since many details are not specified by the plan, there can be many different processes which can be considered valid executions of a given plan. The level of detail in plans varies widely between assembly planners.

Sequential plan A plan is sequential if it can be decomposed into a sequence of steps such that during each step only one component is in motion. This is also called "two-handed". Note that a plan is sequential if it can be executed in this manner. Often it is possible to do two steps in a parallel plan at the same time, as when a pit-crew changes two tires of a race car at the same time. This is still sequential because the simultaneous operations are independent and could be done sequentially. A plan is only non-sequential if two or more parts must be simultaneously inserted in a coordinated motion along different trajectories. Such plans are unusual in practice, but not unknown. This usage is consistent with Natarajan's

definition of handedness. An assembly is n-handed if the minimum number of hands required to build it is n.

Stack plan An stack plan is one in which all the motions occur in a single direction, usually up-and-down. Circuit boards, for example, often have stack plans. An assembly for which a stack plan exists is called a stack assembly.

Steps and operations Just about any interesting plan consists of a collection of step or operations which must be performed, and some information about the order in which those steps or operations must be performed. Usually operations are described either as insertion operations or mating operations.

Subassembly An assembly or subassembly is a partial assembly of one or more parts that is built by the insertion of one or more parts or subassemblies. The distinction between "assembly" and "subassembly" is contextual. We build an assembly from subassemblies, and then that assembly may become a subassembly in a higher-level assembly. Note that the distinction between "partial assembly" and "subassembly" is pretty much unique to me. Most authors use "subassembly" for both. but I feel the distinctions between the two is important for generating good plans. For example, while disconnected partial assemblies often occur in real plans, disconnected subassemblies are virtually unknown.

Totally-ordered assembly sequence A totally-ordered assembly sequence is one in which the operations are executed as they were uniquely-specified. Many plans represent only partially-ordered operations, giving just those sequencing constraints that are necessary.

Appendix C

The 2DTM Object Localization Tool

To deduce the configuration of the assembly objects in the scene is used a model-based tool (2DTM) that localize 3D object models in one or multiple 2D intensity images by performing M-estimation with dynamic correspondences. This tool was originally developed and programmed by Mark D. Wheeler as part of is Ph.D. Dissertation at Carnegie Mellon University [108]. However, some adequations and modifications to both the software and procedures to get the models of the objects had to be realized.

The models of the objects are described as a collection of 3D points on the object's surface which often create edges when visible in intensity images; these points are further referred as *edgel generators*. The edgel generators are matched to identified 2D contrast points that compose intensity edges in the image; these are further referred as intensity edgels. Using these correspondences, an error measure for the pose of the model is defined by measuring a 3D distance between the matching features. The error is minimized by using a robust statistical M-estimator that adjust the values for the model pose parameters. A process of pose refinement is performed by dynamically modifying the correspondences and minimizing the new error measures.

The same procedure can be applied using multiple intensity images, and solving for pose without computing depth. In this case, during each iteration of the algorithm the model edgels generate correspondences with the intensity edgels in every image and the error measure to minimize is composed by the increased number of correspondences. By using multiple images, the localization is expected to become more robust and accurate.

Several types of image features have been used to approach the problem of 3D object localization: pixels, regions, interest points, edge curves, and edgels are some examples. Some of the reasons to use intensity edgels is that they are the most reliable, well-understood, practical, and most popular using present day techniques.

Points in the scene are detected as intensity edgels when discontinuities on the objects due to the surface orientation, surface reflectance, and geometry occur. Unfortunately,

other sources of image discontinuities exist that are not intrinsic to objects, like digitalization noise and casted shadows, that depend on the sensor and the illumination system. However, intrinsic edge discontinuities are still detectable in a wide range of situations and can be accurately predicted.

The edgel model was designed having in mind its effect in the localization process. Though, Wheeler developed a method to automatically extract the object models from a set of range and intensity images, in the case of this work, the edgel generators are derived from CAD models.

C.1 3D-2D Object Representation

The model edgels are represented as a collection of oriented points in three dimensions in the coordinate system of the object, where each edgel has a local visibility criteria associated with it.

The original model was designed to represent free-shape objects that were composed by three types of edgels: surface edgels resulted of reflectance and color discontinuities on the surface of the object, convex edgels resulted of discontinuities of surface normals, and occluding-contours edgels resulting of boundaries at points on a smooth surface where geometry discontinuities are detected from a particular viewing direction.

In the case of polyhedral objects, the occluding contours coincide with edges produced by normal discontinuities and are visible from the set of viewing directions that allow seeing only one of the edge generator faces. Thus, for polyhedral objects, the original analysis to determine occluding-contours edgels can be eliminated.

The 2DTM model contains a set of counters that describe the number of each type of edgels contained by the model and the description of each 3D edgel. Each 3D edgel is described by a set of photometric and geometric attributes.

The most obvious photometric attributes of an edge are the intensities on either side of the edgel, but they were not used because intensities are difficult to predict under variance in the illumination. The reflectance ratio is the only photometric attribute included in the model and it is used when the correspondences for surface edgels are determined. This parameter was selected because it was showed to be invariant to light source conditions and orientations (except for cases of shadowing and specular reflection) [70] when computed from adjacent surface patches with similar reflectance properties and surface normals that are nearly the same. These requirements are not fulfilled by the convex and occluding-contour edgels of the model. The reflectance ratio α is defined as

$$\alpha = \frac{I_{left} - I_{right}}{I_{left} + I_{right}} \tag{C.1}$$

where I_{left} and I_{right} are the intensities to both sides of the edgel or surface patch boundary.

The attribute values were computed when building the model from real images by averaging the observed intensities from different directions and under several illumination conditions. Since, here, the model is not obtained from real images, the reflectance ratio has to be obtained from an additional analysis of the reflectance properties of the object or be ignored.

The construction process for the 3D-2D model representation for 2DTM from CAD models of the assembly parts can be summarized in four steps:

1. Create the VANTAGE CSG model of the assembly parts. If some parts include marks, use the marks model construction tool to describe these.
2. Obtain the boundary representation (B-REP) of the CSG models of the assembly parts. A program was devised that extracts the information of the geometric features of the objects from its CAD models.
3. Create a triangulated representation from the boundary representation of the assembly parts. A program is used that subdivide every polygonal face of a polyhedral object into a triangular grid.
4. Create the 3D edgel generators from the boundary and triangulated representations of the assembly parts. The boundary representation is used to extract the edges of the faces that form the polyhedral objects. This edges are segmented. Every edge segment is a edgel generator. A visibility representation for every edgel generator is stored with it. The triangulated representation of the objects is used as a tool to construct the visibility representation of each edgel generator.

C.2 3D-2D Object Localization

The goal is to localize 3D objects that are expected to be in the scene from 2D intensity images. An rough initial estimate of the object's pose is known and since most probably it includes a certain amount of error, an iterative process of pose refinement will be performed to correct the initial estimation. This scenario is common while performing assembly tasks; known objects in known configurations are expected to pass for a sequence of known transitions. The initial estimate required by the localization system can be extracted from the assembly plan.

2DTM is a method that solves the problem in three steps:

1. Determine the model edgels visibility: which parts (edgels) of the object are visible from the current viewpoint.

2. Determine the Model-to-Image Correspondences: model edgels are connected or related to the image edgels based on a similarity metric in 2D image space.

3. Performs the pose optimization: minimize a 3D error measure using a robust statistical estimator.

The first correspondences will usually be incorrect-especially with large initial pose errors. For localization to work, the above steps are iteratively applied and the visible edgels and its correspondences are dynamically determined. Under the assumption that wrong correspondences has a random pulling effect on the object's pose and true correspondence a consistent one, the model is expected to move in the correct direction. As the model moves through its correct pose more true correspondences are expected to appear with the consistent effect of better the object's pose estimation.

If the wrong correspondences has a consistent wrong tendency and there are not enough true correspondences to counteract the wrong trend the result can worsen the initial error. As in the general case of the correspondence-based optimization methods, the accuracy of the final results depend in the correctness of the identified correspondences.

C.2.1 Edgel Visibility

The first step is to determine from all the model's edgels which are visible from the current viewpoints. The objects can be subject to self-occlusion (some parts of the object occlude some others) and/or external occlusion (different objects occlude parts or all the objects). Determining exactly which parts of the object are occluded can be done using methods as ray-casting or z-buffering, but it is computationally very expensive. Since the visibility test will be performed for every iteration during the pose refinement process, an approximated but efficient representation of the visibility of each edgel under the possible viewing directions was selected for determining edgel visibility under possible self-occlusion. The system ignores external occlusion.

When self-occlusion is not possible, a simple and efficient convex visibility test based on the normal directions associated with the edgels and the viewing directions is defined. In the case of surface edgels the visibility test evaluates if the surface (object's face) where the edgel point lies is visible and its computation is

$$visible_{surface}(\mathbf{x}) = \begin{cases} true & \mathbf{n}_i \cdot \mathbf{v} > cos\theta \\ false & otherwise \end{cases} \tag{C.2}$$

where $\mathbf{n}_i$ is the normal direction of the surface, $\mathbf{v}$ is the viewing direction for the edgel point $\mathbf{x}$, and θ is an angle near 90 degrees which is the maximum orientation angle of the surface such that a rigid edgel on the surface will be detected.

A convex edgel point is declared visible if any of the adjacent surfaces (object's faces) that generate the edge where it lies is visible from the current viewpoint. The edgel point visibility computation is

$$visible_{convex}(\mathbf{x}) = \begin{cases} true & (\mathbf{n}_{left} \cdot \mathbf{v} > cos\theta) \vee (\mathbf{n}_{right} \cdot \mathbf{v} > cos\theta) \\ false & otherwise \end{cases} \tag{C.3}$$

where $\mathbf{n}_{left}$ and $\mathbf{n}_{right}$ are the normals of the left and right generator surfaces, respectively.

When self-occlusion is possible, the visibility test is transformed so that it can be performed by consulting a visibility look-up table (LUT). To create the visibility LUT, the space of possible viewing directions is tessellated into discrete bins representing sets of viewing directions. All the viewing directions of a particular bin are considered equivalent for the visibility computation. Separated LUTs are allocated for each edgel point in the model. Each bin in the visibility LUT contains a binary value indicating whether the point is visible or not from the set of viewing directions which map to the given bin.

All the possible viewing directions are represented as points on the surface of Gaussian sphere. In the case of surface edgel points only a hemisphere needs to be considered since the other will always be occluded.

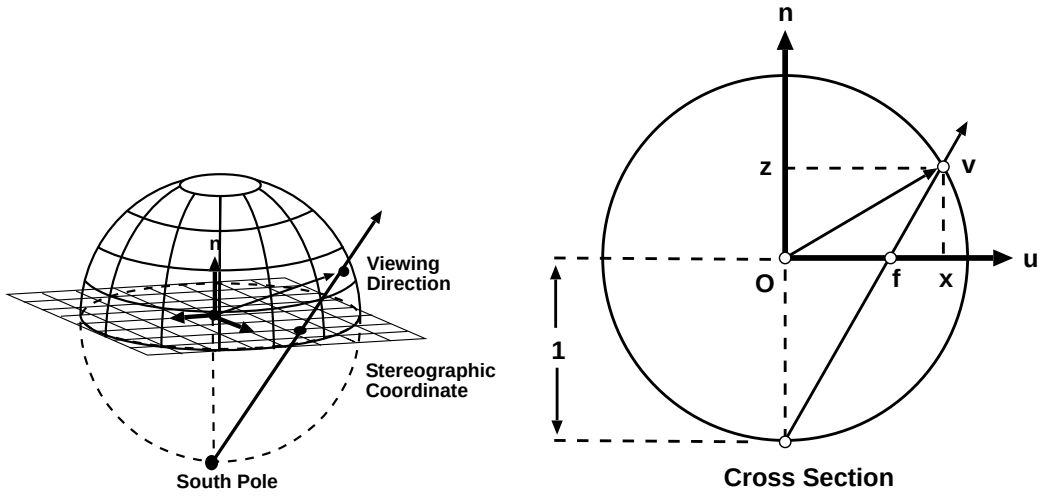
The selection of the tessellation strategy and the mapping of the viewing directions to the visibility LUT's bins was selected based on the following criteria:

- The bins should cover approximately uniform areas of the hemisphere.
- The mapping from viewing directions to bins must be efficient.

A latitudinal/longitudinal discretization of the hemisphere is employed as the tessellation strategy using stereographic projection for the mapping (see fig. C.1a). The projection of the viewing direction $\mathbf{v} = [x\ y\ z]^T$ to 2D stereographic coordinates $[f\ g]^T$ is computed by

$$f = \frac{x}{z+1} \tag{C.4}$$

$$g = \frac{y}{z+1} \tag{C.5}$$



(a) Visibility LUT creation.

(b) Stereographic projection formulation.

Figure C.1: Mapping of viewing directions to a latitudinal/longitudinal tessellated hemisphere by stereographic projection.

C.2.2 Model-to-Image Edgel Correspondence

A local search is used to find the nearest-neighbor correspondences based in a 2D dissimilarity measure computed as

$$\Delta(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| \tag{C.6}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the image coordinates of two points being compared. The nearest-neighbor search is implemented using the k-d tree data structure [28].

Since using only the position coordinates in 2D edgel images is a very weak strategy to get good correspondences, additional geometric and photometric attributes are used.

The 2D normal of the projected 3D edge that contain the candidate edgel is included as a geometric attribute. The normals are represented as 2D unit vectors. Two entries are used for each edgel to cope with the ambiguity on normal direction.

The reflectance ratio computed as

$$\alpha = \frac{I_left - I_right}{I_left + I_right} \tag{C.7}$$

where I_left and I_right are the intensities to both sides of the edgel or surface patch boundary, is the photometric attribute used to determine correspondences for surface edgels. The reflectance ratio is the photometric attribute used to determine correspondences for surface edgels. The reflectance ratio was showed to be invariant to light source conditions and orientation (except for cases of shadowing and specular reflection) for two adjacent surface patches when its normals are nearly the same [70].

C.2.3 Pose Optimization

From the initial rough pose estimate of the object model, the program performs an iterative process of pose refinement by considering the problem in an optimization framework.

Since many outliers are expected in the set of correspondences, the process use a robust M-estimator to solve for the pose of the object. It minimizes

$$E(\mathbf{p}) = \sum_{i \in V(\mathbf{p})} \rho(z_i(\mathbf{p})) \tag{C.8}$$

where z_i is the error of the ith point (edgel) in the model, $\rho(z)$ is the robust M-estimator, and $V(\mathbf{p})$ is the set of visible model edgels at pose $\mathbf{p}$ with respect to the known camera parameters.

A gradient-descent search is used to minimize $E(\mathbf{p})$. The error for each correspondence is formulated as the perpendicular distance between the 3D edgel and the line of sight of the image edgel, s.t. as the shortest motion of the 3D edgel to align it with the image edgel. It is computed as

$$z_i = \|\mathbf{y}_i - \mathbf{y}_i^c\|^2. \tag{C.9}$$

where z_i is the 3D error for the ith edgel correspondence, $\mathbf{y}_i$ are the coordinates for the 3D edgel, and $\mathbf{y}_i^c$ are the coordinates of the 3D projection of the image edgel, which is computed as

$$\mathbf{y}_i^c = (\mathbf{u}_i \cdot \mathbf{v})\mathbf{v}. \tag{C.10}$$

where $\mathbf{u}_i$ are the coordinate of the image edgel and $\mathbf{v}$ the viewing direction (see Figure C.2)

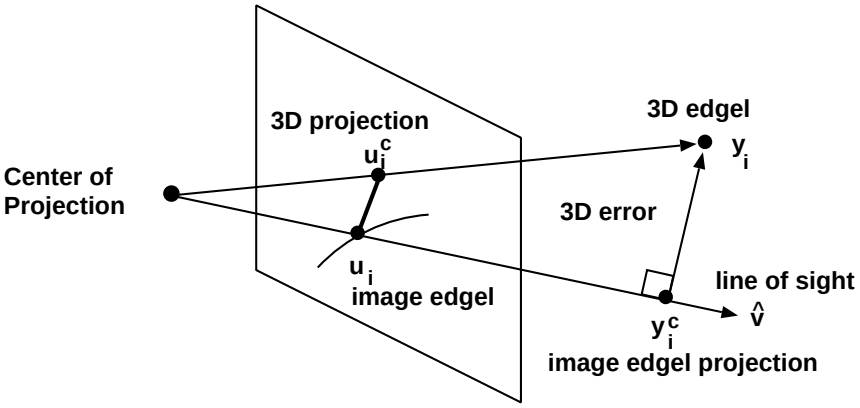


Figure C.2: 3D error computation for a edgel correspondence.