

VISION-BASED GRASP PLANNING OF 3D OBJECTS USING
GENETIC ALGORITHM

by

Zichen Zhang

Submitted in partial fulfillment of the
requirements for the degree of
Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
August 2012

© Copyright by Zichen Zhang, 2012

DALHOUSIE UNIVERSITY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “VISION-BASED GRASP PLANNING OF 3D OBJECTS USING GENETIC ALGORITHM” by Zichen Zhang in partial fulfillment of the requirements for the degree of Master of Applied Science.

Dated: August 1, 2012

Supervisor:

Dr. Jason Gu

Readers:

Dr. Williams J. Phillips

Dr. Yuan Ma

DALHOUSIE UNIVERSITY

DATE: August 1, 2012

AUTHOR: Zichen Zhang

TITLE: VISION-BASED GRASP PLANNING OF 3D OBJECTS USING
GENETIC ALGORITHM

DEPARTMENT OR SCHOOL: Department of Electrical and Computer Engineering

DEGREE: M.A.Sc.

CONVOCATION: October

YEAR: 2012

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions. I understand that my thesis will be electronically available to the public.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

The author attests that permission has been obtained for the use of any copyrighted material appearing in the thesis (other than brief excerpts requiring only proper acknowledgement in scholarly writing), and that all such use is clearly acknowledged.

Signature of Author

To my Mom Ping Ding, and Dad Baoping Zhang

Table of Contents

List of Tables	vii
List of Figures	viii
Abstract	x
List of Abbreviations and Symbols Used	xi
Acknowledgements	xii
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Previous Research	2
1.2.1 Different Robot Hand	2
1.2.2 How to Perceive the Environment	2
1.2.3 Problem Description	3
1.3 Objectives and Contribution	4
1.4 Outline	5
Chapter 2 Background	6
2.1 Force-Closure Grasp	6
2.2 Contact Model	7
2.3 Quality Metrics	7
2.4 Grasp Planning in <i>Eigengrasp</i> Space	9
2.5 Genetic Algorithm	10
Chapter 3 Genetic Algorithm on Grasp Planning	12
3.1 Solution Landscape	12
3.2 Operators	15
3.3 Sampling Method	16
3.3.1 Type1 sampling	17
3.3.2 Type2 sampling	18

3.3.3	Type3 sampling	18
3.3.4	Type4 sampling	18
3.3.5	Type5 sampling	19
3.3.6	Type6 sampling	19
Chapter 4	Results and Discussion	22
4.1	Implementation	22
4.2	Parameter Tuning	28
4.3	Performance	39
4.4	Comparison with Simulated Annealing Planner	47
4.5	Grasp Planning using Another Quality Metric	52
Chapter 5	Conclusion and Future Work	57
5.1	Conclusion	57
5.2	Future Work	58
Bibliography	59

List of Tables

2.1	Variable List	10
3.1	Operator List	17
4.1	Dimension of the search space	24
4.2	Different population size, BLX-0.5(Type1 sampling), $K_{\sigma} = 0.2$ (repetition sampling)	30
4.3	BLX-0.5 with type1 sampling, $K_{\sigma} = 0.2$ with different sampling methods for mutation	32
4.4	BLX-0.5 with type4 sampling, $K_{\sigma} = 0.2$ with different sampling methods for mutation	33
4.5	BLX-0.5 with type3 sampling, $K_{\sigma} = 0.2$	34
4.6	BLX-0 with type1 sampling, $K_{\sigma} = 0.2$	35
4.7	BLX-0.25 with type1 sampling, $K_{\sigma} = 0.2$	35
4.8	Statistics of performance index for type1,3,4 sampling of BLX-0.5	36
4.9	Statistics of performance index for BLX-0, BLX-0.25, BLX-0.5 with type1 sampling	36
4.10	BLX-0.5 with type1 sampling, $K_{\sigma} = 0.3$ with repetition method	38
4.11	BLX-0.5 with type1 sampling, $K_{\sigma} = 0.4$ with repetition method	38
4.12	Statistics of performance index for different K_{σ}	39
4.13	Statistics of the best pre-grasps found from both planners	49
4.14	Execution time of the GA and SA planners	49
4.15	Statistics of the best pre-grasps found from both planners given the same running time	49
4.16	Function evaluations of the GA and SA planners in given time limit	53
4.17	Statistics of the best pre-grasps found from both planners with Q_{final} as the quality measure. Better one indicated in Blue	53

List of Figures

1.1	Robots helping with household work	1
1.2	Popular Robot Hands	2
2.1	Coulomb Friction Model and the <i>friction cone</i> [1]	7
2.2	A generalized flow chart of genetic algorithm	11
3.1	The solution landscape	13
3.2	Eigengrasp movement of Barrett Hand. It has two-dimensional eigengrasp sub-space. EG1,2 refer to the first and second eigengrasp respectively. The number to the right is the corresponding value of this eigengrasp.	14
3.3	BLX- α	16
3.4	BLX- α with six sampling methods	20
3.5	Two different mutation sampling methods with different sampling for BLX-0.5, “m_repeat” means repetition sampling for mutation, “m_truncate” means truncation sampling for mutation	21
4.1	Some of the original objects in the Household Objects and Grasps Data Set	22
4.2	The Graspit! Simulator	23
4.3	The Graphical Interface of the GA grasp planner	23
4.4	The two hand models used in our test and their predefined contact locations	24
4.5	Barrett Hand grasping a glass	29
4.6	Average pre-grasp quality found using BLX-0.5 with type1 sampling, Gaussian Mutation with $K_{\sigma} = 0.2$ and repetition sampling method	37
4.7	The on-line and off-line performance of the genetic algorithm	40

4.8	Visualization of the intermediate process of the GA planner with $p_c = 0.8, p_m = 0.1$. The number on the top left corner denotes the generation. The two red arrows show the two elitists in each generation.	41
4.9	Visualization of the intermediate process of the GA planner with $p_c = 0.8, p_m = 0.01$	42
4.10	Visualization of the intermediate process of the GA planner with $p_c = 0.8, p_m = 0.2$	43
4.11	Visualization of the intermediate process of the GA planner with $p_c = 0.8, p_m = 0.3$	44
4.12	Visualization of the intermediate process of the GA planner with $p_c = 0.7, p_m = 0.1$	45
4.13	Visualization of the intermediate process of the GA planner with $p_c = 0.9, p_m = 0.1$	46
4.14	Best pre-grasps found by GA and SA planners	50
4.15	The corresponding final grasps and the quality	51
4.16	Best pre-grasps found by GA and SA planners with Q_{final} as the quality measure	54
4.17	The corresponding final grasps obtained with Q_{final} as the quality measure	55

Abstract

Vision-based grasp planning can be approached as an optimization problem, where a hand configuration that indicates a stable grasp needs to be located in a large search space. In this thesis, we proposed applying genetic algorithm (GA) to grasp planning of 3D object in arbitrary shapes and any robot hand. Details are given on the selection of operators and parameters of GA. *GraspIt!* simulator [2] is used for implementing the proposed algorithm and as the test environment. A quantitative analysis including the comparison with simple random algorithm and simulated annealing (SA) method is carried out to evaluate the performance of the GA based planner. Both GA and SA grasp planner are tested on different sets of hand-object. And two different quality metrics are used in the planning. Given the same amount of time, GA is shown to be capable of finding a *force-closure* grasp with higher stability than SA.

List of Abbreviations and Symbols Used

K_{σ}	A constant controlling the standard deviation of Gaussian mutation
Q_{final}	A quality metric on the stability of the final grasp
Q_{pre}	Pre-grasp Quality
W_{L_1}	Wrench space constructed using L_1 norm
$W_{L_{\infty}}$	Wrench space constructed using L_{∞} norm
ε	Worst case grasp quality measure
n	Population size
p_c	Crossover probability
p_m	Mutation probability
v	Invariant average case grasp quality measure
BLX-α	Blend Crossover
DOF	Degree of Freedom
EG	EigenGrasp
GA	Genetic Algorithm
GWS	Grasp Wrench Space
p.d.f.	Probability density function
RCGA	Real-Coded Genetic Algorithm
SA	Simulated Annealing Algorithm
STD	Estimated STandard Deviation
TWS	Task Wrench Space

Acknowledgements

I would like to thank my supervisor Dr. Jason Gu for giving me this opportunity to work with him and for all his guidance. I would also like to thank Dr. Yuan Ma and Dr. Williams J. Phillips for being on my committee and for their time. And I would also like to thank Nova Scotia Health Research Foundation (NSHRF), Faculty of Engineering and Department of Electrical & Computer Engineering at DAL for their financial support.

To all my colleagues in the Robotics Lab, I appreciate the time sharing space, sharing thoughts and sharing laughter with you. In no particular order, thank you, Shijie Zhou, for discussing with me on all the courses we took together, the research projects we did together and on ideas that came to my mind “randomly”, for showing me the dedication to research, for all your advice on my research and life; Yuanlong Yu, for your advice on choosing my research topic; Ze Yu, for kindly providing the course materials; Kun Zhan, for being someone I look up to in terms of research ability; Siva Kumar, for bringing so much laughter to our whole lab; Hamidreza Moslehi, for encouraging me when I felt a little uncertain about my research. I wish to say thank you to all my friends at DAL engineering for being a part of my “boring” life at university. Particularly, thank you Xiaoou Mao, for having coffee break and “ice cream cone break” with me - my only entertainment sometimes, for bringing me food and sharing your apartment when I had to work overnight at university, for being so lazy which makes me feel pretty good about myself.

Thank you to all the people lived or are living in the little house at Sherwood Street. You all have been a very important part of my life. Thank you, Roland & Bernice Alexander, for all your help especially when I first came to Canada, you make me feel like having a family here, and this thesis would not have been completed without your input and patience in revising almost all my research paper; Jinyi Liu, for teasing me and giving me pressure with your degree; Peiheng Wu, for often staying up late so that I do not feel lonely when I am working at midnight; Carla, for sharing the happiness of singing; Youngchan, Sujin and Juwon, my dear Korean friends, for bringing so much fun to my life.

Last but not least, to my parents, who have always been my source of inspiration. I am proud of you as much as you said how much you were proud of me. This thesis is for you.

Chapter 1

Introduction

1.1 Motivation

The advancement of robotics has been an important part of the progress of mankind over the past few decades. Today's robots not only find their applications in industry, military, but also start to be more involved in human-centered environments. Several robots have been designed to help people with everyday work, like cleaning the floor, flipping pancakes and even folding towels, as shown in Fig. 1.1. Due to the population aging and the lack of human resources in delivering home care, there has been an increasing need for robots that are capable of assisting people at home, especially elderly or disabled people.

Think about a common task in a home environment: bring me an apple. Before we can deliver it, the first step is to pick up the target object and hold it firmly, that is **grasping**. Grasping is necessary for the robots to complete high-level tasks. Thus it has become a very active field of research. This thesis will look into the problem of grasp planning on 3D objects, to find a way for the robots to be capable of grasping objects in a stable and robust manner.

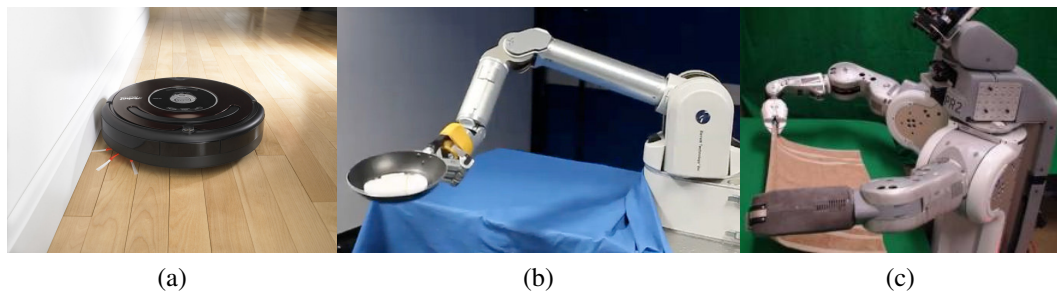


Figure 1.1: Robots helping with household work

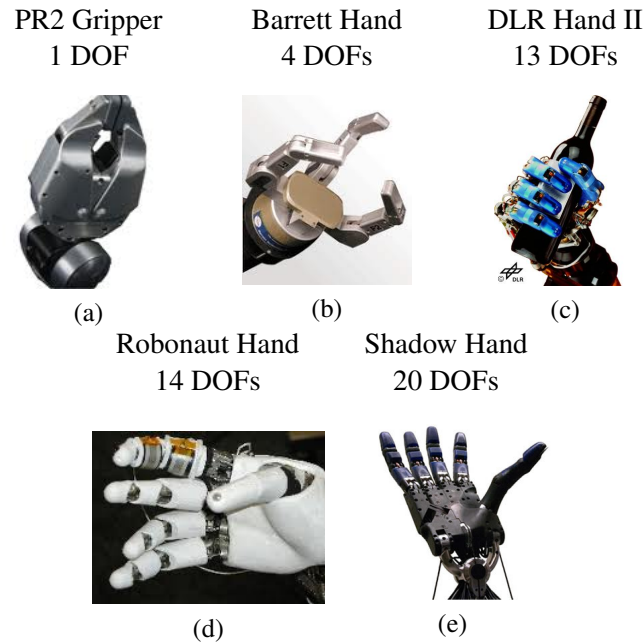


Figure 1.2: Popular Robot Hands

1.2 Previous Research

1.2.1 Different Robot Hand

There has been a large body of work in designing robot hands. Five of the most popular robot hands with different precisions and degree of freedoms (DOFs) are shown in Fig. 1.2. Based on the complexity of the hand, they can be categorized into three types: gripper, dexterous hands and anthropomorphic hands. Gripper is a simple end-effector, with only 1 DOF, like the PR2 Gripper. Dexterous hands have more DOFs and are able to perform some grasping work which requires more dexterity of the hands, e.g., the Barrett Hand. The anthropomorphic hands are delicately designed with the goal of replicating the human hand with similar shape and comparable functionality, like the Robonaut hand and Shadow Hand.

1.2.2 How to Perceive the Environment

Before a robot can plan and execute grasps, it has to be able to perceive the environment, gather some information about the object and potential obstacles. Most sensory information is obtained through tactile and vision feedback.

Vision feedback is the most widely used method. A grasp is planned based on the

geometrical information obtained from an image sensor. People have successfully used vision feedback to plan grasps on 2D objects and 3D objects. Tactile sensor is often used to compensate the vision information of the object. It allows the robot to be aware of the properties of the contact surface and adjust force and torque in real time. And it also can be used when vision information is not available. This makes the grasp more robust in unstructured and cluttered environment. Here we focus on vision-based method only. Note that by saying “vision-based”, we simply mean that we do not rely on any tactile information in grasp planning. And all the data used in this work is from a database that contains the geometry of objects rather than raw data from image sensors. How to utilize tactile sensors for grasp planning is an active research field in its own right. Readers who are interested are referred to [3, 4, 5] for detailed information.

1.2.3 Problem Description

There have been two general methods used in the vision-based robotic grasping research: *synthesis approach* and *heuristic approach* [6]. *Synthesis approach* was adopted when the research was first started decades ago. It dealt with some basic problems in grasping, such as how to evaluate the grasp quality. A grasp is defined in the contact space of the object. Once the grasp quality metrics were defined and became widely used, research had been focused on actual grasping. The intrinsic limitation of this approach that it requires a precise object model becomes the bottleneck to achieve real-time grasping on novel objects. Due to the uncertainty associated with real sensor data and the disturbance from the environment, it is impossible to construct a complete and accurate object model from sensor data. And also it is not the natural way of grasping an object from a human’s perspective.

Heuristic approach is adopted to solve the above problems, where the grasp is planned in the configuration space of the hand. A grasp action starts with “pre-grasp”-a set of hand poses close to, but not in touch with the object-inspired by the fact that humans unconsciously preshape the hand before the actual grasp [7]. It defines the starting posture and approach direction of the hand. After the pre-shape is determined, the hand is moved along the direction toward the object until in contacts. Then fingers are closed to conform to the surface of the object to complete the entire grasp action. The pre-grasps can be obtained in this way: for simple gripper, by some simple heuristic rules based on the object shape or for more dexterous hand, by searching for hand poses that will most likely yield

grasps with good quality.

Both approaches can be considered as optimization problems. But the solution space is too vast to search in an effective manner. And it is discontinuous since there are some surface points or hand configurations that we want to avoid. Also the quality measures for evaluating the stability of a grasp are often non-linear. As a general-purpose optimization method, Genetic Algorithm (GA) is widely used to tackle this type of problem.

In the literature, some work has been done in applying genetic algorithm to grasp planning. A. Chella *et al.* proposed a hybrid method of genetic algorithm and neural network to planar object grasping [8], where a dataset is first generated off-line by using genetic algorithm and then trained by neural networks for the purpose of real-time application. This method only dealt with 2D objects in superellipses shape. N. Daoud [9] used genetic algorithm to find grasps for 3D objects with an LMS mechanical hand. It had not been tested on other hand types and only limited information was given on the performance of the algorithm. S. Mannepalli *et al.* [10] proposed using GA to find good fingertip placements in 2D space of prismatic objects. Sangkhavijit, C. *et al.* [11] applied GA in finding 4-fingered stable grasps from surface points of 3D objects. Mesgari, H. *et al.* [12] worked on a problem where an simple end-effector mounted on a robotic arm grasps a planar object with only one-point contact. GA was adopted to find the best grasping point which maximizes a performance index called MAG.

The above works fall into the category of *synthesis grasp planning*. To the best of our knowledge, although GA has been mentioned as a possible optimization method in the scenario of *heuristic grasp planning* [2], not much work has been done in applying genetic algorithm to this problem. The only relevant work dates back to 1998, when J.J Fernandez *et al.* proposed a genetic approach to find good grasps with three-fingered robot hands [13]. Their work had some limitations. Firstly, they only tested on three-fingered hands. Secondly, only a certain hand-object placement was allowed. Lastly, only some pre-defined 3D objects with regular shapes were tested.

1.3 Objectives and Contribution

GA was often used to deal with the optimization problem in grasp planning. Previous work of applying GA to grasp planning had their limitations as discussed in the last section. In addition, no quantitative result has been addressed in the literature regarding the

performance of the GA based grasp planner and few details have been found on the parameter selection. These problems make it difficult to compare with planners based on other algorithms. To fill the gap of current research and gain a better understanding of GA's applicability in the context of grasp planning optimization, we study the effect of applying GA on *heuristic grasp planning* of 3D objects in arbitrary shapes and different hands in this thesis. We carefully choose the operators and parameters of GA taking into account the characteristics of the solution landscape in this grasp planning problem. And the proposed GA grasp planner is applied on different sets of hand-object including 3D objects in different shapes and hand models with different number of DOFs to examine its performance and robustness. Comparison with other algorithms such as simple random algorithm and simulated annealing algorithm (SA) are done for further evaluation of the GA planner. The execution time of SA and GA planners are considered for a fair comparison of their performance. Two different quality metrics are used in the planning. For both quality metrics, the proposed GA planner is shown to outperform the SA planner, which has been generally used as the optimization method in *heuristic grasp planning*.

1.4 Outline

This thesis is structured as follows. Chapter 2 reviews some of the important background knowledge in the domain of robot grasp planning. Chapter 3 formulates the optimization problem in grasp planning. The components and important concepts in designing a genetic algorithm based grasp planner are described. The selection of operators and parameters of the GA planner are discussed in detail. In Chapter 4, the performance of the GA based planner is examined with quantitative analysis including comparisons with a simple random planner and an SA based planner. Chapter 5 concludes the thesis and presents the future work.

Chapter 2

Background

In this chapter, we review background materials on robot grasping and genetic algorithms. Essential knowledge necessary for understanding the mechanism of grasping was introduced, including the definition of a stable grasp, the quality metrics to evaluate a grasp, and the methodology to deal with the high DOFs of dexterous robot hands. Genetic algorithm is briefly described in the end. Its basic components and working mechanism are reviewed.

2.1 Force-Closure Grasp

First we would like to distinguish between two terms, grasping and manipulation. Grasping refers to the action and state of holding an object firmly using a hand, while manipulation addresses the ability of moving the object in hand, for example, try moving a ping-pong ball between multiple fingers. Our research is in the area of robot grasping. In the literature, there are two types of grasps in general, originally suggested in [14]: *power grasps* and *precision grasps* (also referred to as *fingertip grasps* [15], or *pinch grasp* [16]). *Precision grasps* means holding an objects with fingertips only, offering better dexterity. *Power grasps* on the other hand, allow the use of inner links and palms to provide a more stable grasp with better robustness against external disturbances. It is more suitable on human-made objects [16]. In this thesis, we focus on creating *power grasps* because we aim at developing robot grasping ability in household environment.

Force closure is a notion generally used in evaluating if a grasp is stable. Here we adopt the definition in [2]:

A force-closure grasp can resist all object motions provided that the end-effector can apply sufficiently large forces at the unilateral contacts.

The term *Form-closure* also exists in the literature. It is a similar idea as *force-closure* except that it is achieved only through frictionless contact constraints. In this thesis, when

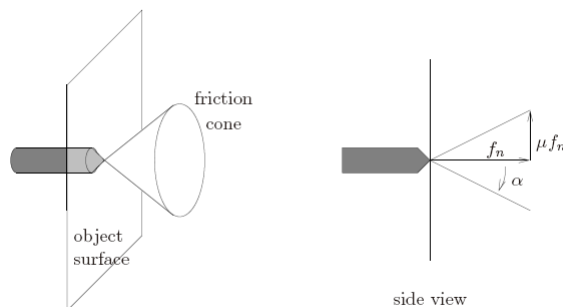


Figure 2.1: Coulomb Friction Model and the *friction cone*[1]

a grasp has *force-closure* on an object it is considered to be a stable grasp.

2.2 Contact Model

The fact that we use force-closure to describe grasp stability indicates that we consider friction of the contact. We use *Coulomb Friction Model*, which can be defined as:

$$|f^t| \leq \mu f^n \quad (2.1)$$

where f^t is the tangent force, f^n is the normal force and μ is the friction coefficient on this contact. This model can be illustrated in Fig. 2.1.

μ determines the shape of the *friction cone*. And the direction of the total force should be inside this cone. **Point contact with friction model** is the commonly used model for frictional contact. This model approximates the *friction cone* with a *friction pyramid*. The boundary of this pyramid consists of several force vectors. The total force is represented as a linear combination of these force vectors. For fingers which are made of compliant materials, such as rubber, the **soft finger model** should be used. In this model, torque at a contact is also defined in a cone around normal. For more details, readers are referred to [1].

2.3 Quality Metrics

A grasp can be described in terms of “the wrench space”, which contains the information of forces and torques that the hand exerts on the object. In the case of grasping a 3D object, the wrench space is 6 dimensional. The particular task that the hand is required to perform

is defined as Task Wrench Space (TWS) and the wrench that a grasp is able to apply on the target object is called Grasp Wrench Space (GWS). If the TWS falls inside the GWS, the grasp is considered to be stable, or *force-closure*. The grasp quality is usually defined as a ratio of the TWS and GWS [17].

In [18], two methods of constructing a unit GWS were introduced. One method tries to bound the sum magnitude of the contact normal forces to 1N, creating a GWS named W_{L_1} , while the other one bounds the maximum magnitude to 1N, creating a GWS named W_{L_∞} .

Given m contacts, let w_1, w_2, \dots, w_m denote the wrenches at these contact by applying unit normal forces, f_1, f_2, \dots, f_m denote the normal forces. The total wrench is:

$$w = \sum_{i=1}^m f_i w_i \quad (2.2)$$

Considering friction, the wrench can be expressed as:

$$W_i = \{w_{i,1}, \dots, w_{i,k}\}$$

$$k : \text{number of force vectors to approximate the friction cone} \quad (2.3)$$

$$W_{L_1} = \text{ConvexHull}\left(\bigcup_{i=1}^m \{W_i\}\right) \quad (2.4)$$

$$W_{L_\infty} = \text{ConvexHull}\left(\bigoplus_{i=1}^m \{W_i\}\right) \quad (2.5)$$

The wrench W_{L_1} is a convex hull over the wrenches at every contact. W_{L_∞} is a convex hull of the Minkowski sum over the wrenches at each contact, which quickly becomes complicated with a large number of contacts. W_{L_1} is chosen in our work because of its computational efficiency .

Two widely used quality measures for evaluating the stability of a grasp are ε quality and ν quality. The former one measures the smallest euclidean distance from the origin to the surface of the GWS, ε , which is a worst case quality measure. To overcome the limitation of the ε quality that it varies with the choice of torque origin, the volume of the unit GWS ν is proposed as an invariant average case quality measure, which is called the volume quality measure [19]. For force-closure grasps, $0 < \varepsilon < 1$, $\nu > 0$. The larger these two quality measures are, the more stable a grasp is.

2.4 Grasp Planning in *Eigengrasp* Space

As mentioned earlier, *heuristic grasp planning* can be approached as an optimization problem: to search for a stable grasp from a high-dimensional space of possible hand configurations. One of the main difficulties is that the search space is high-dimensional due to the large number of hand degree of freedoms (DOFs). Principal component analysis on grasping data [20] reveals that a certain low-dimensional space contributes significantly to the success of the grasp action. Inspired by this, *eigengrasp* [21] was proposed to reduce the dimensionality, where the high-dimensional space was projected to a low-dimensional control space while maintaining sufficient information needed for finding stable grasps.

The first step of the grasp planning is to find a good pre-grasp that is expected to yield a force-closure final grasp. Then the final grasp will be executed and stability will be checked as the second step. The quality metric proposed in [21] is used to evaluate the quality of a *pre-grasp*:

$$Q_{pre} = \sum_{all\ contacts} \delta_i \quad (2.6)$$

where δ_i is a measure of the distance between the desired contact locations on the hand and the object. The contact locations on the hand are selected to be on the fingers and palms in our study to create a power grasp. This quality measure assumes that the closer the hand is from the object, the better potential it has to give a stable power grasp on the object. For details, the reader is referred to [21].

The lower the Q_{pre} , the closer the hand is from the object and the better the pre-grasp is. The optimization goal is to find a *pre-grasp* that can minimize this quality measure. Therefore, the grasp planning problem in the configuration space of the hand can be represented as [21]:

$$\underset{p,w}{\operatorname{argmin}} Q(p,w), \text{ subject to : } p \in \mathfrak{R}^b, w \in \mathfrak{R}^6 \quad (2.7)$$

where $Q(p,w) : \mathfrak{R}^d \mapsto \mathfrak{R}$ is the objective function to be minimized over the variable space of dimension $d = b + 6$. b is dimension of the eigengrasp space, p is a vector representing the hand posture, and w is a vector of the position and orientation of the wrist. In our case, we use axis-angle representation for the 3D rotation. The variables used are listed in Table 2.1. A larger range of Tz is chosen for tall objects.

Table 2.1: Variable List

Property	Name	Definition	Range
Position	Tx	x-coordinate	[-250,250]
	Ty	y-coordinate	[-250,250]
	Tz	z-coordinate	[-250,350]
Orientation	θ	angle between the z-axis and the axis	$[0,\pi]$
	ϕ	angle between the projection of the axis on x-y plane and x-axis	$[-\pi,\pi]$
	α	rotation angle around the axis	$[0,\pi]$
Eigengrasp	EG[0,..,b]	amplitude along the eigengrasp dimension	[-4,4]

2.5 Genetic Algorithm

Genetic Algorithm (GA) is a global optimization algorithm originally developed by John Holland [22]. It is one type of evolutionary algorithm, which draws inspiration from Darwin’s theory of evolution. Solutions of a problem are encoded into a string, which is called “chromosome”. Each member in the string is called “gene”, which can be a binary number or floating-point number. GA starts with an initial “population”, which is a certain number of solutions/chromosomes that typically are randomly chosen from the solution space. At the beginning of every iteration, the chromosomes are evaluated and each will be given a fitness value. The population will evolve by “selection” and “reproduction” at every generation. “Selection” means some chromosomes in this population will be chosen as the “parents” based on their fitness, to go through the reproduction operators, mainly “crossover” and “mutation”. “Crossover”, or more generally “recombination” is a process where the child chromosome is obtained by exchanging the information of parent chromosomes. “Mutation” is applied to one chromosome by making a random change on it. Both crossover and mutation are applied with a probability, p_c and p_m , respectively. The children generated by crossover and mutation will be put into the next generation. The iteration goes on until a termination condition is satisfied. It can either be a certain number of generations has been reached, a good enough solution has been found, or the population reaches a stable state where no better results are expected to be produced in successive generations.

There are a variety of variations of genetic algorithms. A generalized flowchart of the

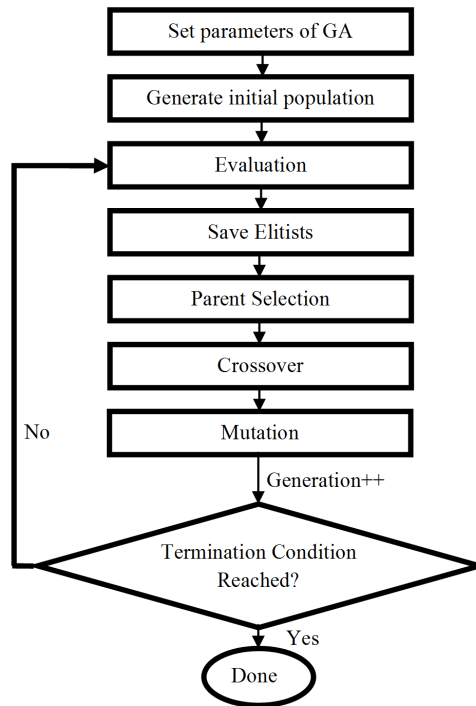


Figure 2.2: A generalized flow chart of genetic algorithm

genetic algorithm can be illustrated in Fig 2.2.

The pre-grasp quality function (2.7) - the objective function in grasp planning - is non-linear, high-dimensional and discontinuous. On the other hand, we do not have a full understanding of the search space. Genetic algorithm is considered to be a suitable algorithm for this type of problem.

The fundamental theorem of GA, also called *Schema Theorem*, is stated as follows:

Short, low-order, above average schemata receive exponentially increasing trials in subsequent generations [23].

“Schemata” is a subset of the chromosome. Schema Theorem explains how GA is able to evolve to a better solution. Note that although in theory, GA is proven to have global convergence, practical applications do not always follow the theory. It is mainly due to the limitation on the hypothetically unlimited population size and unlimited number of iterations. Finite population will inevitably mislead the search to a different solution subspace with what is theoretically predicted [24]. Consequently, instead of converging to a global optimum, the algorithm may end up with a sub-optimal solution.

Chapter 3

Genetic Algorithm on Grasp Planning

In this chapter, we will apply GA on grasp planning. We start with a close look at the search space. The selection of operators and parameters are then investigated in detail. To understand and overcome the sampling bias caused by crossover operators, we test six types of sampling methods.

3.1 Solution Landscape

There have been lots of methods of genetic operators proposed in the past literature. The operators as well as the representation of solutions have to be carefully chosen according to the characteristics of the search space, which is the space of all possible solutions to a problem.

First we need to decide on the encoding for a solution. As discussed in the previous section, the variables in this problem take real value. We naturally adopt floating-point representation. GA that uses floating-point encoding is called Real-Coded Genetic Algorithm (RCGA). It provides higher accuracy and faster speed as opposed to binary-coded GA.

Each chromosome can thus be represented as:

$$chromosome = \langle Tx, Ty, Tz, \theta, \phi, \alpha, EG_0, EG_1, \dots \rangle \quad (3.1)$$

The position and orientation of the hand is defined in terms of the contact space of object. We define the range of the variables so that possible solution space is around the object, as shown in Table 2.1. The solution landscape is illustrated in Fig. 3.1. The red area shows the space taken up by the target object. The *pre-grasp* is considered to be illegal if the hand and the object get in touch. The legal solutions fall in the space outside the object. The closer to the object, the better the pre-grasp is. “Good” denotes the space that gives the best pre-grasps. As it extends outward, the quality goes down, denoted by “Normal”. In the solution space close to the boundary (e.g. when the position variables take values to the minimum or maximum, the hand is far away from the object), the pre-grasps has

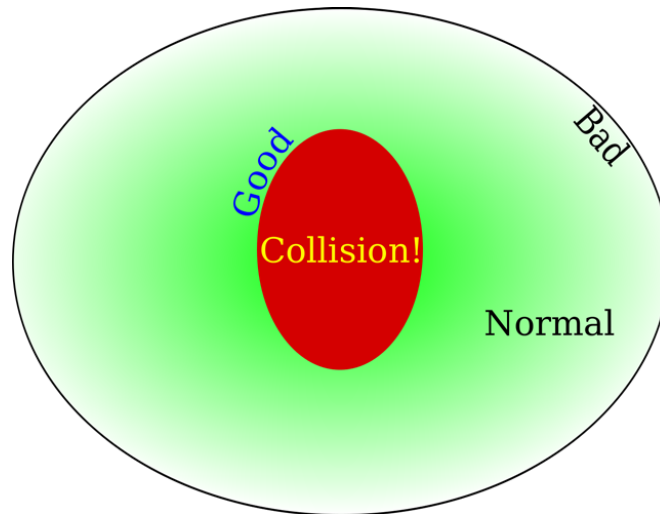


Figure 3.1: The solution landscape

a very low quality, denoted as “Bad”. Although “Bad” pre-grasps are not considered to be illegal, we want to avoid them. “Normal” and “Bad” do not indicate particular quality value. They are just two terms used to informally show the transition trend of the solutions. In the grasp planning, we will only search for solutions that do not collide with the object. That is, if any illegal solution is obtained during any step of the GA, it will be dropped and that GA step will be repeated until a legal solution is found. This makes the solution space discontinuous. Imagine the action of moving the hand from inside the object to outside, there will be a sudden transition from the worst grasp to the best. It addresses one of the difficulties for an optimization algorithm to find the best solution.

If we take a close look at each variable individually, we can find that the good solutions are not distributed evenly in the search space. There is a certain range that is supposed to give a better solution. In order for GA to succeed in solving a problem, we must take into account the problem specific knowledge in designing or choosing the genetic operators. Thus it is necessary to gain a better understanding of these variables.

For the three position variables denoting the position of the the wrist, we want the search to be focused on the center area, where the hand is closer to the object. For the three variables representing the orientation of the hand, variable θ and ϕ encode the axis of rotation. θ means the angle between the z-axis and this axis. ϕ denotes the angle between the projection of this axis on x-y plane and x-axis. And α refers to the rotation angle around the axis. $\theta = 0$ means the palm of the hand is facing upward, while $\theta = \pi$ means the palm

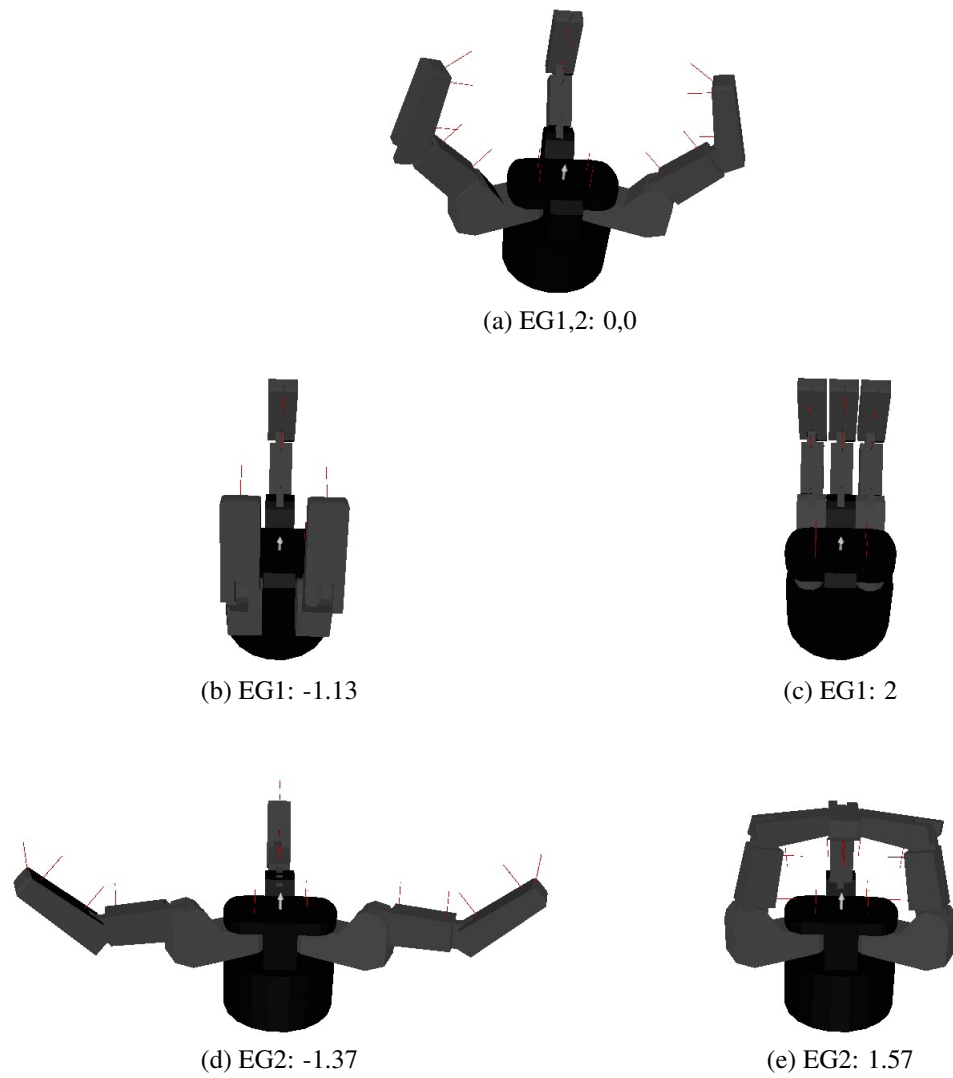


Figure 3.2: Eigengrasp movement of Barrett Hand. It has two-dimensional eigengrasp subspace. EG1,2 refer to the first and second eigengrasp respectively. The number to the right is the corresponding value of this eigengrasp.

facing down. ϕ can be viewed as the projection of the direction of the palm on the x-y plane, the range of which goes from $-x$ direction to x direction. Changing α can be thought of as rotating the hand to find a correct orientation of the fingers while keeping the palm facing the same direction. For the variables that represent the eigengrasp (which indirectly control the joint movement), some example movements along the eigengrasp dimensions of Barrett Hand are shown in Fig. 3.2. There is no preferred range for the orientation and eigengrasp variables. The possible solutions should be searched evenly throughout the entire range.

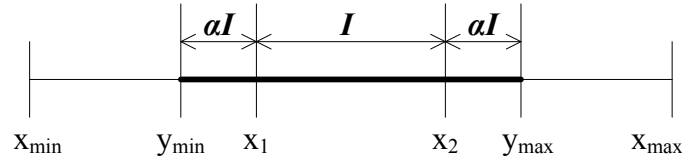
3.2 Operators

In this section, we discuss the details of each component of the GA planner.

The first step is to choose the *parent selection* method, i.e., which chromosome will be chosen from current population to be saved to the mating pool to go through the rest operators. The most popular two selection methods are Roulette Wheel Selection and Tournament Selection.

In Roulette Wheel Selection, the probability of a chromosome of being chosen is proportional to their fitness. The main disadvantage of this method is that chromosomes with low fitness score rarely get the chance to be selected. The much better solutions will dominate the population very fast which leads to premature convergence. Another disadvantage is that it can be very slow to move toward a better chromosome when the entire population have very similar fitness [25]. In Tournament Selection, T_s individuals are chosen to compete with each other and the one with better fitness score wins. T_s stands for tournament size. There are two slightly different implementations: Tournament Selection Without replacement (TSWOR) and Tournament Selection With Replacement (TSWR). In TSWR, candidates for each tournament are randomly chosen, while in TSWOR, an individual does not enter a tournament if it has already been chosen. The best chromosome in a population may have many copies or no copies in the mating pool using TSWR, while the best is guaranteed to have exactly T_s copies in the mating pool in TSWOR [26]. TSWOR is chosen in our implementation, with a tournament size of two.

For crossover operator, we employed blend crossover (BLX- α) operator proposed in [27], which is defined as: Suppose we have two parents x_1, x_2 , children y_1, y_2 are uniformly chosen from the interval $[C_{min} - I\alpha, C_{max} + I\alpha]$ at random, where $C_{min} = \min\{x_1, x_2\}$, $C_{max} = \max\{x_1, x_2\}$, $I = C_{max} - C_{min}$, $\alpha \in [0, 1]$. This can be illustrated in Fig. 3.3. α is normally

Figure 3.3: BLX- α

set to a number bigger than 0, to make the children generated span a slightly larger space than the parents, which from a statistical perspective compensates for the shrinking of the solution space over the generations [28]. The effect that the solution space will be attracted towards a certain area preferred by the search operator is also called the search bias [29]. There will be further discussion in the next section.

Gaussian Mutation is applied as the mutation operator. A new gene value is obtained by adding to the current value a number drawn from a Gaussian distribution $N(0, \sigma)$, where σ is a user-specified parameter [25]. Let

$$\sigma_i = K_\sigma \cdot r_i \quad (3.2)$$

We want to select the parameter K_σ to make sure that the possible solution can cover the full range. Since the value range for a Gaussian distribution is approximately equal to 6σ , we choose $K_\sigma = 0.2$ as an initial value, where r_i is the range of the value of $gene_i$. Larger values of K_σ will be tested in order to find the most appropriate value.

Elitism is often used to prevent the loss of the best chromosome in a population. In our case, two best chromosomes (also called Elitists) up to the present generation are kept and added to the next population. At each generation, the offspring are compared to the elitists. The elitists are replaced with the best two of the offspring if they have better quality and are kept otherwise. In addition, we adopt the *generational model* for survivor selection, i.e., the entire population are replaced by their offspring at each generation. To summarize, the operators are listed in Table 3.1.

3.3 Sampling Method

As discussed earlier, there is an inherent bias caused by the crossover operator, whereby the search will go toward a certain area rather than the whole space. We want this area to contain optimum solutions, or relieve this bias. We also discussed the distribution of the

Table 3.1: Operator List

Operators	Method
Representation	Floating-point Numbers
Parent Selection	TournamentSelectionWithoutReplacement, $T_s = 2$
Crossover	BLX- α
Mutation	Gaussian Mutation
Elitism	Two Elitists, added to the next population
Survivor Selection	Generational Model

good solutions in the search space in the previous section. With this *a priori* information we have, it is necessary to investigate the bias of BLX- α crossover operator to find out if this bias is beneficial for the search.

In BLX- α crossover, children solutions are generated by sampling from an extended area around the parents. In [30], the search bias is examined with three sampling methods. Instead in this section, we will show the bias with six sampling methods.

We consider one dimensional search without loss of generality. This search space is given by

$$X = \{x \ ; \ x_{min} \leq x \leq x_{max}\} \quad (3.3)$$

3.3.1 Type1 sampling

In type1 sampling, we repeat sampling from the range $[y_{min}, y_{max}]$, until the offspring are in the feasible range $[x_{min}, x_{max}]$. All the notations that we use here are consistent with those in Fig. 3.3.

$$y = \begin{cases} y_{min} + u(y_{max} - y_{min}), & \text{if } x_{min} \leq y \leq x_{max} \\ \text{repeat sampling,} & \text{otherwise.} \end{cases} \quad (3.4)$$

where

$$y_{min} = x_1 - \alpha(x_2 - x_1) \quad (3.5a)$$

$$y_{max} = x_2 + \alpha(x_2 - x_1) \quad (3.5b)$$

$$u : \text{uniform random number} \in [0.0, 1.0]. \quad (3.5c)$$

We assume that the parents are uniformly distributed in the space $[x_{min}, x_{max}]$.

3.3.2 Type2 sampling

In type2 sampling, the offspring is generated from a modified space where all the values are feasible:

$$y = y'_{min} + u(y'_{max} - y'_{min}), \quad (3.6)$$

where

$$y'_{min} = \begin{cases} x_{min}, & y_{min} < x_{min}, \\ y_{min}, & \text{otherwise} \end{cases} \quad (3.7a)$$

$$y'_{max} = \begin{cases} x_{max}, & y_{max} > x_{max}, \\ y_{max}, & \text{otherwise} \end{cases} \quad (3.7b)$$

3.3.3 Type3 sampling

The offspring are generated as follows:

$$y = \begin{cases} y'_{min} + u_1(c - y'_{min}), & \text{if } u_2 \geq 0.5, \\ c + u_1(y'_{max} - c), & \text{if } u_2 < 0.5. \end{cases} \quad (3.8)$$

where

$$c = \frac{(x_1 + x_2)}{2} \quad (3.9a)$$

$$u_1, u_2 : \text{uniform random number} \in [0.0, 1.0]. \quad (3.9b)$$

Here, c is the center of the two parents.

3.3.4 Type4 sampling

In type4 sampling, instead of truncating y_{min} and y_{max} as in Type2, we truncate y if it is out of the range $[x_{min}, x_{max}]$:

$$y' = y_{min} + u(y_{max} - y_{min}), \quad (3.10a)$$

$$y = \begin{cases} x_{min}, & \text{if } y' < x_{min}, \\ x_{max}, & \text{if } y' > x_{max}, \\ y', & \text{otherwise} \end{cases} \quad (3.10b)$$

3.3.5 Type5 sampling

Type5 sampling is similar with Type3 except that it truncates the offspring in the last step rather than truncating y_{min} and y_{max} .

$$y' = \begin{cases} y_{min} + u_1(c - y_{min}), & \text{if } u_2 \geq 0.5, \\ c + u_1(y_{max} - c), & \text{if } u_2 < 0.5. \end{cases} \quad (3.11a)$$

$$y = \begin{cases} x_{min}, & \text{if } y' < x_{min}, \\ x_{max}, & \text{if } y' > x_{max}, \\ y', & \text{otherwise} \end{cases} \quad (3.11b)$$

where

$$c = \frac{(x_1 + x_2)}{2} \quad (3.12a)$$

$$u_1, u_2 : \text{uniform random number} \in [0.0, 1.0]. \quad (3.12b)$$

3.3.6 Type6 sampling

Type6 sampling is a variation from Type3. There is no truncation done in this method. The steps are repeated until the offspring is in the feasible range.

$$y = \begin{cases} \begin{cases} y_{min} + u_1(c - y_{min}), & \text{if } u_2 \geq 0.5, \\ c + u_1(y_{max} - c), & \text{if } u_2 < 0.5. \end{cases}, & \text{if } x_{min} \leq y \leq x_{max} \\ \text{repeat sampling}, & \text{otherwise} \end{cases} \quad (3.13)$$

where

$$c = \frac{(x_1 + x_2)}{2} \quad (3.14a)$$

$$u_1, u_2 : \text{uniform random number} \in [0.0, 1.0]. \quad (3.14b)$$

We run the BLX-0.5 operator on the randomly generated parents x_1, x_2 , using the above six sampling methods. Each one is run for 5,000,000 times. The probability density function (p.d.f.) of offspring y generated with each method are shown in Fig. 3.4. BLX-0 and BLX-0.25 with type1 sampling are also included.

From the figure, we can see that the BLX- α operator has an inherent bias towards the center of the search space. Increasing α will reduce this bias as expected, since the

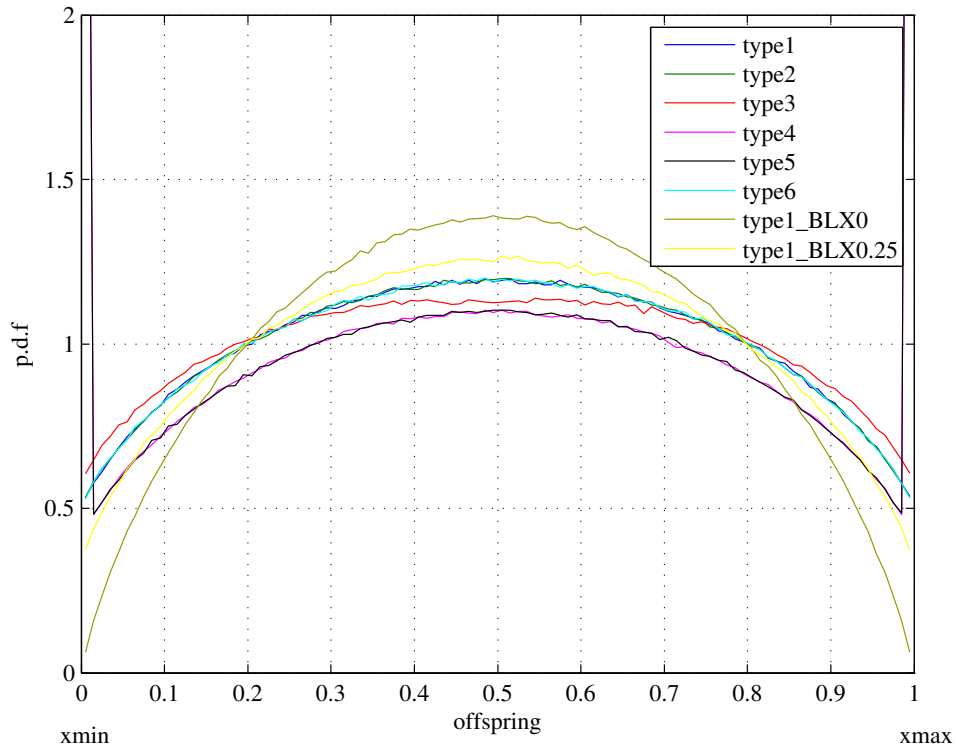


Figure 3.4: BLX- α with six sampling methods

sampling range is extended, leading to an increasing chance for the boundary space to get searched. For BLX-0, almost no solutions fall in the boundary space. This will likely make GA converge too quickly which is referred to as premature convergence. This indicates the population is confined to a small space containing a local optimum in its earlier stage. As the diversity of the population goes down, the search narrows drastically, since crossing over a homogeneous population does not produce solutions radically different from those in the current population.

Comparing these six sampling methods, we can see that the p.d.f of type(1,2,6) are almost identical, and type(4, 5) are similar as well. We will treat them as equivalent. Type3 sampling produces solutions more evenly distributed in the search space. Type(1,2,6) has highest p.d.f. on the center while the boundary gets a lowest p.d.f. The solutions of type(4,5) have a very high p.d.f near the boundary because of the truncation. This leads to the lowest p.d.f in the remaining search space. We would prefer type(4,5) only when the global optimum is very close to the boundary of the search space. Type3 sampling would be the one preferred if no *a priori* information is known about the solution space because

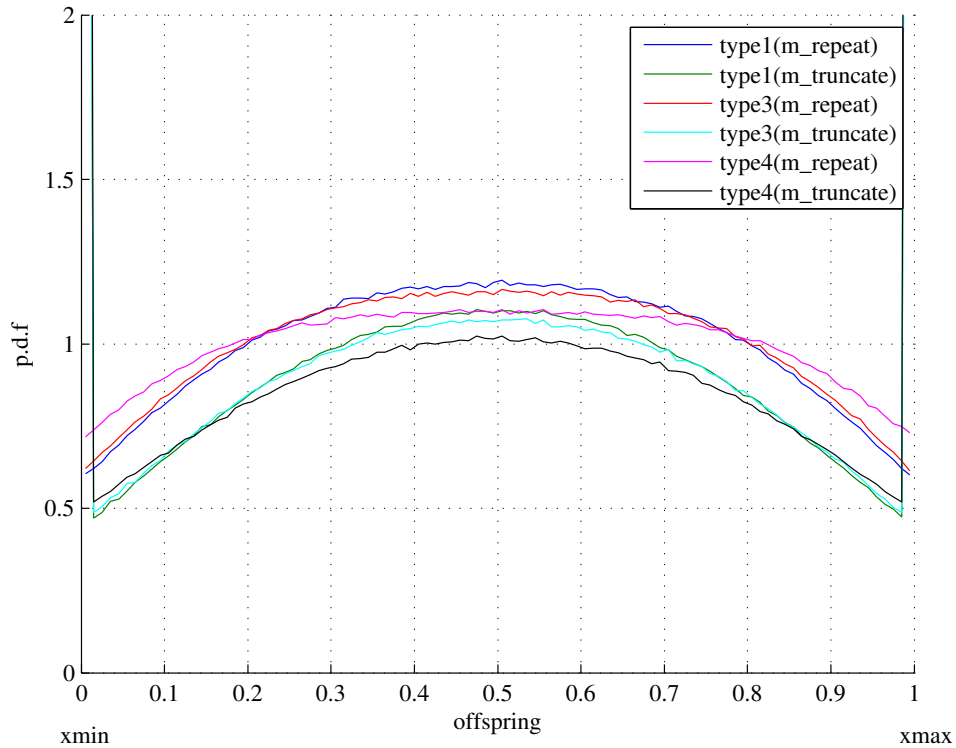


Figure 3.5: Two different mutation sampling methods with different sampling for BLX-0.5, “m_repeat” means repetition sampling for mutation, “m_truncate” means truncation sampling for mutation

of its less bias.

Note that the above probability density functions are generated empirically. For the theoretical proof on the p.d.f of offspring produced by BLX operator, readers are referred to Appendix A in [29].

In the above discussion, we focused on crossover operator only. In fact, in the implementation of Gaussian Mutation, there also can be two sampling methods: one is repeating until the children are in the feasible space, the other is truncating the children to the limit of the feasible range. The effect of these two sampling methods on mutations are illustrated in Fig. 3.5. We call these two methods the “repetition method” and “truncation method” respectively. The solutions of the truncation method concentrate on the boundary while other areas are less likely to be searched as opposed to the repetition method.

Chapter 4

Results and Discussion

In this chapter, we will run the Genetic Algorithm based planner with different parameters to test its capabilities and show the process of parameter tuning. The performance of the GA-based planner will be analyzed and compared with planners based on Simulated Annealing Algorithm and Random Algorithm. The best pre-grasps found by the planner will be saved. The corresponding final grasps will also be executed for a complete analysis.

4.1 Implementation

The algorithm is implemented in C++ under a modified version of *GraspIt!* [2] which runs in ROS framework [31]. All the tests are performed in this modified *GraspIt!* simulator, on a desktop computer with a AMD Athlon II Quad-core 2.9Ghz CPU and 6G RAM. *GraspIt!* is a widely used open-source robot grasping simulator, as shown in Fig. 4.2. It provides an interactive environment for the user to do the grasp planning on any given hand model. Internally, it uses Qhull [32] to build the convex hull of the wrench space. Then the grasp quality ϵ and v can be obtained. *GraspIt!* can serve as a test bed for new robot hand design, grasp planning algorithms. The objects used for the tests are imported from the *Household Objects and Grasps Data Set* [33]. Original objects in this database are shown in Fig. 4.1.



Figure 4.1: Some of the original objects in the Household Objects and Grasps Data Set

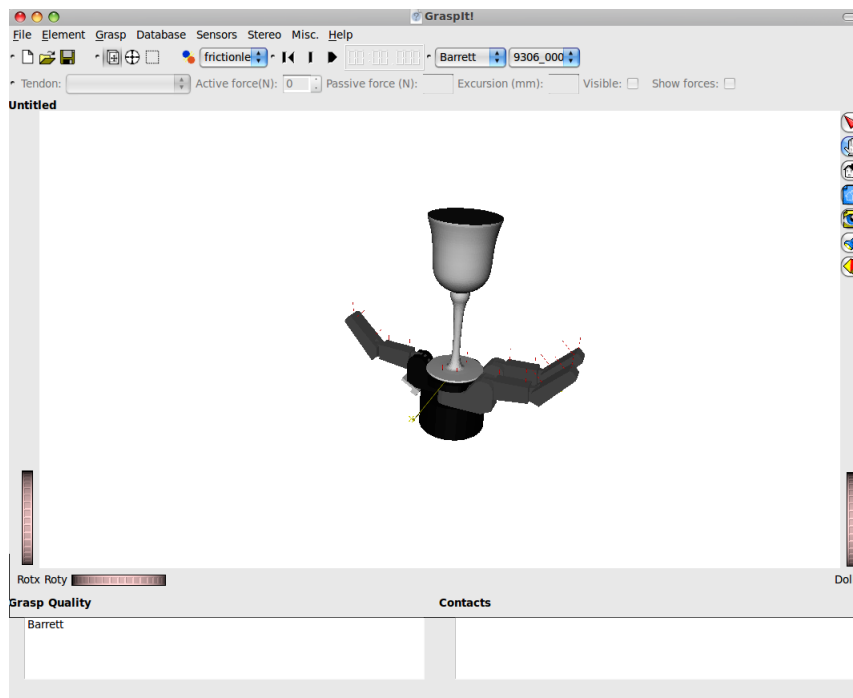


Figure 4.2: The Grasplit! Simulator

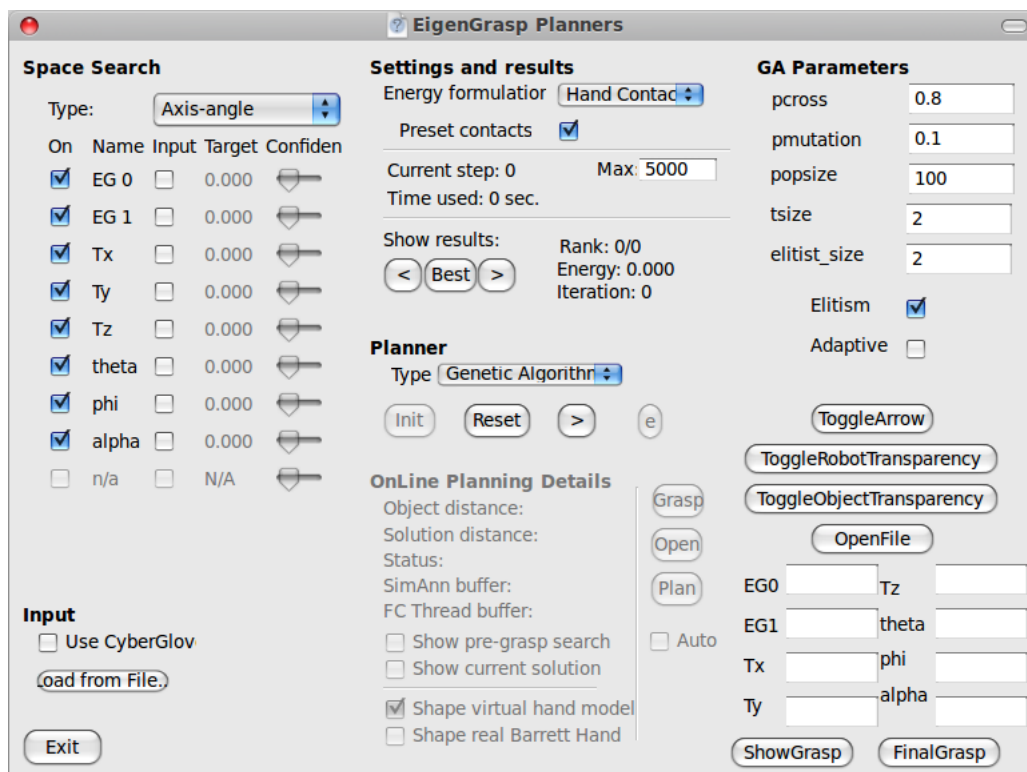
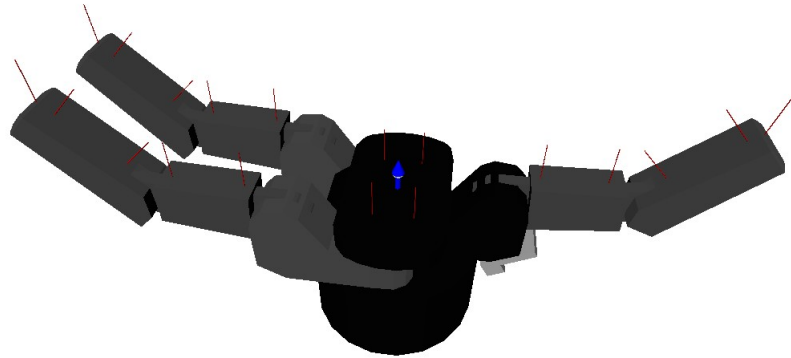
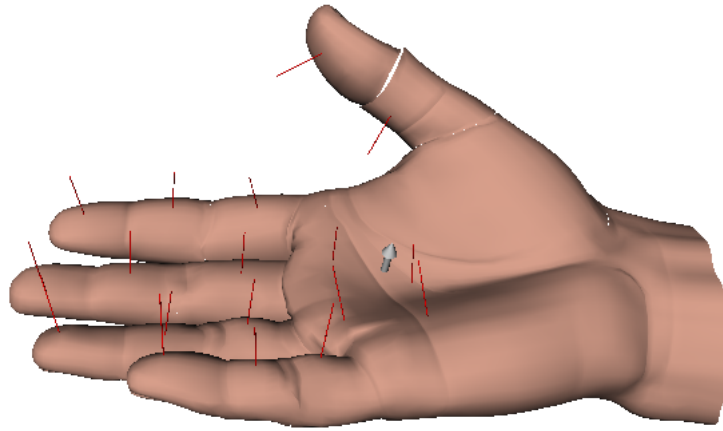


Figure 4.3: The Graphical Interface of the GA grasp planner



(a) Barrett Hand



(b) Human Hand

Figure 4.4: The two hand models used in our test and their predefined contact locations

Table 4.1: Dimension of the search space

Hand Model	DOFs and Eigengrasp Space	GA Encoding Length
Barrett	4DOFs \mapsto 2 EG	8 genes
Human Hand	20DOFs \mapsto 6 EG	12 genes

The interface for starting GA grasp planner and setting parameters is shown in Fig. 4.3.

The Barrett Hand and the Human Hand model released with the *GraspIt!* simulator are employed in this study. The DOFs and the eigengrasp (EG) space of the hand models are listed in Table. 4.1. We choose these two hand models because they are good representation

of the type of hand they belong to. Barrett Hand is a simple dexterous hand and human hand model can be thought as an anthropomorphic hand. Also the difference in their DOFs are helpful in evaluating the optimization algorithm. The contact locations defined for both hands are shown in Fig. 4.4, denoted with red lines.

The materials of all the objects used in our study are assumed to be wood. For the two hand models, the palm of the Barrett Hand is considered to be plastic and the fingers are wood. For Human Hand, the fingertips are rubber and all the remaining parts are plastic. The contacts with the fingertips of the human hand are analyzed by using the **soft finger model**, while all the other contacts are constructed using **point contact with friction model**.

Before we start to carry out the tests of grasp planning, we list our assumptions here:

- no gravity is considered (the object is very light)
- all the objects are rigid
- a 3D model of the grasped object is available(for evaluating the quality of the grasps)

The genetic algorithm used in our implementation is laid out as in Algorithm 1. The hand state is encoded into a chromosome. A population of chromosomes evolve at each generation by applying *crossover* and *mutation* operators to the parents from the mating pool, which keeps the parents selected by the *parent selection* operator. All the parents are saved in the mating pool. The quality of each chromosome is calculated by Eq. 2.6. n stands for the population size. The extra two slots in the population are used to keep Elitists. The best solutions over generations are saved in the “BestChromList”. *flip* is a function with a boolean return value. It returns *true* with the specified probability. Random number is used in almost every component of GA: initialization, selection, crossover and mutation. There is a limitation of the system supplied random number generator “rand()”, that there is sequential correlation in successive calls to the generator. To guarantee that the performance of GA is not affected by the weakness of “rand()”, we used the random number generator of L’Ecuyer with Bays-Durhan shuffle and added safeguards that were described in [34] and implemented in [35]. The period of it is 2×10^{18} .

Algorithm 1 Genetic Algorithm on Grasp Planning

```

{Generate initial population with  $(n + 2)$  chromosomes}
for all chromosome do
    A random number for each gene within the range
end for
Evaluate the quality of each chromosome
Save two Elitists
Generation = 1

while Generation  $\neq$  MaxGeneration do
    clear newPop
    {select  $(n + 2)$  parents from currentPop, save into the mating pool }
    tournamentSelectionWithoutReplacement(currentPop)
    for  $i \leq n$  do
        parent1 = currentPop.mat(i)
        parent2 = currentPop.mat(i+1)
        {crossover}
        if flip(crossoverprobability) then
            crossover(parent1,parent2)
        end if
        {mutation}
        for all Genes in parent1,parent2 do
            if flip(mutationprobability) then
                Gaussian Mutation
            end if
        end for
        Evaluate the quality of parent1,parent2
        Add parent1,parent2 to the newPop
         $i = i + 2$ 
    end for
    continue on the next page
  
```

```
{Add Elitists to newPop}
newPopElitist1 = best individual of the newPop
newPopElitist2 = second best of the newPop
if Q(newPopElitist1) < Q(currentElitist1) then
    Add newPopElitist1 to the newPop
    newPopElitist2 = newPopElitist1
else
    Add currentElitist1 to the newPop
end if
if Q(newPopElitist2) < Q(currentElitist2) then
    Add newPopElitist2 to the newPop
else
    Add currentElitist2 to the newPop
end if

{Elitists are the last two individuals in the newPop}
newPopElitists = newPop(n),newPop(n + 1)
currentPopulation = newPopulation
currentElitists = newPopElitists
if Q(currentElitists) < Q(worst_BestPool) then
    insert newPopElitists into BestChromList
    worst_BestPool= worst quality in BestChromList
end if
Generation ++
end while
```

4.2 Parameter Tuning

Given particular GA operators, parameters tuning is crucial for achieving good performance. Three parameters are typically considered:

- n : Population size
- p_c : Probability of crossover
- p_m : Probability of mutation

In our case, we also decide on which sampling method to use. And we will tune α in the BLX operator and K_σ in the Gaussian Mutation operator.

It is suggested that good solutions can be obtained at a high crossover probability, a low mutation probability, and a moderate population size [36]. However, there is no general theory on parameter selection. The optimal parameters are generally problem-dependent. It may differ with the type of problem and operators. And the importance of crossover versus mutation has always been debated, directly determining the range of the parameter. In practice, parameters are often empirically tuned until satisfactory results are obtained.

To simplify the tuning process we will start with some general recommendations of parameters: $n = 30 - 100$, $p_c = 0.5 - 1.0$ and $p_m = 0.01 - 0.1$. On the other hand, in order to gain a better understanding of this optimization problem in grasp planning, we are not restricted to these recommendations: some different sets of parameters will be tested, with a higher mutation probability p_m .

A larger population size generally leads to a better solution because it contains more diverse schemata, i.e., a larger search space is explored [23]. But increasing the population size is not always useful. Firstly, GA will slow down because more computing power is required. Secondly, the population size will reach a limit when increasing it does not add diversity to the search.

The role of crossover is a main driving force to exchange information between chromosomes to exploit the search space. A high p_c may lead to a faster convergence speed, which is usually not desired, because that means less search space will be explored and consequently the global optimum is less likely to be found. A low p_c on the other hand should also be avoided. With the existence of mutation, the chromosome may be changed

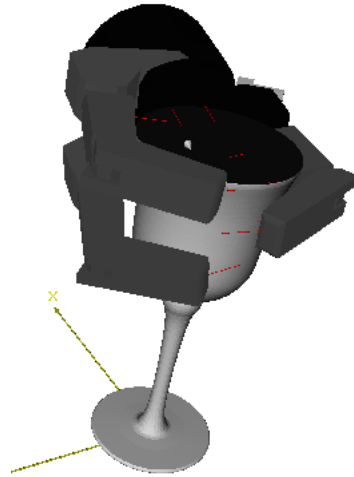


Figure 4.5: Barrett Hand grasping a glass

even before the schemata in them gets the chance to be exchanged with others, i.e., the available information is not fully exploited.

The purpose of mutation is to maintain diversity of the population and prevent premature convergence. If the p_m is too low, the population will converge quickly to a local optimum. However, p_m cannot be too high either, otherwise, it will destroy the useful schemata and make it resemble a random search. To tune the parameters, the program is executed with a Barrett Hand grasping a glass, as shown in Fig. 4.5.

Some preliminary tests show that the GA planner usually does not produce better solutions after 5,000 generations with a population size of 50 or 100. Optimization with each set of parameters is performed over 5,000 generations and the best pre-grasp found is saved as well as the running time. To account for the stochastic nature of GA, each test is repeated five times. The best pre-grasp qualities are taken as the average from the five runs, denoted as “Average” in the tables. We first compare the performance between a population size of 50 and 100. BLX-0.5 crossover with type1 sampling method, and $K_\sigma = 0.2$ are used in this comparison. We test the planner using different combinations of p_c and p_m . The result is shown in Table 4.2. Note that in this table, we also show the best pre-grasp quality found from the five runs, which is put in the parentheses next to “Average”. “STD” is the estimated standard deviation. The best average solution achieved at different population sizes are highlighted in red, while the best solution is marked in blue. The following results will be presented in the same format.

Table 4.2: Different population size, BLX-0.5(Type1 sampling), $K_{\sigma} = 0.2$ (repetition sampling)

n=50		(a) $n = 50$																	
		0.1				0.2				0.3				0.4				0.5	
P_c		Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD		
0.5	32.883(28.259)	4.101	25.329(20.009)	5.543	19.996(14.053)	7.742	16.586(15.419)	1.141	18.172(15.51)	2.407	18.603(15.393)	3.48							
0.6	29.281(23.485)	6.369	20.061(13.743)	5.864	17.679(14.352)	5.67	16.798(15.816)	0.935	19.042(17.282)	1.742	18.533(15.756)	2.56							
0.7	33.775(29.789)	2.351	26.842(19.476)	4.228	18.235(14.728)	6.283	18.824(16.742)	2.355	19.045(16.048)	5.272	21.711(16.628)	5.883							
0.8	24.754(17.299)	7.909	20.181(13.628)	8.637	17.875(14.506)	7.004	19.664(15.122)	5.484	20.005(17.44)	2.735	20.926(18.041)	2.476							
0.9	31.993(22.941)	5.713	28.151(27.448)	0.596	16.768(14.922)	3.017	18.008(15.673)	2.654	20.087(15.994)	3.02	19.03(15.875)	1.958							
1	28.706(18.665)	5.788	18.036(13.757)	5.565	19.602(15.164)	6.145	18.751(15.966)	3.768	21.901(19.411)	4.781	20.673(19.145)	1.39							
n=100		(b) $n = 100$																	
		0.1				0.2				0.3				0.4				0.5	
P_c		Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD		
0.5	26.815(21.408)	5.349	17.982(14.626)	6.821	16.098(15.145)	0.859	18.69(16.568)	1.673	18.815(16.443)	2.304	18.978(16.095)	2.083							
0.6	31.07(22.53)	5.196	24.997(17.697)	6.166	16.846(15.048)	2.133	18.141(15.039)	1.796	18.976(15.058)	2.62	20.344(19.419)	0.979							
0.7	26.915(15.918)	6.81	25.668(14.885)	6.079	17.726(16.264)	1.363	21.124(17.531)	4.891	18.792(16.916)	2.257	20.068(17.475)	1.723							
0.8	26.752(19.639)	5.692	16.96(13.678)	6.629	18.327(16.909)	1.687	20.306(17.657)	2.414	21.594(19.206)	2.083	20.853(17.717)	3.204							
0.9	20.189(15.906)	5.213	23.235(14.039)	8.149	15.828(15.428)	0.259	19.953(16.827)	2.385	19.995(18.763)	0.973	22.74(20.27)	2.112							
1	21.723(14.738)	6.912	15.64(13.988)	2.051	19.037(16.306)	3.207	19.415(17.549)	1.481	21.136(18.151)	2.608	23.739(21.234)	1.622							

We can see that with a population size $n = 100$, the planner gives a better average best solution at $p_c = 1.0, p_m = 0.1$ compared to the one at $n = 50, p_c = 0.6, p_m = 0.2$. For some parameter sets, a larger population shows a better consistency in finding a good solution in multiple runs. The column $p_m = 0.2$ in both tables is a good example. Although $n = 50$ is better than $n = 100$ in terms of the best solution found in all the runs, $n = 100$ wins when taking the average of different runs. This suggests that the planner is less susceptible to local optimum with $n = 100$. If we take the average of all the values in the whole table, it will be 21.57(17.597) for $n = 50$ and 20.707(17.002) for $n = 100$ in the format of “Average(Best)”. The larger population size overall yields better results for different tested parameters. A population size of 100 is used throughout the following parameter tuning process.

We continue testing using the two sampling methods for mutation operator, with a $K_\sigma = 0.2$. Results are shown in Table. 4.3 and 4.4. With either type1 or type4 BLX-0.5 crossover in use, the repetition method is superior to truncation method in every aspect: average best, best, and standard deviation. This result was as expected. According to Fig. 3.5, the truncation method has a strong bias toward the boundary. This bias acts on the three position variables, leading the search to the space far away from the object and thus degrades the performance of GA. The repetition method has a smaller bias and this bias concentrates toward the center where good solutions can be found. The promising areas will be more likely to be searched over.

Repetition sampling method for the mutation operator will be used in all the following tests. Next, we will compare the six sampling methods for BLX- α crossover operator and tune the α . We want to see how the bias indicated on Fig. 3.4 impacts the performance of GA planner. Since type1 and type4 sampling of BLX-0.5 are already tested and type1=type2=type6 and type4=type5 as previously noted, we will only run the test on type3 for BLX-0.5. Results are listed in Table. 4.5. The best average qualities obtained by different sampling methods appear to be very close. To avoid a wrong inference due to the stochastic factor, we evaluate the sampling methods based on the overall performance in the whole table with different p_c and p_m . The performance index for all six types are summarized in Table 4.8. “Average” is the best average solution of all p_c, p_m . “Best” is the best solution from five runs of all p_c, p_m . “Average of all parameters” means taking the average of the average solutions of all p_c, p_m . “Average best of all parameters” means

Table 4.3: BLX-0.5 with type1 sampling, $K_\sigma = 0.2$ with different sampling methods for mutation

(a) Repetition Method												
Repetition Sampling	p_m											
	0.01		0.1		0.2		0.3		0.4		0.5	
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
0.5	26.815(21.408)	5.349	17.982(14.626)	6.821	16.098(15.145)	0.859	18.69(16.568)	1.673	18.815(16.443)	2.304	18.978(16.095)	2.083
0.6	31.07(22.53)	5.196	24.997(17.697)	6.166	16.846(15.048)	2.133	18.141(15.039)	1.796	18.976(15.058)	2.62	20.344(19.419)	0.979
0.7	26.915(15.918)	6.81	25.668(14.885)	6.079	17.726(16.264)	1.363	21.124(17.531)	4.891	18.792(16.916)	2.257	20.068(17.475)	1.723
0.8	26.752(19.639)	5.692	16.96(13.678)	6.629	18.327(16.909)	1.687	20.306(17.657)	2.414	21.594(19.206)	2.083	20.853(17.717)	3.204
0.9	20.189(15.906)	5.213	23.235(14.039)	8.149	15.828(15.428)	0.259	19.953(16.827)	2.385	19.995(18.763)	0.973	22.74(20.27)	2.112
1	21.723(14.738)	6.912	15.64(13.988)	2.051	19.037(16.306)	3.207	19.415(17.549)	1.481	21.136(18.151)	2.608	23.739(21.234)	1.622
(b) Truncation Method												
Truncation Sampling	p_m											
	0.01		0.1		0.2		0.3		0.4		0.5	
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
0.5	29.838(26.385)	3.276	19.383(14.346)	6.88	17.393(14.211)	6.832	22.401(14.897)	7.253	19.059(15.392)	4.117	25.848(16.067)	9.172
0.6	24.247(17.537)	6.977	26.105(13.77)	6.902	22.98(17.4)	6.669	18.2(16.96)	0.833	23.628(20.017)	6.109	21.934(17.689)	4.349
0.7	24.181(17.126)	5.305	26.007(14.44)	6.519	25.511(15.523)	6.396	26.06(15.577)	6.305	23.426(16.299)	8.941	24.557(18.605)	5.789
0.8	22.294(14.687)	8.622	29.038(27.531)	0.851	27.676(17.294)	5.831	25.184(17.09)	6.868	23.336(16.22)	6.814	22.259(16.258)	6.119
0.9	29.148(22.814)	3.637	25.008(13.515)	6.563	22.895(18.001)	5.915	21.126(19.266)	1.678	23.18(19.054)	6.681	27.662(21.134)	4.526
1	25.315(17.732)	6.915	25.624(13.565)	6.783	23.632(18.907)	5.29	22.344(18.64)	4.745	28.775(23.623)	4.553	28.511(20.138)	5.116

Table 4.4: BLX-0.5 with type4 sampling, $K_\sigma = 0.2$ with different sampling methods for mutation

(a) Repetition Method														
Repetition Sampling	pm													
	0.01	0.1			0.2			0.3			0.4			0.5
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
0.5	27.64(16.072)	8.124	24.123(14.248)	9.261	15.358(14.082)	1.45	27.991(21.086)	4.089	20.185(15.557)	6.901	26.132(17.31)	6.203	26.132(17.31)	6.203
0.6	23.958(15.613)	8.223	26.998(17.941)	6.649	18.53(14.286)	3.892	26.086(17.861)	6.331	26.624(18.404)	5.979	23.393(18.756)	5.455	23.393(18.756)	5.455
0.7	26.571(19.625)	6.69	16.55(13.496)	6.168	18.742(15.168)	5.616	29.674(21.38)	4.721	28.184(18.265)	6.442	29.171(20.246)	5.022	29.171(20.246)	5.022
0.8	26.732(15.418)	8.093	27.04(18.548)	4.756	16.068(13.943)	1.361	25.934(14.92)	8.25	28.173(18.245)	6.053	29.241(19.111)	6.04	29.241(19.111)	6.04
0.9	25.555(13.861)	7.305	29.09(27.446)	1.791	24.022(18.91)	6.282	27.87(20.301)	6.316	31.441(21.493)	5.606	26.666(19.481)	6.259	26.666(19.481)	6.259
1	32.001(29.763)	2.137	26.314(13.545)	7.148	24.223(16.02)	6.186	24.457(17.587)	5.927	30.023(20.82)	6.27	32.193(25.747)	3.893	32.193(25.747)	3.893
pc														

(b) Truncation Method														
Truncation Sampling	P_m													
	0.01	0.1			0.2			0.3			0.4			0.5
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
0.5	26.676(16.957)	6.79	25.652(14.558)	6.824	20.659(14.296)	7.225	27.991(21.086)	4.089	20.185(15.557)	6.901	26.132(17.31)	6.203	26.132(17.31)	6.203
0.6	27.639(18.761)	6.626	22.425(13.489)	7.98	26.797(16.075)	6.067	26.086(17.861)	6.331	26.624(18.404)	5.979	23.393(18.756)	5.455	23.393(18.756)	5.455
0.7	30.099(19.049)	7.027	26.019(14.174)	6.774	24.693(15.651)	7.838	29.674(21.38)	4.721	28.184(18.265)	6.442	29.171(20.246)	5.022	29.171(20.246)	5.022
0.8	29.651(14.877)	8.538	27.452(23.278)	2.491	29.699(28.321)	0.933	25.934(14.92)	8.25	28.173(18.245)	6.053	29.241(19.111)	6.04	29.241(19.111)	6.04
0.9	31.611(29.023)	1.978	27.854(19.441)	4.714	26.936(19.335)	6.153	27.87(20.301)	6.316	31.441(21.493)	5.606	26.666(19.481)	6.259	26.666(19.481)	6.259
1	29.67(19.651)	5.659	30.858(29.432)	2.609	27.983(18.494)	5.767	24.457(17.587)	5.927	30.023(20.82)	6.27	32.193(25.747)	3.893	32.193(25.747)	3.893
P_c														

Table 4.5: BLX-0.5 with type3 sampling, $K_\sigma = 0.2$

BLX-0.5,type3	P_m											
	0.01		0.1		0.2		0.3		0.4		0.5	
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
0.5	25.226(20.349)	4.735	23.588(15.289)	5.729	15.6(14.94)	0.677	20.466(15.499)	6.008	18.155(15.433)	2.371	20.26(18.727)	1.733
0.6	32.012(25.606)	4.495	19.588(13.67)	6.668	16.747(15.711)	1.27	20.049(17.016)	2.479	20.262(16.335)	3.151	20.288(18.394)	1.476
0.7	24.274(13.853)	9.149	20.853(13.724)	7.946	20.661(17.029)	5.724	16.29(15.552)	0.635	21.329(18.056)	2.93	19.122(17.901)	1.438
0.8	30.564(20.631)	7.14	20.043(13.56)	8.578	18.841(15.531)	2.651	17.137(15.649)	1.164	23.197(19.115)	4.09	22.103(20.125)	1.626
0.9	22.248(14.389)	7.792	21.718(13.706)	7.588	19.204(15.275)	3.185	22.461(17.795)	6.097	21.484(20.068)	0.933	22.801(18.071)	3.029
1	28.761(20.058)	5.78	20.244(14.1)	8.271	20.007(16.213)	3.269	20.917(18.47)	2.079	23.363(21.424)	1.905	23.561(21.804)	1.387

 P_c

Table 4.6: BLX-0 with type1 sampling, $K_{\sigma} = 0.2$

BLX-0	P_m											
	0.01		0.1		0.2		0.3		0.4		0.5	
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
0.5	32.189(23.236)	6.809	22.804(15.936)	8.893	18.015(14.971)	6.536	18.09(16.383)	1.775	19.793(17.655)	1.513	21.952(18.451)	1.97
0.6	27.746(18.896)	6.988	23.614(16.038)	6.713	15.87(14.781)	1.198	20.464(16.801)	5.314	18.81(15.448)	2.182	20.372(16.902)	2.862
0.7	25.735(19.21)	6.868	22.391(16.442)	5.765	16.106(15.519)	0.455	20.777(17.232)	5.039	21.258(16.942)	2.93	20.565(16.712)	3.423
P_c 0.8	28.397(24.125)	6.961	21.269(16.518)	5.049	18.174(14.207)	6.23	22.393(19.868)	4.707	19.522(17.962)	1.583	23.069(17.813)	6.917
0.9	23.5(15.79)	5.25	20.975(15.496)	5.615	16.806(15.792)	0.775	21.574(20.546)	0.628	22.618(18.038)	5.037	22.266(17.98)	3.013
1	27.493(22.173)	4.286	29.178(25.358)	2.304	20.538(16.73)	5.839	20.119(18.59)	2.131	22.169(19.576)	2.124	25.488(21.11)	3.213

Table 4.7: BLX-0.25 with type1 sampling, $K_{\sigma} = 0.2$

BLX-0.25	P_m											
	0.01		0.1		0.2		0.3		0.4		0.5	
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
0.5	31.766(23.008)	7.786	22.193(15.683)	6.404	15.124(14.519)	0.8	19.742(17.729)	1.541	20.224(16.191)	3.018	21.06(16.234)	3.799
0.6	26.901(20.273)	4.439	16.655(13.911)	2.349	20.966(13.978)	8.105	18.899(17.527)	2.016	23.579(18.917)	4.245	22.875(20.389)	2.689
0.7	22.792(17.104)	5.056	20.791(16.117)	4.888	18.087(14.303)	6.675	19.137(15.857)	3.075	26.369(22.278)	3.976	20.266(17.898)	2.406
P_c 0.8	23.805(18.244)	4.65	20.03(14.218)	6.312	17.822(14.257)	6.987	20.089(17.401)	2.789	22.738(19.635)	2.478	22.142(16.348)	3.692
0.9	29.96(17.578)	6.999	21.674(15.071)	6.44	18.524(14.336)	5.7	21.25(20.574)	0.862	21.837(19.214)	3.166	20(16.426)	3.733
1	25.937(21.942)	4.666	22.302(14.88)	5.154	16.363(15.141)	1.264	20.985(16.866)	2.805	28.273(23.983)	3.55	24.98(18.377)	4.569

Table 4.8: Statistics of performance index for type1,3,4 sampling of BLX-0.5

	Type(1,2,6)	Type3	Type(4,5)
Average	15.64	15.60	15.358
Best	13.678	13.56	13.496
Average of all parameters	20.707	21.484	25.638
Average best of all parameters	17.002	17.196	18.182

Table 4.9: Statistics of performance index for BLX-0, BLX-0.25, BLX-0.5 with type1 sampling

	BLX-0	BLX-0.25	BLX-0.5
Average	15.87	15.124	15.640
Best	14.207	13.911	13.678
Average of all parameters	22.003	21.837	20.707
Average best of all parameters	17.923	17.4	17.002

taking the average of the best solutions of all p_c, p_m .

In Table 4.8, the performance in the top two rows are quite similar. From the bottom two rows which is the overall performance, we conclude that the order of performance is $\text{type1} > \text{type3} > \text{type4}$, with type1 being the best. Bad performance of type4 is likely due to a strong bias toward the boundary. Comparing type1 with type3, a little more bias towards the center seems to be helpful in the performance. This bias leads the search toward the center space where a better pre-grasp is more likely to be found, while the population diversity is maintained with enough solutions from the boundary because of a moderate p.d.f. near the boundary.

Results obtained with BLX-0 and BLX-0.25 operators are listed in Table 4.6 and 4.7. And the performance index for different α are summarized in Table 4.9. The BLX-0.5 operator outperforms the other two in three out of four performance indexes. The overall performance is shown in the bottom two rows, which indicates that $\text{BLX-0.5} > \text{BLX-0.25} > \text{BLX-0}$.

Recalling that for the three position variables, the bias toward the center is desired, where the hand is closer to the object. But a strong bias may cause premature convergence. For other variables, we want the search to evenly cover the entire space. In this case, BLX-0.5 with type1 sampling provides a better balance. As a result, we will use BLX-0.5 as the

crossover operator.

After determining the sampling methods to use and the α of the BLX operator, we will tune the parameter K_σ . The result of $K_\sigma = 0.2, 0.3, 0.4$ is presented in Table 4.3, 4.10 and 4.11. The impact of these three K_σ are also evaluated based on the overall performance, listed in Table 4.12. $K_\sigma = 0.2$ outperforms the other two in terms of the average of the solutions obtained from all the parameters p_c and p_m . It will be used throughout later tests. The average pre-grasp quality found using BLX-0.5 with type1 sampling, Gaussian

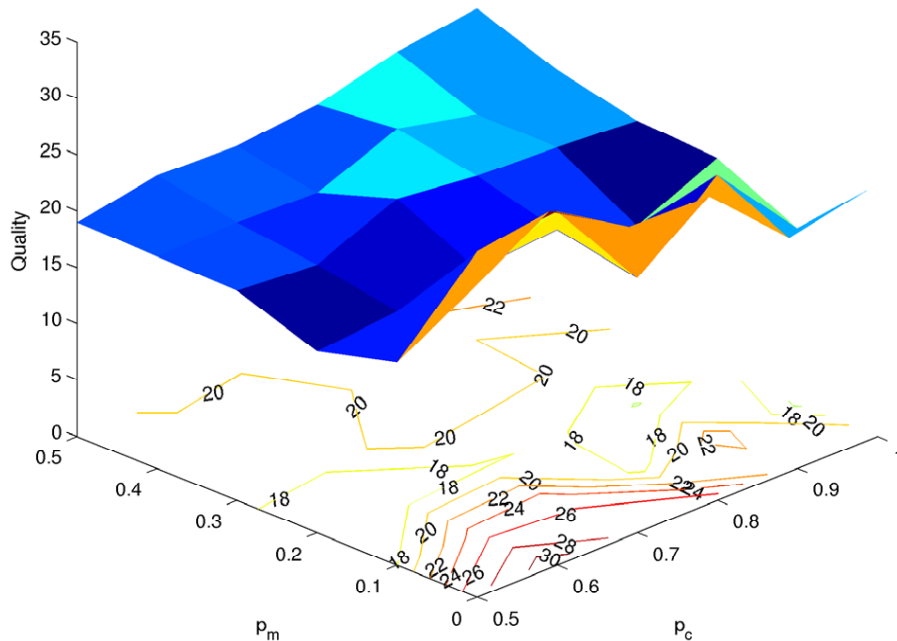


Figure 4.6: Average pre-grasp quality found using BLX-0.5 with type1 sampling, Gaussian Mutation with $K_\sigma = 0.2$ and repetition sampling method

Mutation with $K_\sigma = 0.2$ and repetition sampling method are shown in Fig. 4.6(data from Table. 4.3). We will use type1 sampling with $p_c = 0.8, p_m = 0.1$ for the following tests. This set of p_c, p_m is chosen since it finds the best solution and also determines a good average best solution. It should be recognized that, the search space is different when either the hand or the object is changed. The parameters we chose may not be applicable for other hand-object combinations. Thus the robustness of the GA planner is very important

Table 4.10: BLX-0.5 with type1 sampling, $K_\sigma = 0.3$ with repetition method

$K_\sigma = 0.3$	P_c	P_m															
		0.01			0.1			0.2			0.3			0.4			0.5
		Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
	0.5	22.078(16.879)	7.36	20.712(14.134)	9.011	18.593(14.91)	6.271	18.268(16.316)	1.623	19.596(17.816)	3.134	20.12(16.033)	2.746				
	0.6	34.822(30.357)	5.693	20.936(14.887)	7.757	18.278(16.378)	1.503	18.946(16.481)	3.616	20.178(16.495)	5.439	19.596(17.114)	1.74				
	0.7	24.761(14.936)	8.455	19.953(14.906)	6.279	19.667(15.897)	5.814	20.412(16.475)	4.276	19.903(17.026)	2.18	23.617(18.851)	3.745				
	0.8	18.924(14.224)	5.062	17.314(13.918)	5.606	18.885(15.886)	3.212	20.611(17.892)	3.213	21.308(17.914)	3.15	23.628(18.413)	4.513				
	0.9	20.173(14.264)	7.307	23.358(14.549)	7.228	18.784(15.97)	2.37	19.312(18.477)	0.694	21.144(18.777)	3.559	20.208(18.168)	2.433				
	1	21.945(15.768)	5.951	23.992(14.679)	6.853	20.599(16.429)	5.629	21.864(18.824)	2.001	24.225(22.1)	1.486	24.369(22.478)	1.778				

Table 4.11: BLX-0.5 with type1 sampling, $K_\sigma = 0.4$ with repetition method

$K_\sigma = 0.4$	P_c	P_m															
		0.01			0.1			0.2			0.3			0.4			0.5
		Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
	0.5	29.845(24.688)	4.24	18.703(14.161)	5.712	17.444(16.184)	1.642	20.107(15.825)	4.96	18.76(14.399)	2.628	21.6(18.887)	2.866				
	0.6	28.913(24.033)	3.905	23.671(17.113)	8.128	16.815(14.533)	2.361	20.775(16.39)	4.461	21.666(15.549)	4.66	19.915(17.925)	1.71				
	0.7	28.731(16.114)	8.341	19.691(13.503)	8.124	19.218(14.299)	3.653	23.817(17.387)	4.649	18.144(15.983)	3.532	20.226(19.057)	1.444				
	0.8	26.259(14.343)	11.857	22.869(15.301)	6.93	17.846(15.744)	1.223	22.877(17.612)	5.031	21.948(17.294)	4.078	22.053(19.822)	3.009				
	0.9	21.611(17.935)	5.539	22.752(13.675)	8.307	17.542(14.468)	2.674	19.591(17.629)	1.187	18.549(16.617)	1.171	20.225(19.172)	1.012				
	1	24.443(14.209)	7.416	24.837(14.27)	6.532	22.386(18.297)	5.207	20.744(17.661)	2.318	23.427(20.614)	1.736	25.595(22.158)	3.091				

Table 4.12: Statistics of performance index for different K_σ

	$K_\sigma = 0.2$	$K_\sigma = 0.3$	$K_\sigma = 0.4$
Average	15.64	17.314	16.815
Best	13.678	13.918	13.503
Average of all parameters	20.707	21.141	25.638
Average best of all parameters	17.002	17.073	18.182

in grasp planning. Further tests will be presented in the following sections.

4.3 Performance

The on-line performance and off-line performance proposed by De Jong [36] are used to monitor the evolution of the quality of the grasps over generations and evaluate the convergence performance of GA. On-line performance at generation t is an average of the best from each generation in the past. Off-line performance keeps track of the best solution $Q(best)$ up to each generation and is calculated by taking an average of $Q(best)$ of the past generation at generation t . We run the GA planner with the best parameters we selected from the previous section. The on-line and off-line performance over 5,000 generations as well as the elitists and the $Q(best)$ of each generation are shown in Fig. 4.7. In this plot, we also include the on-line and off-line performance of a simple random planner. For a fair comparison, the random planner is executed for $5,000 * 100 = 500,000$ function evaluations. And the results for both planners are presented in the figure over function evaluations rather than generations.

The figure shows that the population of the planner evolves rapidly over the first 200-250 generations and grows slowly afterwards. In the later stage, the population reaches a stable status. Most individuals are within a small space around the elitist. During this time period, crossover hardly produces new individuals and is not efficient in the search of the solution space. However, the diversity is maintained by mutation so that the rest of the configuration space still gets the chance to be searched over, which is indicated in the blue line. Compared with the simple random planner, we can see that GA is far more efficient. That is the reason we do not want to use a high mutation probability for GA, since that will make it act like a random algorithm.

To better understand the intermediate evolving process, we visualize all the solutions

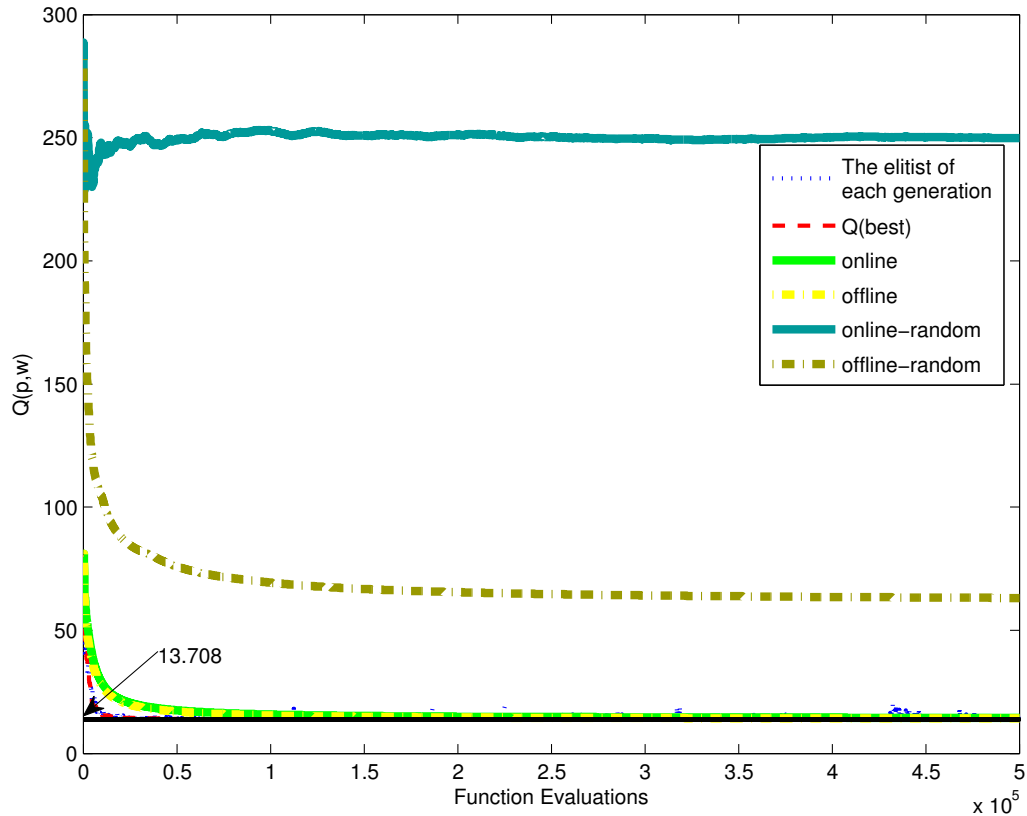


Figure 4.7: The on-line and off-line performance of the genetic algorithm

in the population at every generation. Each chromosome is represented as an arrow, which shows the centering position and orientation of the palm, as illustrated in Fig. 4.4a. The **blue arrow** is located at the center of the palm, perpendicular to the surface. It is a projection of the entire search space on the 6D space of the hand position and orientation. This should give us some idea of the distribution of solutions at each generation, even though it does not cover the eigengrasp posture space. Screenshots of the visualized solutions are taken at 1 per 10 generations. The first 170 generations are shown in Fig. 4.8. We can see that the solutions in the first generation are evenly distributed because they are uniformly chosen at random. Then the population starts to concentrate toward the area which gives better pre-grasps. Most solutions are around the side of the top part of the glass. At this stage, mutation operator is keeping the diversity of the population by randomly changing the genes. Thus the other part of the solution space gets explored with some solutions, which prevents the search from getting stuck in a local minimum.

The intermediate process using different p_c and p_m is illustrated in Fig. 4.9 to Fig. 4.13

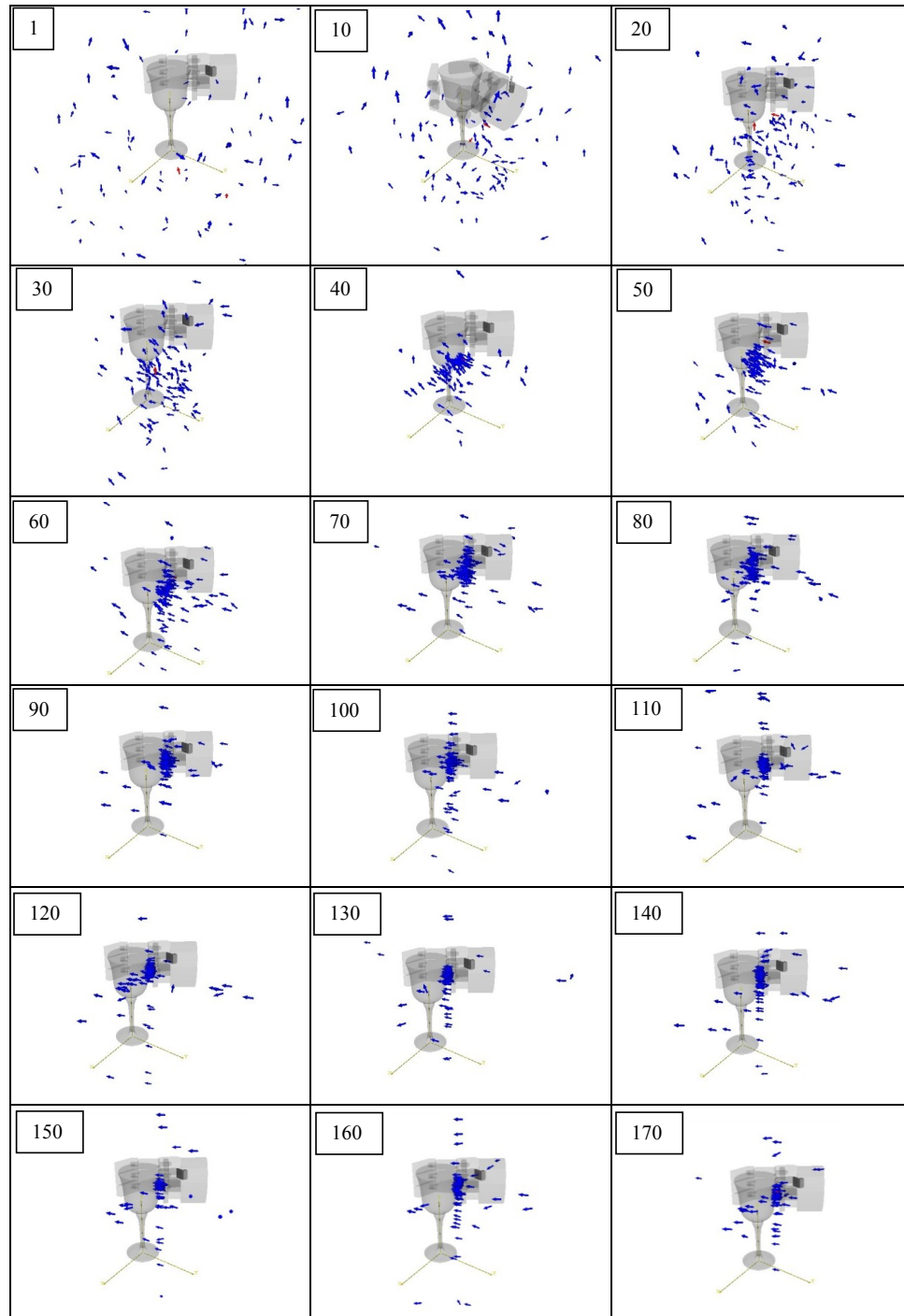


Figure 4.8: Visualization of the intermediate process of the GA planner with $p_c = 0.8$, $p_m = 0.1$. The number on the top left corner denotes the generation. The two **red arrows** show the two elitists in each generation.

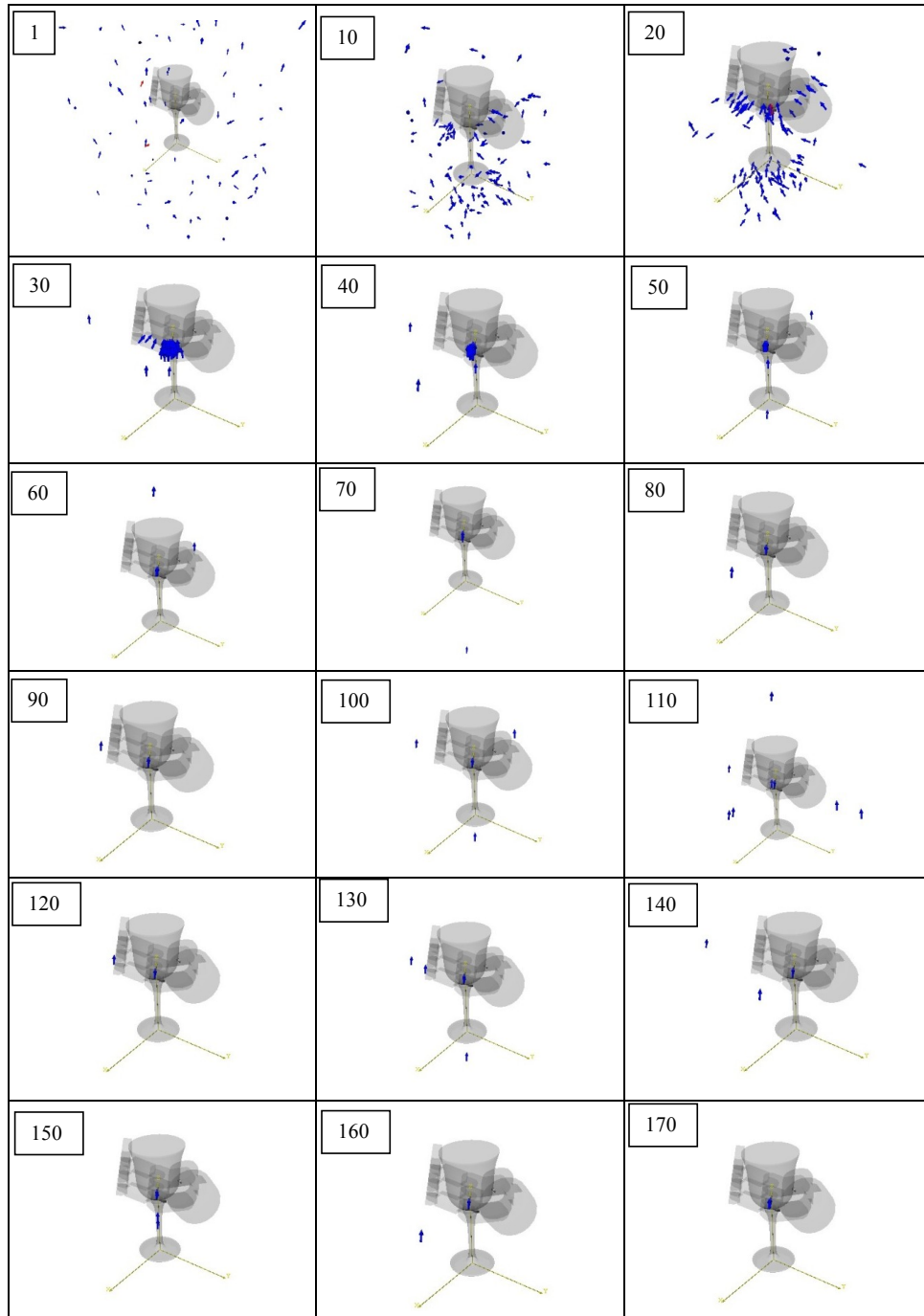


Figure 4.9: Visualization of the intermediate process of the GA planner with $p_c = 0.8$, $p_m = 0.01$.

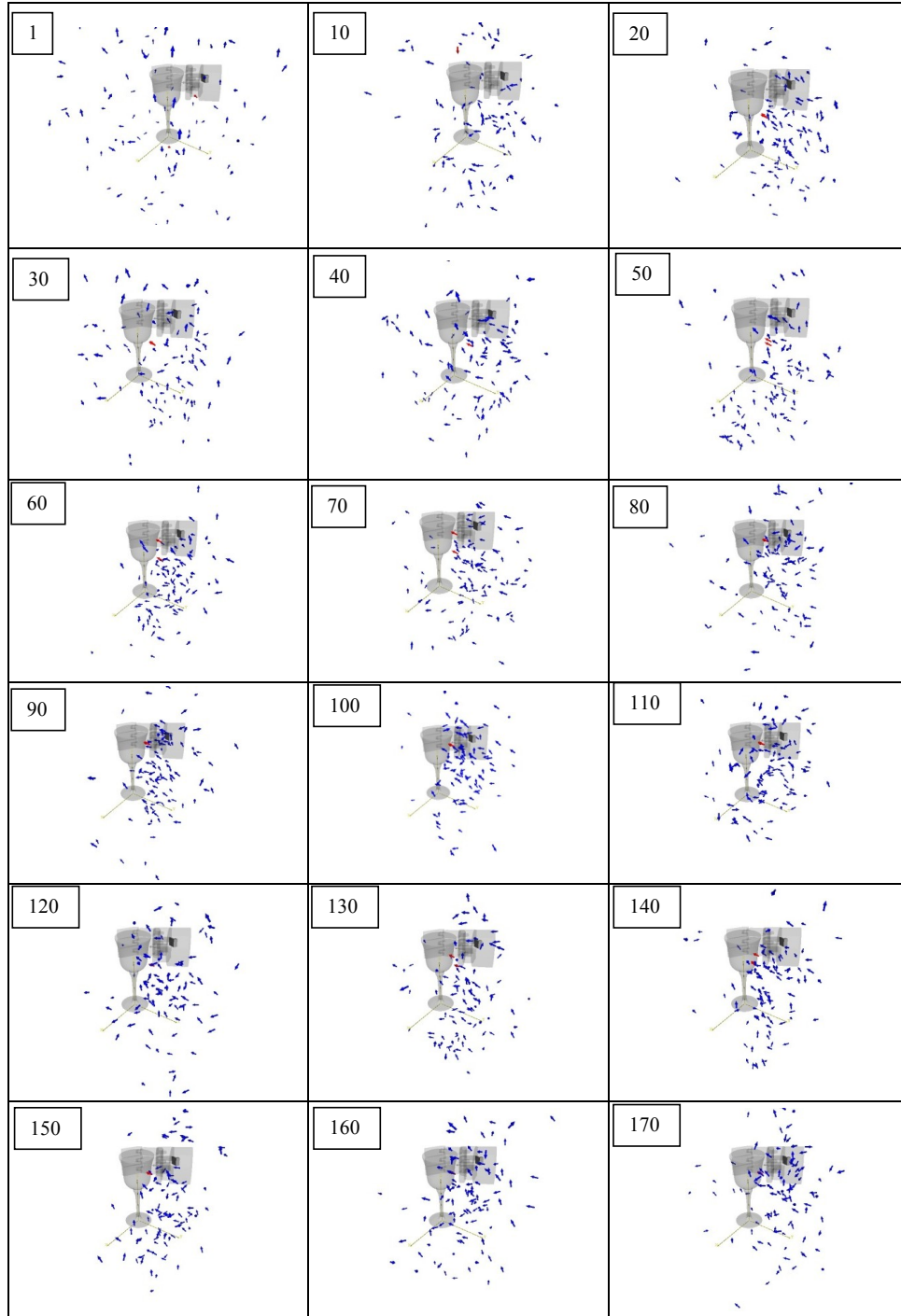


Figure 4.10: Visualization of the intermediate process of the GA planner with $p_c = 0.8, p_m = 0.2$.

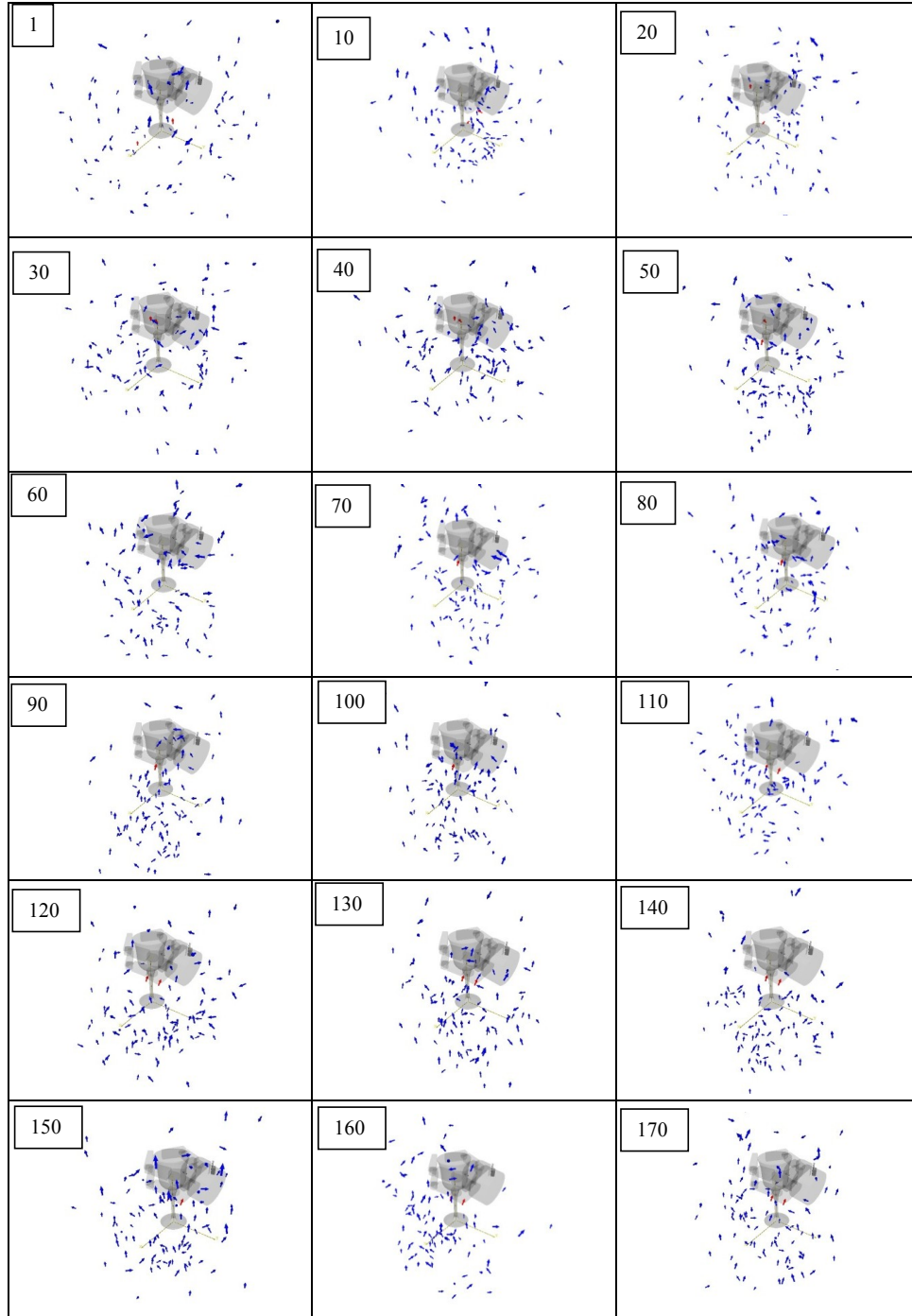


Figure 4.11: Visualization of the intermediate process of the GA planner with $p_c = 0.8, p_m = 0.3$.

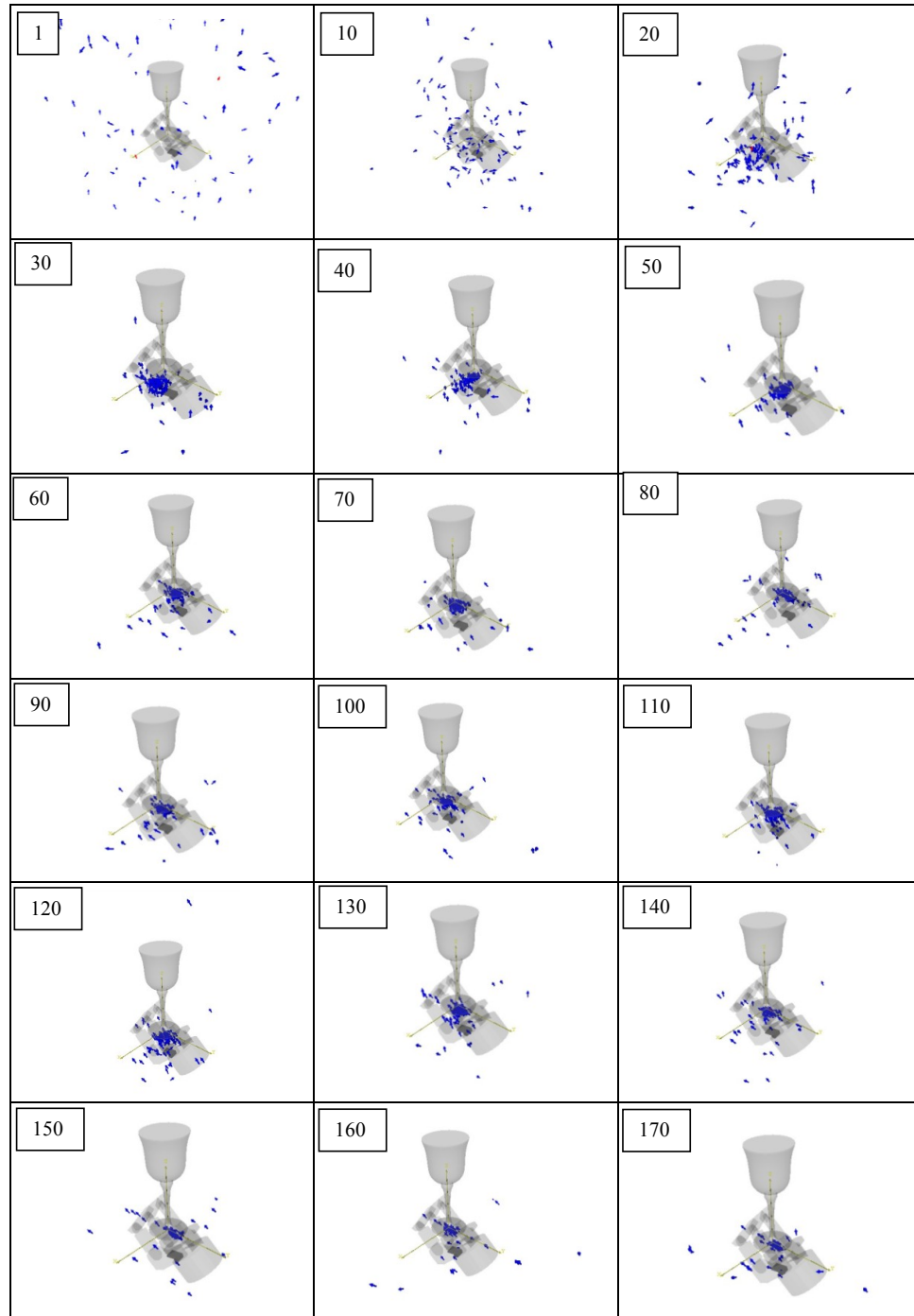


Figure 4.12: Visualization of the intermediate process of the GA planner with $p_c = 0.7, p_m = 0.1$.

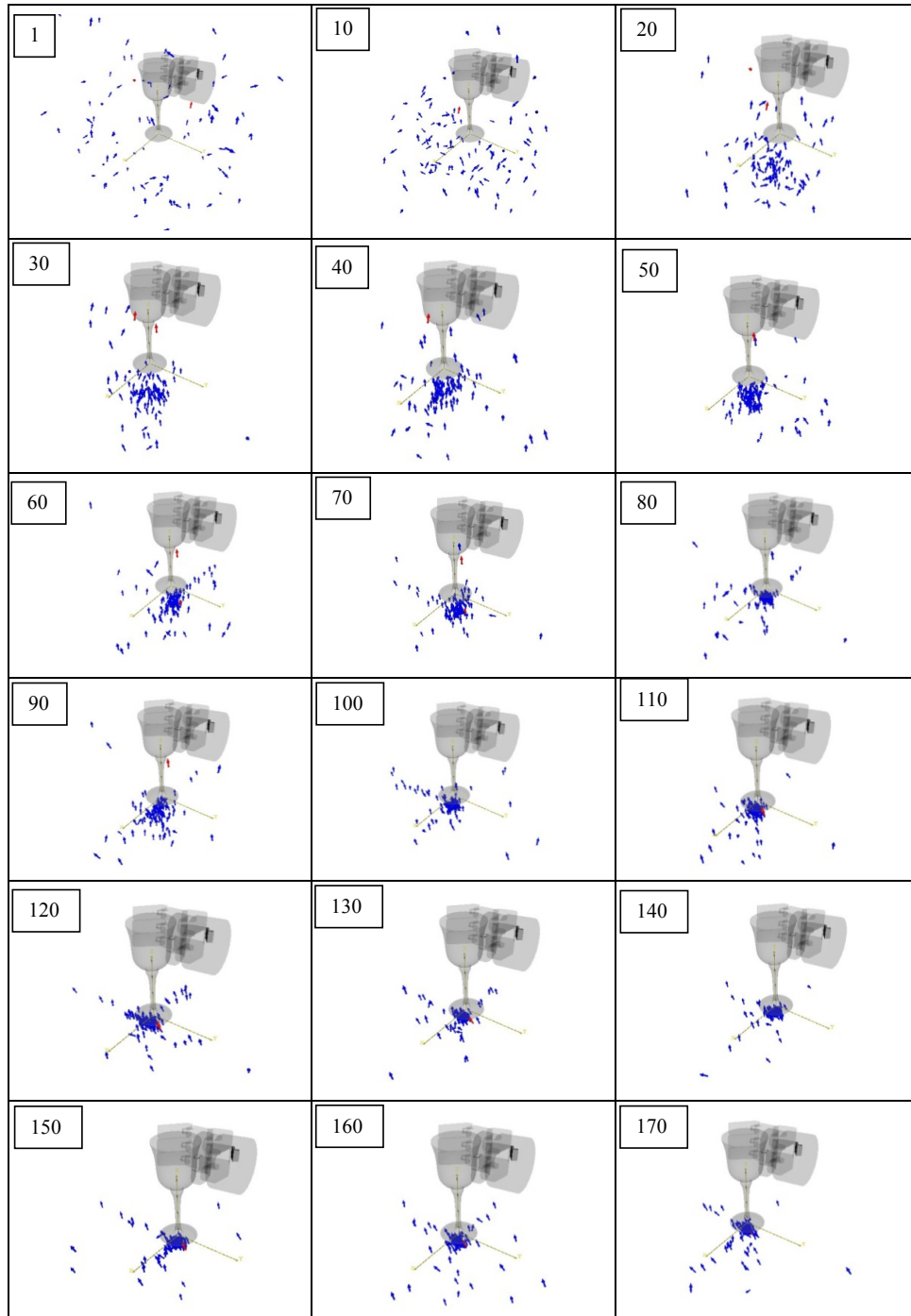


Figure 4.13: Visualization of the intermediate process of the GA planner with $p_c = 0.9, p_m = 0.1$.

to help us study the effect of changing p_c and p_m . When $p_c = 0.8, p_m = 0.01$, premature convergence occurs since the p_m is too low to keep the diversity of the population. The population converges quickly to a local-optimum at generation 60, as shown in Fig. 4.9. And the search has little opportunity to jump out of the local-optimum because very few new solutions are generated. From Fig. 4.10, 4.11, we can see that increasing p_m significantly increases the diversity of the population. When $p_m = 0.2$, we can still observe that the search focuses on a certain area, while when $p_m = 0.3$, the search starts to behave like a random search, with solutions evenly distributed in the search space. Keeping $p_m = 0.1$ and changing p_c (0.7,0.8,0.9), however, does not impact the search as much in terms of the population diversity and convergence speed, as shown in Fig. 4.12, 4.13. Increasing p_c will make the information exchange between solutions faster, but it does not necessarily lead to premature convergence with the existence of mutation operator and a proper p_m .

4.4 Comparison with Simulated Annealing Planner

Simulated Annealing (SA) is another popular algorithm for global optimization. Simulated annealing algorithm and genetic algorithm have some similarities. They are both stochastic methods that are suitable for solving nonlinear, multimodal and high-dimensional optimization problems. SA uses one solution at every iteration. The solution tends to jump to a better state (with lower “energy” , the term for the objective function used in SA), while still accepting higher energy at a lower probability to avoid being stuck at a local optimum. It can be viewed as GA with mutation operator only and the population size is one. Theoretically, global convergence holds for both of them. The proof can be found in [37] for SA and [38, 39, 40] for GA. The practical performance of both algorithms are problem-dependent. In [21], SA was applied on grasp planning to find good pre-grasps. Quantitative results were reported that it achieved a good performance. And it was implemented in the original release of *GraspIt!* simulator. However, since the global optimum pre-grasp is unknown to us, we are still not sure how good a pre-grasp can be. A comparison between GA and SA grasp planners would help us understand their applicability on this problem and have further evaluation on GA’s performance using SA planner as a benchmark. We test both GA and SA planner on the same sets of hand-object combinations and compare their performance. We use the default parameters for SA planner that

implemented in Graspit! simulator. It was stated in [21] that increasing the iterations beyond 70,000 does not improve the performance. Thus the SA planner is performed over 70,000 iterations. And the GA planner is terminated at 5,000 generations, which is 500,000 function evaluations. Each test is repeated five times and the best pre-grasp quality results are averaged. These results are given in Table 4.13. And the best pre-grasps found for both planners on each hand-object combination are shown in Fig. 4.14.

The results show that GA planner outperforms SA planner in most cases in terms of the best pre-grasps found. And it is robust to different hand-object combinations. The average execution time of the two algorithms is listed in Table 4.14. The SA planner performs faster than GA planner. This shows GA is better in finding global optimal pre-grasp with a sacrifice in calculation speed.

In fact, the optimization in grasp planning usually finds its application in off-line use, such as building a database [41]. Thus the difference in execution time is not a big concern in this work. However, for a fair comparison, we want to see what happens if the two algorithms are given the same amount of time. We run the GA planner with the same time as the average time of SA planner listed in Table 4.14. The result is also obtained from five runs on each hand-object combination, shown together with result of SA planner from Table 4.13 in Table 4.15. The better average pre-grasp quality obtained for each hand-object from the two planners is marked in red. GA planner performs better in five out of eight cases. This shows that even given the same amount of time, the performance of GA is comparable to that of SA. We recognize that it is helpful to have GA as another option in the grasp planning task. In applications such as building a database, we can run both GA and SA planners and save the better result into the database.

The final grasps resulting from the pre-grasps in Fig. 4.14 are executed and shown in Fig. 4.15. “e” and “v” refers to the ϵ quality and v quality. The object is set to transparent to show the contact between the hand and object. Note that, the quality of pre-grasp is an informal estimate of the final grasp quality. But they are not equivalent. A good pre-grasp may result in a non *force-closure* grasp and a better pre-grasp may lead to worse final grasp as indicated in Fig. 4.15. We also notice from both figures that the difference in best quality from both planners is usually too minimal to be noticed in terms of the hand posture and position.

Table 4.13: Statistics of the best pre-grasps found from both planners

Planner Type	Glass		Bottle		Mug		Spray Bottle	
	Barrett	Human	Barrett	Human	Barrett	Human	Barrett	Human
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
SA	13.968(13.717)	0.297	12.032(11.382)	0.574	15.317(13.22)	2.155	10.005(8.75)	1.422
GA	13.819 (13.421)	0.689	11.84 (10.774)	1.15	11.773 (11.56)	0.189	9.017 (8.732)	0.312
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
	14.91(14.364)	0.494	12.945 (10.34)	2.706	9.722(8.972)	1.11		

Table 4.14: Execution time of the GA and SA planners

Planner Type	Execution Time(seconds)	
	Barrett	Human Hand
SA (70,000 iterations)	125	183
GA (500,000 function evaluations)	226	272

Table 4.15: Statistics of the best pre-grasps found from both planners given the same running time

Planner Type	Glass		Bottle		Mug		Spray Bottle	
	Barrett	Human	Barrett	Human	Barrett	Human	Barrett	Human
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
SA	13.968(13.717)	0.297	12.032(11.382)	0.574	15.317(13.22)	2.155	10.005(8.75)	1.422
GA(Same Time with SA)	13.89 (13.567)	0.317	11.983 (10.866)	0.998	13.131 (11.951)	1.709	10.681(8.814)	1.172
	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD
	14.91(14.364)	0.494	13.793 (11.976)	1.597	10.801(9.622)	0.686		

Figure 4.14: Best pre-grasps found by GA and SA planners

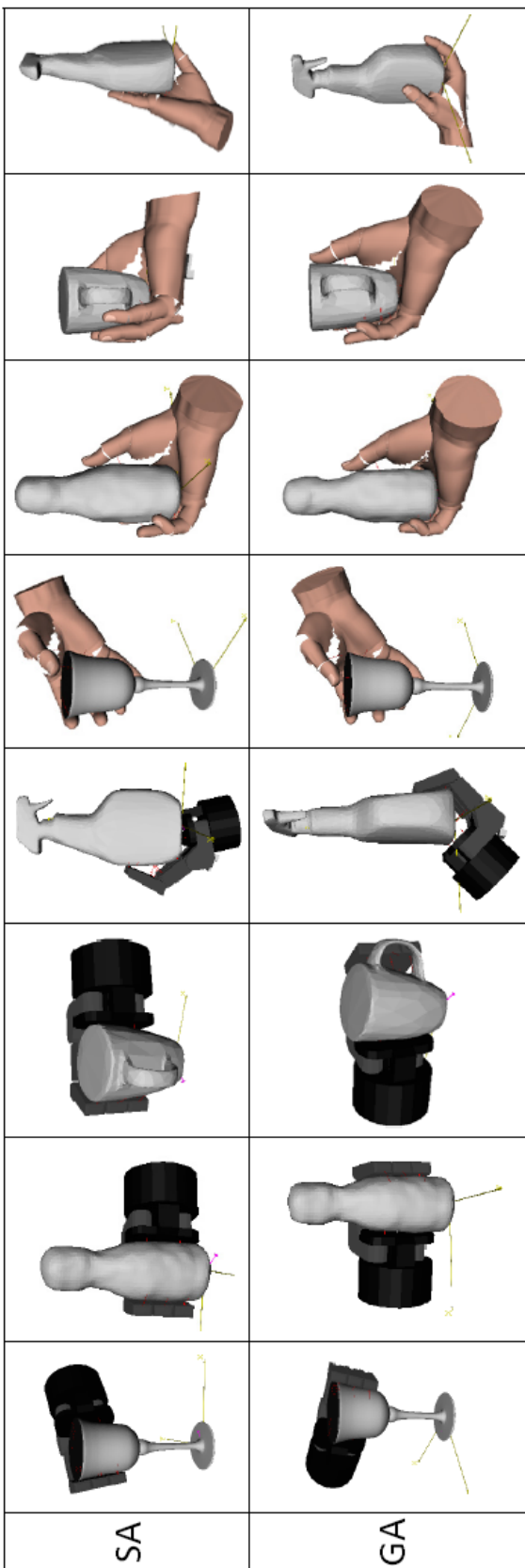
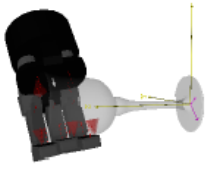
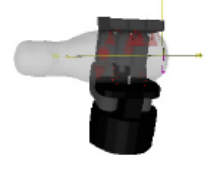
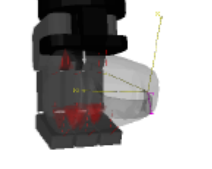

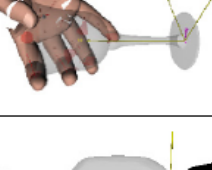
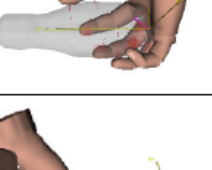
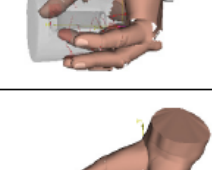

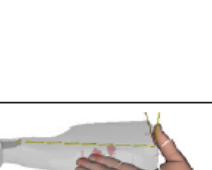
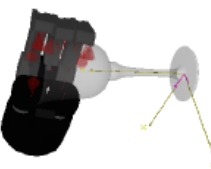
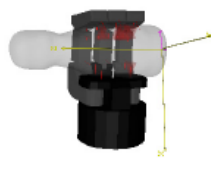
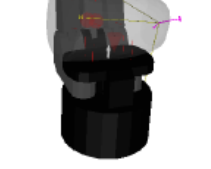

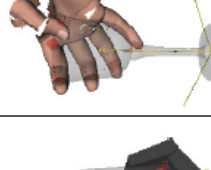
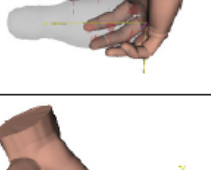
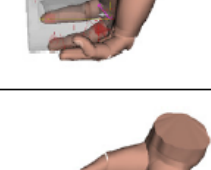
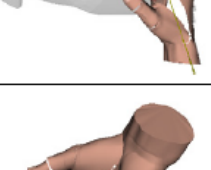



Figure 4.15: The corresponding final grasps and the quality

SA									
SA	e:0.008 v:0.002	e:0.0129 v:0.0074	e: 0.0275 v:0.00679	e:-1 v: 0.000384	e:0.00479 v:0.374	e:0.0913 v:0.247	e:0.244 v:0.682	e:-1 v:0.0824	
GA									
GA	e: 0.0122 v:0.0157	e: 0.0329 v:0.00341	e:-1 v:0.0414	e:-1 v: 0.000285	e:0.06115 v:0.516	e: 0.0916 v: 0.267	e:0.263 v:1.278	e:0.0896 v:0.113	

4.5 Grasp Planning using Another Quality Metric

The pre-grasp quality metric that we used is fast to compute but it has a limitation. It was focused on forming an enveloping grasp around the object, which from a stability standpoint may not be a force-closure grasp. And a pre-grasp with a low (better) quality is not necessarily an enveloping grasp. For instance, if the hand has all the fingers fully opened, and is positioned very close and parallel to the surface of the object (like the spray bottle used in our study), then the hand is so close to the object that it may yield a very good pre-grasp quality, but the hand is not even enveloping the object. Or for the mug, some good pre-grasps with the palm close to the handle may have some problem in stability. This limitation was also indicated from the results of last section.

As mentioned earlier, in off-line applications, the speed of the grasp planning method is not as important as the stability of the grasp that it can find. To find better final grasps, we propose planning the pre-grasps using the quality of the final grasp directly. Instead of using pre-grasp quality as the objective function, each pre-grasp is evaluated with its corresponding final grasp so that the stability of the solution is guaranteed. For every pre-grasp found by the optimization algorithm, we move the hand along the approaching direction defined by the pre-grasp by maximum 50mm until the hand is in contact with the object and then close the fingers to complete the final grasp. If no contact is found in this 50mm distance, the hand is moved back to its initial position and the fingers are closed. Then the ε quality and v quality can be obtained. We use a combination of these two quality measures to evaluate the final grasp:

$$Q_{final} = -(100\varepsilon + 30v) \quad (4.1)$$

This Q_{final} is used as the objective function of the optimization. The negative value is taken so that it is a minimization problem, consistent with the pre-grasp quality metric we used in previous sections. We use ε as the primary quality measure. It gets more weight than v quality. And the scale was chosen just to make the quality large enough so that it is easier to compare.

Since the execution of the final grasp takes a lot of computation power, both GA and SA algorithms run very slow on this problem. We run both planners with a time limit: 1,000 seconds for Barrett Hand and 1,500 seconds for Human Hand. The quality of the pre-grasp found from five runs of both planners was summarized in Table 4.17. And the

Table 4.16: Function evaluations of the GA and SA planners in given time limit

Planner Type	Function Evaluations	
	Barrett(1,000 seconds)	Human Hand(1,500 seconds)
SA	18,142	11,913
GA	16,750	12,488

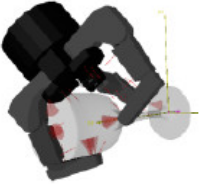
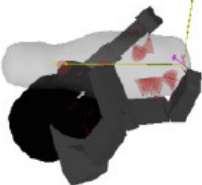

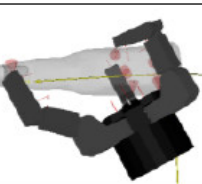
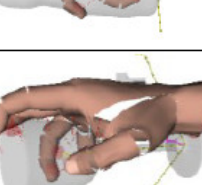
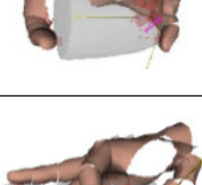
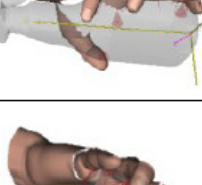
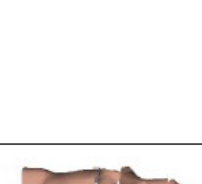
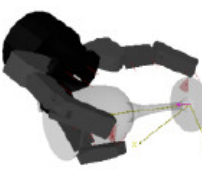
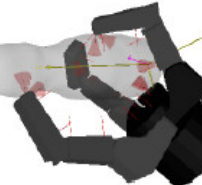
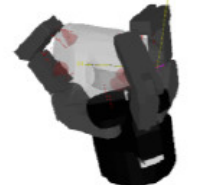
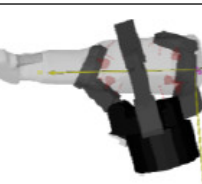
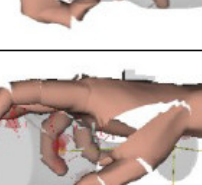
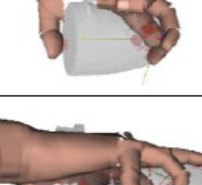
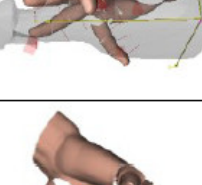

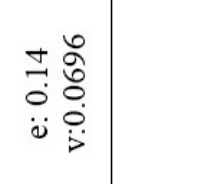
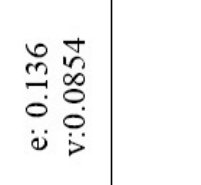
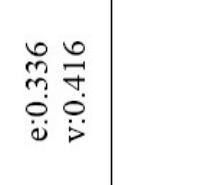
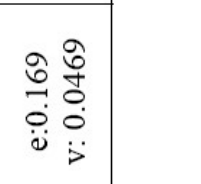
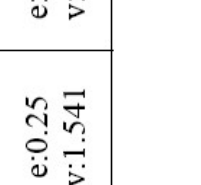
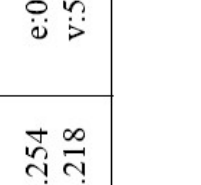
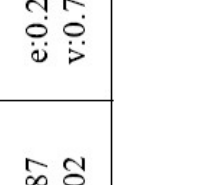
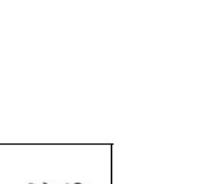







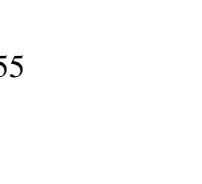
Table 4.17: Statistics of the best pre-grasps found from both planners with Q_{final} as the quality measure. Better one indicated in Blue

Planner Type	Glass			Bottle			Mug			Spray Bottle			Human			
	Barrett	Human	Barrett	Barrett	Human	Barrett	Barrett	Human	Barrett	Human	Barrett	Human				
SA	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD	Average(Best)	STD		
	-13.573(-14.56)	0.87	-64.103(-70.911)	9.278	-12.992(-14.123)	1.176	-58.32(-59.832)	2.489	-39.51(-43.992)	3.902	-198.512 (-202.374)	6.304	-12.852(-15.905)	2.652	-35.337(-41.819)	7.244
GA	-15.576 (-16.659)	1.073	-66.453 (-71.21)	7.208	-13.918 (-16.147)	1.969	-59.56 (-61.986)	3.65	-42.266 (-46.071)	4.749	-194.499(-214.712)	29.574	-14.462 (-18.262)	3.291	-41.474 (-49.492)	10.921

Figure 4.16: Best pre-grasps found by GA and SA planners with Q_{final} as the quality measure



Figure 4.17: The corresponding final grasps obtained with Q_{final} as the quality measure

SA		e:0.126 v:0.0639		e:0.129 v:0.0398		e:0.323 v:0.389		e:0.144 v:0.0498		e:0.293 v:1.386		e:0.273 v:1.084		e:0.559 v:4.883		e:0.275 v:0.476
SA		e:0.126 v:0.0639		e:0.129 v:0.0398		e:0.323 v:0.389		e:0.144 v:0.0498		e:0.25 v:1.541		e:0.254 v:1.218		e:0.587 v:5.202		e:0.272 v:0.743
GA		e:0.14 v:0.0696		e:0.136 v:0.0854		e:0.336 v:0.416		e:0.169 v:0.0469		e:0.25 v:1.541		e:0.254 v:1.218		e:0.587 v:5.202		e:0.272 v:0.743
GA		e:0.14 v:0.0696		e:0.136 v:0.0854		e:0.336 v:0.416		e:0.169 v:0.0469		e:0.25 v:1.541		e:0.254 v:1.218		e:0.587 v:5.202		e:0.272 v:0.743

best pre-grasps and their corresponding final grasps are shown in Fig. 4.16 and 4.17. Both planners are able to find *force-closure* grasps. GA clearly outperforms SA on all the objects when using Barrett Hand. For grasp planning with Human Hand, the performance of GA and SA are close. GA is slightly better with better solutions for three of the four objects tested.

The average of the function evaluations that the planners were able to perform within the given time limit are listed in Table 4.16. The total function evaluations of both planners are very close. As discussed in previous sections, GA evolves very fast to a good solution in the first 200 generations. For grasp planning using Barrett Hand, GA was able to finish about 165 generations, enough to get a *force-closure* solution with better stability than SA. When using Human Hand, more DOFs were involved in the optimization. On average, GA was able to finish about 125 generations. Within these number of generations, GA performed a little better than SA but does not show as big an advantage as in the case of Barrett Hand.

Grasp planning with this final grasp quality metric (4.1) is able to yield pre-grasps that result in *force-closure* final grasps. This method takes much longer time than the method using (2.6) as the quality metric, since the evaluation of each solution involves executing the final grasp. However, the final grasp obtained are more stable than those using (2.6) as the quality measure, because the objective function (4.1) is directly related to the stability of the final grasp.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we proposed applying genetic algorithm to vision-based grasp planning on 3D objects. This algorithm can be used on 3D object in arbitrary shapes and any robot hand with the assumption that the objects are rigid and a model of the object was already obtained from the image sensor.

We firstly reviewed the related work and basic knowledge in the domain of robot grasping. *Heuristic grasp planning approach* was used because of its better applicability in practice. The planning process was divided into two steps. The first step was to find a *pre-grasp* that is more likely to yield a stable final grasp. And in the second step, the grasp is completed by moving the hand toward the object until getting in contact and closing the fingers to perform a *power grasp*. The grasp planning was formulated as an optimization problem with the objective of finding a good *pre-grasp*. The quality measures of *pre-grasp* and final grasp were introduced.

The concepts essential in building a genetic algorithm based grasp planner were introduced. The operators and parameters of genetic algorithm were described in detail. They were known to have a crucial impact on the performance and the best parameters may depend on specific problems. To better understand this optimization problem in order to choose the proper operator, the solution landscape were examined as well as the bias introduced by the crossover operator with different sampling methods.

We implemented this algorithm using *GraspIt!* simulator. Barrett Hand grasping a glass was used as a demonstration for choosing parameters and evaluating the performance. Extensive tests were carried out to choose the appropriate parameters for grasp planning, including the population size, probability of crossover and mutation, different sampling methods for crossover and mutation, the α of the BLX- α operator, and the variance used in Gaussian Mutation operator. A set of parameters that give the best performance were chosen for further tests. The quantitative results on a number of hand-object combinations

indicate that genetic algorithm can be used to obtain a good pre-grasp posture and is robust to different solution space introduced by different hand or object. Compared to the grasp planner based on simulated annealing algorithm, the GA planner was superior in most cases in terms of the average best solution obtained. Although the time issue is not the main concern of this work (since neither of these methods are fast enough for real-time application), we included the execution time for the complete details of the information. The GA planner worked slower when both planners were given enough time to reach a stable state. Then we ran GA planner with the same amount of time as SA planner for a fair comparison. The performance of GA was comparable with that of SA. To overcome the limitation of the pre-grasp quality, we investigated the possibility of using another quality metric for this problem. In the optimization process, each pre-grasp was evaluated based on its corresponding final grasp with a stability quality measure consisting of ϵ quality and ν quality. With this quality metric as the objective function, GA and SA planners were both able to find force-closure grasps for each hand-object set. And overall, GA outperforms SA given the same amount of time with this quality measure.

5.2 Future Work

Future work will be focused on improving the performance of the GA planner. The parameter tuning process in this thesis was basically a brute-force method. We would like to try some advanced methods for parameter tuning to see how they work, like Factorial Experiment [42] and Nelder-Mead algorithm.

The robustness is very important for GA in grasp planning since the search space changes with the hand and object. We will apply adaptive methods like the one proposed in [43] to tune the parameters p_c and p_m on-the-fly. And since GA is intrinsically parallel, it would be interesting to investigate the performance of a parallel GA, utilizing the power of multiple-core computers. Furthermore, considering the resemblance between SA and GA, hybrid methods which combine them [44] to take the best from both worlds may largely improve the performance. Similar hybrid methods also include combining GA and Nelder-Mead algorithm [45] to utilize the global exploration ability of GA and the local search ability of Nelder-Mead.

Bibliography

- [1] R. M. Murray, Z. Li, and S. S. Sastry, “A mathematical introduction to robotic manipulation,” 1994.
- [2] A. T. Miller, “Graspit!: A versatile simulator for robotic grasping,” Ph.D. dissertation, Columbia University, New York, USA, June 2001.
- [3] D. Goeger, N. Ecker, and H. Woern, “Tactile sensor and algorithm to detect slip in robot grasping processes,” in *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, Feb. 2009, pp. 1480–1485.
- [4] J. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. Kuchenbecker, “Human-inspired robotic grasp control with tactile sensing,” *Robotics, IEEE Transactions on*, vol. 27, no. 6, pp. 1067–1079, Dec. 2011.
- [5] J. Platt, R., A. Fagg, and R. Grupen, “Nullspace composition of control laws for grasping,” in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2, 2002, pp. 1717–1723 vol.2.
- [6] D. Bowers and R. Lumia, “Manipulation of unmodeled objects using intelligent grasping schemes,” *Fuzzy Systems, IEEE Transactions on*, vol. 11, no. 3, pp. 320–330, June 2003.
- [7] A. Miller, S. Knoop, H. Christensen, and P. Allen, “Automatic grasp planning using shape primitives,” in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, Sept. 2003, pp. 1824–1829 vol.2.
- [8] A. Chella, H. Dindo, F. Matraxia, and R. Pirrone, “Real-time visual grasp synthesis using genetic algorithms and neural networks,” in *Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence on AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, ser. AI*IA '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 567–578.
- [9] N. Daoud, J. Gazeau, S. Zeghloul, and M. Arsicault, “A fast grasp synthesis method for online manipulation,” *Robotics and Autonomous Systems*, vol. 59, no. 6, pp. 421–427, 2011.
- [10] S. Mannepilli, A. Dutta, and A. Saxena, “A multi-objective ga based algorithm for 2d form and force closure grasp of prismatic objects,” *I. J. Robotics and Automation*, vol. 25, no. 2, 2010.
- [11] C. Sangkhavijit, N. Niparnan, and P. Chongstitvatana, “Computing 4-fingered force-closure grasps from surface points using genetic algorithm,” in *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*, Dec. 2006, pp. 1–5.

- [12] H. Mesgari, F. Cheraghpour, and S. Moosavian, “Application of mag index for optimal grasp planning,” in *Mechatronics and Automation (ICMA), 2011 International Conference on*, Aug. 2011, pp. 2171–2176.
- [13] J. Fernandez and I. Walker, “Biologically inspired robot grasping using genetic programming,” in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 4, May 1998, pp. 3032–3039 vol.4.
- [14] J. R. Napier, “The prehensile movements of the human hand,” *J Bone Joint Surg Br*, vol. 38-B, no. 4, pp. 902–913, Nov. 1956.
- [15] X.-Y. Zhang, Y. Nakamura, K. Goda, and K. Yoshimoto, “Robustness of power grasp,” in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, May 1994, pp. 2828–2835 vol.4.
- [16] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, “Learning to grasp objects with multiple contact points.” in *ICRA. IEEE*, 2010, pp. 5062–5069.
- [17] N. S. Pollard, “Parallel methods for synthesizing whole-hand grasps from generalized prototypes,” MIT, Tech. Rep., 1994.
- [18] C. Ferrari and J. Canny, “Planning optimal grasps,” in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, May 1992, pp. 2290–2295 vol.3.
- [19] A. T. Miller and A. T. Miller, “Graspit!: A versatile simulator for robotic grasping,” *IEEE Robotics and Automation Magazine*, vol. 11, pp. 110–122, 2004.
- [20] M. Santello, M. Flanders, and J. F. Soechting, “Postural Hand Synergies for Tool Use,” *Journal of Neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, Dec. 1998.
- [21] M. Ciocarlie and P. Allen, “Hand posture subspaces for dexterous robotic grasping,” *The International Journal of Robotics Research*, vol. 28, pp. 851–867, 07/2009 2009.
- [22] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975.
- [23] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [24] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. New York, NY, USA: Springer-Verlag New York, Inc., 1994.
- [25] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.
- [26] K. Sastry and D. E. Goldberg, “Modeling tournament selection with replacement using apparent added noise,” in *Intelligent Engineering Systems Through Artificial Neural Networks. Proceedings of the Conference ANNIE 2001*, vol. 2, 2001, pp. 129–134.

- [27] Eshelman, “The CHC Adaptive Search Algorithm : How to Have Safe Search When Engaging in Nontraditional Genetic Recombination,” *Foundations of Genetic Algorithms*, pp. 265–283, 1991.
- [28] H. Pohlheim, “GEATbx - genetic and evolutionary algorithm toolbox for use with matlab. <http://www.geatbx.com/>,”
- [29] Y. Yoon, Y.-H. Kim, A. Moraglio, and B.-R. Moon, “A theoretical and empirical study on unbiased boundary-extended crossover for real-valued representation,” *Inf. Sci.*, vol. 183, no. 1, pp. 48–65, Jan. 2012.
- [30] S. Tsutsui and D. E. Goldberg, “Search space boundary extension method in real-coded genetic algorithms,” *Information Sciences*, pp. 133–3, 2001.
- [31] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA Workshop on Open Source Software*, 2009.
- [32] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, “The quickhull algorithm for convex hulls,” *ACM TRANSACTIONS ON MATHEMATICAL SOFTWARE*, vol. 22, no. 4, pp. 469–483, 1996.
- [33] M. Ciocarlie, C. Pantofaru, K. Hsiao, G. Bradski, P. Brook, and E. Dreyfuss, “A side of data with my robot: Three datasets for mobile manipulation in human environments,” *IEEE Robotics & Automation Magazine, Special Issue: Towards a WWW for Robots*, vol. 18, June 2011.
- [34] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C (2nd ed.): the art of scientific computing*. New York, NY, USA: Cambridge University Press, 1992.
- [35] K. Sastry, “Single and multiobjective genetic algorithm toolbox in c++,” University of Illinois at Urbana-Champaign, Urbana IL, Tech. Rep. IlliGAL Report No. 2007016, 2007.
- [36] K. A. De Jong, “An analysis of the behavior of a class of genetic adaptive systems.” Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 1975.
- [37] C. J. P. Blisle, “Convergence theorems for a class of simulated annealing algorithms on rd,” *Journal of Applied Probability*, vol. 29, no. 4, pp. pp. 885–895, Dec. 1992.
- [38] R. F. Hartl, “A global convergence proof for a class of genetic algorithms,” 1990.
- [39] D. Greenhalgh and S. Marshall, “Convergence criteria for genetic algorithms,” *SIAM J. Comput.*, vol. 30, no. 1, pp. 269–282, Apr. 2000.
- [40] G. Rudolph, “Convergence analysis of canonical genetic algorithms,” *Neural Networks, IEEE Transactions on*, vol. 5, no. 1, pp. 96 –101, Jan 1994.

- [41] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *IEEE Intl. Conf. on Robotics and Automation*, 2009.
- [42] L. H. Lee and Y. Fan, "An adaptive real-coded genetic algorithm," *Applied Artificial Intelligence*, vol. 16, no. 6, pp. 457–486, 2002.
- [43] M. Srinivas and L. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 24, no. 4, pp. 656–667, Apr. 1994.
- [44] S. W. Mahfoud and D. E. Goldberg, "Parallel recombinative simulated annealing: A genetic algorithm," *Parallel Computing*, vol. 21, no. 1, pp. 1–28, 1995.
- [45] R. Chelouah and P. Siarry, "Genetic and nelder-mead algorithms hybridized for a more accurate global optimization of continuous multim minima functions," *European Journal of Operational Research*, vol. 148, no. 2, pp. 335–348, July 2003.